



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Yan Feng

Battery Assembly Execution System

Information Technology
2018

FOREWORD

As a joint degree student, I have spent two years studying at the VAMK University of Applied Sciences. This thesis symbolizes the culmination of all my hard work. Here, I would like to express my appreciation to everyone who helped and guided me during this time. It is thanks to their support that I remained optimistic whenever I faced challenges.

First of all, Dr. Yang Liu, my supervisor, provided invaluable suggestions and instructions for my thesis. Without his help, my final project could not have been realized according to the clear plan he helped me formulate. Thus, I modified and optimized my thesis under his guidance, and I wish to express my profound appreciation for this assistance.

Furthermore, I also wish to express appreciation to all the teachers and staff who educated and provided me with vast professional knowledge and practical skills, including Mr. Timo Kankaanpaa, Dr. Chao Gao, Dr. Ghodrat Moghadampour and everyone else who assisted me with kindness and patience.

Finally, it is impossible for me not to thank my parents. If they had not agreed with my decision to come to Vaasa, everything would be different. Whenever I encountered obstacles, it was their comfort that gave me the courage and strength to persevere. I also want to thank my friends whom I met here in Finland. I wish them all the best for the future.

ABSTRACT

Author	Yan Feng
Title	Battery assembly execution system
Year	2018
Language	English
Pages	79
Name of Supervisor	Yang Liu

With the increased popularity of manufacturing execution systems, there is an urgent need for such a system which connects the relevant robot and enterprise resource planning system. This project uses this system in order to complete the assembly and picking up of batteries in factories. The main work is divided into five modules.

The first module is concerned with the enterprise resource planning system, Odoo, which is used to acquire several manufacturing orders and handle them. Then, three modules related to the different robots execute their tasks after receiving the order. In the ABB robot's program, the order information is assigned. After the ABB robot finishes the mission, the mobile robot delivers the assembled battery module. The CMMO controller is responsible for picking up the battery module. While the mobile robot moves in the working range, the manufacturing order status is set as done. The final module relates to strategy. Based on the original module organizational design, two creative features are proposed and researched under the supervisor's guidance. In the future, they could help optimize the application.

This project was completed using Java on a PC equipped with Windows 10. The following software packages support this project: Odoo, RobotStudio, MobilePlanner, MobileEyes and Festo configuration tool.

In all, this project is designed for factory use and improving operator efficiency.

Keywords Odoo, ABB robot, Omron robot, CMMO controller, Strategy design

CONTENTS

TIIVISTELMÄ

ABSTRACT

1	INTRODUCTION	11
1.1	Purpose.....	11
1.2	Overview Structure	11
1.3	Introduction to Adept Lynx	11
1.3.1	Mobile robot.....	11
1.3.2	Adept Lync key features	12
1.3.3	Adept Lynx Software	13
1.3.3.1	MobilePlanner	13
1.3.3.2	MobileEyes	14
1.3.4	Adept Lynx command language	14
1.4	Introduction to motor controller CMMO-ST	14
1.4.1	Motor controller	14
1.4.2	CMMO-ST specifications	15
1.4.3	Software for configuration and commissioning	15
1.4.3.1	Web service	15
1.4.3.2	FCT (Festo configuration tool).....	16
1.5	Introduction to the Odoo system.....	16
1.5.1	ERP system	16
1.5.2	Odoo key features	17
1.5.3	Odoo programming	18
1.6	ABB robot introduction	19
1.6.1	ABB's IRB 1200	19
1.6.2	ABB robot software	20

1.6.3	ABB Robot web service.....	21
2	OVERALL STRUCTURE	23
2.1	Project structure	23
2.2	Module description	23
2.3	Project flowchart.....	25
3	ODOO MODULE	27
3.1	Environment configuration	27
3.1.1	Installation.....	27
3.1.2	Component Extension	28
3.2	Odoo Programming.....	31
3.2.1	Odoo Java API	31
4	ABB ROBOT MODULE	38
4.1	Environment configuration	38
4.1.1	Demo deployment	38
4.1.2	Parameter creation.....	39
4.2	ABB robot programming.....	40
4.2.1	ABB robot web services.....	40
4.2.2	RobotWare services	41
4.2.3	Operations on RAPID data.....	42
5	OMRON ROBOT MODULE.....	44
5.1	Environment configuration	44
5.1.1	Initial map creation	44
5.1.2	Editing maps by adding objects	45
5.1.3	Working with Tasks	47
5.2	Omron robot programming	49
5.2.1	Working with Putty	50

5.2.2	ARCL Implementation.....	52
6	CMMO CONTROLLER MODULE.....	54
6.1	Environment configuration.....	54
6.1.1	Device configuration.....	54
6.1.2	Parameter setting.....	57
6.1.3	Enabling the Web server.....	58
6.2	CMMO controller programming.....	59
6.2.1	Web server queries.....	60
7	STRATEGY MODULE.....	62
7.1	Initial programming.....	62
7.2	Future features.....	63
7.2.1	Scheduling Algorithm Based on Priority Table.....	63
7.2.1.1	Deadline/value priority table design.....	64
7.2.1.2	Deadline/value priority table design implement.....	66
7.2.2	Going through all points in the shortest path.....	69
7.2.2.1	TSP problem.....	69
7.2.2.2	Greedy algorithm.....	70
7.2.2.3	Greedy algorithm for solving TSP problems.....	71
8	RESULTS.....	72
8.1	Preparatory work.....	72
8.2	Implementation.....	72
8.2.1	Starting the project.....	72
8.2.2	Project Process.....	72
9	SUMMARY.....	77
10	REFERENCES.....	78

LIST OF ABBREVIATIONS

AMR	Autonomous Mobile Robot
AGV	Autonomous Guided Vehicle
IDE	Integrated Development Environment
AIV	Autonomous Indoor Vehicle
ARCL	Advanced Robotics Command Language
PC	Personal Computer
MES	Manufacturing Execution System
FCT	Festo Configuration Tool
CRM	Customer Relationship Management
RPC	Remote Procedure Calls
HTTP	HyperText Transfer Protocol
ERP	Enterprise Resource Planning
REST	Representational State Transfer
API	Application Programming Interface
TSP	Travelling Salesman Problem

LIST OF FIGURES

Figure 1. Adept Lynx/11/	13
Figure 2. MobilePlanner	14
Figure 3. CMMO-ST/14/ product overview	15
Figure 4. FCT panels	16
Figure 5. ERP modules/4/	17
Figure 6. Odoo overview	18
Figure 7. OpenERP/13/ architecture.....	19
Figure 8. RobotStudio panels	21
Figure 9. robot web service/8/ architecture	22
Figure 10. Whole project architecture	23
Figure 11. Whole project flowchart	25
Figure 12. Adding the new database.....	29
Figure 13. Adding the specified module.....	29
Figure 14. Adding product materials	30
Figure 15. Adding manufacturing orders.....	30
Figure 16. Odoo module achitecture	31
Figure 17. Configuration code.....	32
Figure 18. Configuration in the property file.....	32
Figure 19. Authentication in the property file and its code	33
Figure 20. Preparation of the calling methods in the property file and its code.....	34
Figure 21. Parameters of the searching method in the property file.....	34
Figure 22. Searching method code	34
Figure 23. Parameters of the reading method in the property file	35
Figure 24. Reading method code	36
Figure 25. Parameters of the writing method in the property file	37
Figure 26. Writing method code	37

Figure 27. Recent file in RobotStudio	38
Figure 28. Current program structure	39
Figure 29. Current program order information.....	39
Figure 30. ABB robot module architecture	40
Figure 31. The subset of services and resources in robot web services	41
Figure 32. The subset of services and resources in RobotWare services	42
Figure 33. GET rapid symbol data	43
Figure 34. Updating a RAPID variable	43
Figure 35. Marking the map goals.....	46
Figure 36. Drawing the advanced areas and lines	47
Figure 37. Details in the move task	48
Figure 38. Omron robot module architecture	49
Figure 39. Server Information in MobilePlanner	51
Figure 40. Content in putty.....	51
Figure 41. Initialization of the related variables(1)	52
Figure 42. Initialization of the related variables (2)	52
Figure 43. Code for patrolling the route once (1).....	53
Figure 44. Code for patrolling the route once (2).....	53
Figure 45. Code for patrolling the route once (3).....	53
Figure 46. Three connections with the CMMO controller	54
Figure 47. Controller configuration.....	55
Figure 48. Axis Motor Unit configuration.....	56
Figure 49. Valve profile selection	56
Figure 50. Homing Switch with index negative	58
Figure 51. Uploading and downloading setting.....	58
Figure 52. Device control enabling	59
Figure 53. CMMO controller module architecture.....	59

Figure 54. Homing query code	60
Figure 55. Homing Web page.....	60
Figure 56. Stop query code.....	61
Figure 57. Stop Web page	61
Figure 58. Strategy module architecture.....	62
Figure 59. Priority table design for EDV and VED.....	66
Figure 60. Code for EDV table design (1).....	67
Figure 61. Code for EDV table design (2).....	67
Figure 62. Code for EDV table design (3).....	67
Figure 63. Code for EDV table design (4).....	68
Figure 64. Code for EDV table design (5).....	68
Figure 65. Code for EDV table design (6).....	68
Figure 66. Code for EDV table design (7).....	68
Figure 67. Code for EDV table design (8).....	69
Figure 68. Solution to the TSP problems	70
Figure 69. Solution in the Greedy algorithm.....	71
Figure 70. Starting the project	72
Figure 71. Result capture (1)	73
Figure 72. Result capture (2)	73
Figure 73. Mobile robot environment.....	74
Figure 74. Omron robot reaction	74
Figure 75. Result capture (3)	75
Figure 76. CMMO controller reaction.....	75
Figure 77. Result capture (4)	75
Figure 78. Selected order status.....	76

1 INTRODUCTION

1.1 Purpose

Nowadays, the problems with existed MES systems is that they only accept the same version machine and lack interface capabilities for the specified ERP system. In this way, this thesis aims to illustrate MES system implementation based on the commonality of the Odoo—ERP system and then the controls of Omron and ABB robots. During this process, these machines cooperate and tackle a set of tasks so as to realize battery assembly.

1.2 Overview Structure

This thesis is divided into five components: the ABB robot module, Omron robot module, CMMO controller module, strategy module and Odoo module. The chapter two provides an overview of this project's structure, detailing the individual modules and whole process work flow. In chapter three, the Odoo system is not only researched and developed, but the methods for determining manufacturing module order and updating order status are also considered. In chapter four, the ABB robot program is deployed in RobotStudio, with the method for assigning order outlined. Chapter five focuses on the Omron mobile robot, detailing the means of creating, modifying and adding tasks to the scanned map. Moreover, this module is also responsible for controlling the mobile robot and directing its actions. Chapter six outlines the CMMO controller, discussing connection selection and its operation process. Chapter seven briefly describe the original version before introducing two innovative points. The final chapter sums up the thesis and provides a conclusion.

1.3 Introduction to Adept Lynx

1.3.1 Mobile robot

A mobile robot is an automatic machine capable of locomotion. In other words, it can move around in its environment and is not fixed to a single physical location.

Mobile robots can be autonomous, meaning they can navigate an uncontrolled environment without the need for physical or electro-mechanical guidance devices. Alternatively, mobile robots can rely on guidance devices for traveling pre-defined

navigation routes in relatively controlled space (AGV). In contrast, industrial robots are usually relatively stationary, consisting of a jointed arm (multi-linked manipulator) and a gripper assembly (or end effector), attached to a fixed surface.

1.3.2 Adept Lynx key features

The Adept Lynx, a brand small new mobile robot, is a self-navigating Autonomous Indoor Vehicle (AIV) designed for dynamically moving materials in challenging environments including confined passageways as well as dynamic and populated locals.

Unlike traditional autonomously guided vehicles (AGVs), Lynx requires no facility modifications, such as floor magnets or navigational beacons, saving users up to 15% in deployment costs.

Lynx includes Adept's proprietary software and controls allowing the intelligent navigation of people and unplanned obstacles, which would otherwise render traditional AGVs incapacitated. Moreover, it can be programmed and functional within a day.

It is designed for developers, integrators, and end-users because its system can be customized for various applications and payloads. Furthermore, manufacturing, warehousing, clean tech, and laboratories are ideal environments for Lynx.

Its specifications are as follows:

- Weight 60 kg, haul-load 60 kg
- Operating time 13 hours, recharge time 3.5 hours, autonomous recharging on dock
- Zero degree turning radius thanks to differential steering
- Maximum speed of 4mph
- Programmable voice and audio prompts
- Optional joystick for shenanigans



Figure 1. Adept Lynx/11/

1.3.3 Adept Lynx Software

To realize the envisaged functions, the Adept Lynx software is applied and illustrated in this thesis.

1.3.3.1 MobilePlanner

In order to set up AIV autonomous mobile activities, the user must map the operating space and configure its parameters. The MobilePlanner software is used to render this map and carry out configuration (Figure 2). The license key is required to open it.

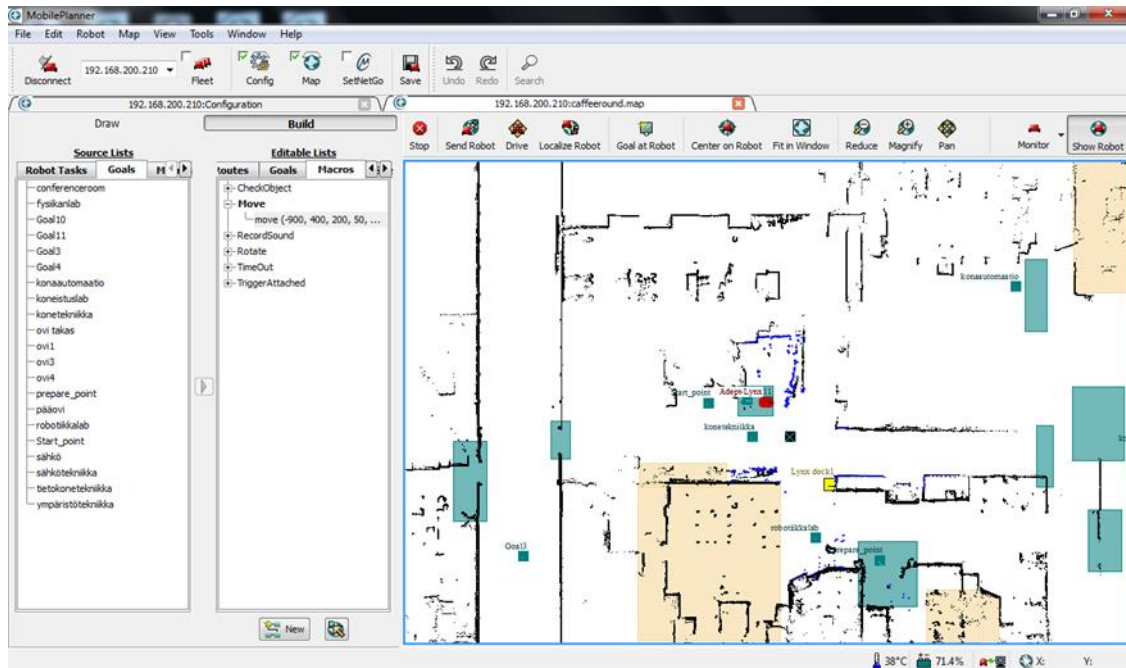


Figure 2. MobilePlanner

1.3.3.2 MobileEyes

The MobileEyes software can monitor multiple AIV's activities, executing them in the mapped space.

1.3.4 Adept Lynx command language

ARCL (Advanced Robotics Command Language) is a simple, text-based, command-and-response operating system designed to integrate a fleet of Adept mobile robots with an external automation system.

Among its functions, users can operate and monitor the mobile robot, its accessories and payload over the network. In other words, ARCL serves to automate the specified mobile robot. Meanwhile, the Telnet or putty provides access to the ARCL commands from a command prompt which supports user debugging.

1.4 Introduction to motor controller CMMO-ST

1.4.1 Motor controller

A motor controller is one or more devices which govern electric motor performance in some manner. Typically, a motor controller includes a manual or automatic means for

starting and stopping the motor, selecting rotation direction or speed, regulating torque, and safeguarding against overloads and faults.

1.4.2 CMMO-ST specifications

In this project, the motor controller, CMMO-ST, receives commands from a user to perform functions of its electric cylinder, which is its axis motor unit. /9/

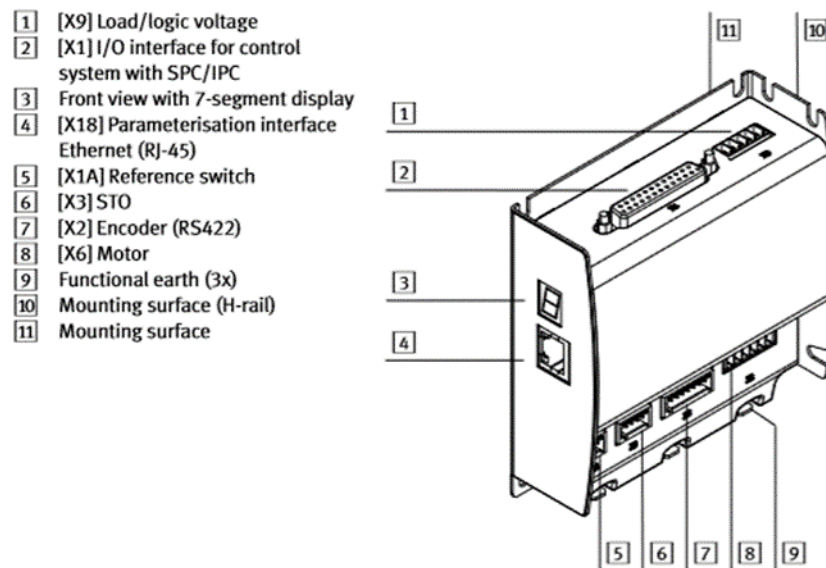


Figure 3. CMMO-ST/14/ product overview

1.4.3 Software for configuration and commissioning

1.4.3.1 Web service

Certain functions can be performed through the motor controller's integrated web server

- 1) The CMMO-ST's status can be determined through the web server.
- 2) Parameterization and commission of positioning systems for the optimized motion series can be simplified.
- 3) FCT parameter file transmission is enabled so that configuration data can be uploaded to a computer or downloaded to the motor controller.

1.4.3.2 FCT (Festo configuration tool)

The Festo Configuration Tool (FCT) is Windows-based software for the parameterization, commissioning and diagnostics of drives with configurable motor-axis combinations and positioning systems (OMS).

In FCT commissioning, configuration and parameterization are carried out through a page-oriented workflow. FCT enables the following:

- a) Configuration of the entire Festo modular system of axes and motors;
- b) Configuration of user-specific axes
- c) Mechanicals
- d) Use of the motor controller's maximum function range;
- e) Extended status displays, diagnostics options and test functions;

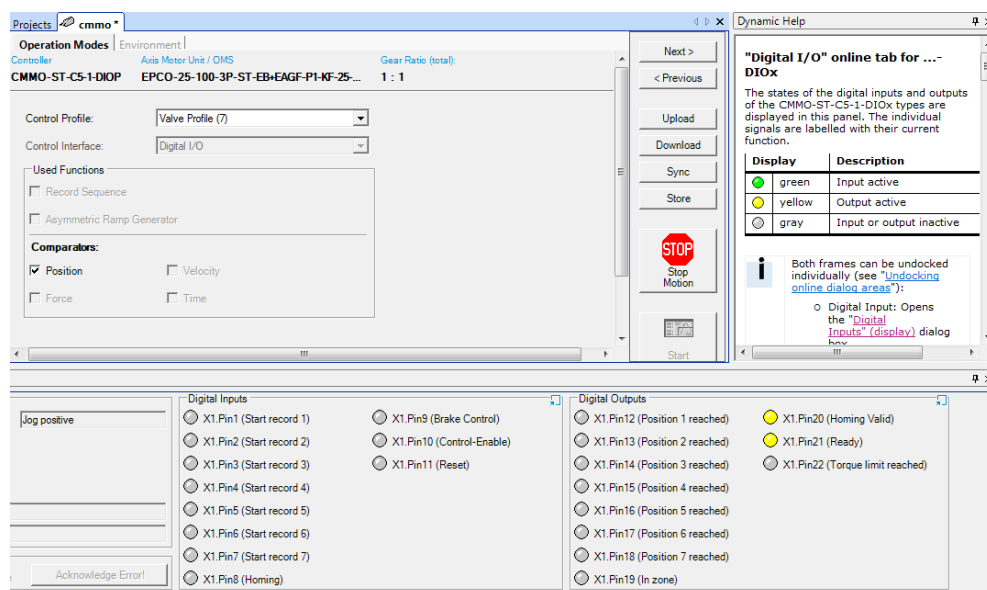


Figure 4. FCT panels

1.5 Introduction to the Odoo system

1.5.1 ERP system

As an integrated management system for core business processes, Enterprise resource planning (ERP) typically takes place in real-time and is mediated by both software and

technology. In some cases, it is considered a form of business-management software, capable of collecting, storing, managing and interpreting data from various business activities. With regards to its implementation, the common database maintained by a database management system supplies ERP with an integrated and continuously updated view of core business processes.

ERP contributes in three aspects. Firstly, the business resources related to the core process can be tracked, including cash, raw materials, production capacity and business commitment status (orders, purchases, and payroll). Secondly, the data from various departments are shared among the system's multiple applications. Furthermore, ERP optimizes the information flow of business functions, constructing a common framework with outside stakeholders.



Figure 5. ERP modules/4/

1.5.2 Odoo key features

Odoo is an all-in-one management software, meaning it offers multiple business applications which together form a complete suite suitable for companies of all sizes.

As a piece of intelligent business software, the Odoo modules cover almost all user requirements including CRM, digital activities, billing, accounting, manufacturing, warehousing (including project management), and inventory.

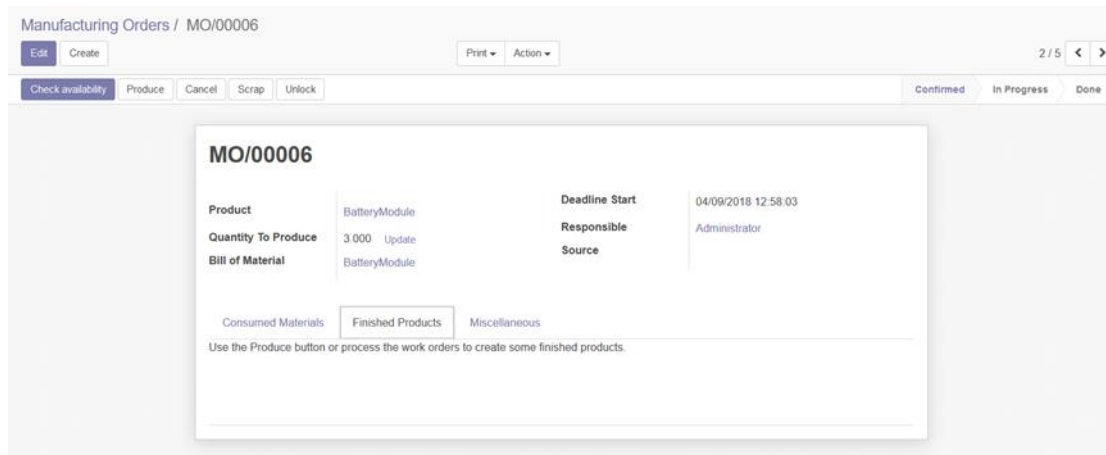


Figure 6. Odoo overview

1.5.3 Odoo programming

Although Odoo usually extends internally via modules, many of its features and all of its data are also available for external analysis or integration with a range of tools. Therefore, Odoo, otherwise known as OpenERP, is based on client/server architecture, with the client and server communicating using XML-RPC protocol.

XML-RPC is a web technology consisting of a set of tools for constructing distributed applications on top of existing web infrastructures. The web is consistently applied as a "transport layer" which does not guarantee a direct human interface via the browser.

Extensible Markup Language (XML) provides a vocabulary for describing Remote Procedure Calls (RPC), which is then transmitted between computers using HyperText Transfer Protocol (HTTP). RPC provides a mechanism for developers to define interfaces that can be called over a network. These interfaces can be simple or complex. XML-RPC calls are conducted between two parties: the client (the calling process) and the server (the called process). A server is made available at a given URL.

XML-RPC therefore allows multiple computers with different operating systems written in different languages to share processing. The RPC approach spares programmers the trouble of learnings underlying protocols, networking, and various implementation details.

XML-RPC can be used with Python, Java, Perl, PHP, C, C++, Ruby and Microsoft's .NET. Implementations are widely available for platforms such as Unix, Linux, Windows and Macintosh. Thus, part of Odoo's Model Reference API is easily available over XML-RPC and accessible in multiple languages.

The diagram below synthesizes OpenERP's client server architecture. The OpenERP server and the OpenERP clients communicate using XML-RPC.

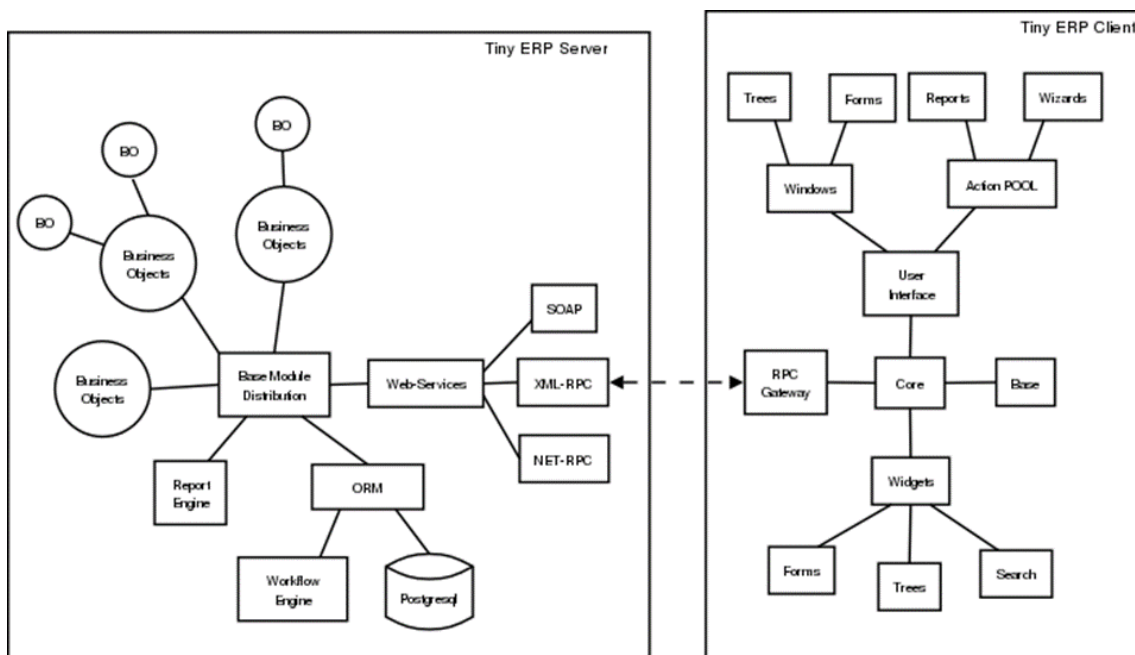


Figure 7. OpenERP/13/ architecture

Because Odoo (OpenERP) logic is already configured for the server side, the client side's specification and functions are very simple, with responsibility limited to posting data such as forms, lists and trees and sending results back to the server. Thus, it is easy for users to construct a client and communicate with the existed server.

1.6 ABB robot introduction

1.6.1 ABB's IRB 1200

Nowadays, industrial robots, those used for manufacturing, are emerging across more and more sectors. These robots are automated, programmable and capable of movement on at least two axes, with most able to accomplish various industrial processes including welding, painting, assembly, picking and placing printed circuit boards, packaging and

labeling, palletizing, product inspection, and testing. Thanks to their high endurance, speed, and precision, these robots can assist humans in handling materials.

Taking this thesis's topic into account, the robot belonging to the ABB 1200 family can assemble batteries in a given time. ABB's IRB 1200 is a perfect industrial robot which saves the user space and time thanks to its small size and operating efficiency. More specifically, it addresses the requirements that material handling and machine tending applications have in relation to flexibility, ease of use, compactness and short cycle times without compromising large working envelopes.

1.6.2 ABB robot software

Offline programming is the best means for maximizing the return on investment for robot systems. The problem lies in converting the robot controller to a PC application.

RobotStudio, which is an exact copy of the real software package that runs robots in production, is built on the ABB Virtual Controller and has solved this problem. This application allows realistic simulations to be performed, using real robot programs and configuration files identical to those used in practice. With this, users can program on their PC in the office without halting production.

Considering its functions, RobotStudio provides the tools for increasing user robot system profitability by letting tasks such as training, programming, and optimization be performed without disturbing production.

In all, Robot Studio's advantages include:

1. Lower risk
2. Quicker start-up
3. Shorter change-over
4. Increased productivity.

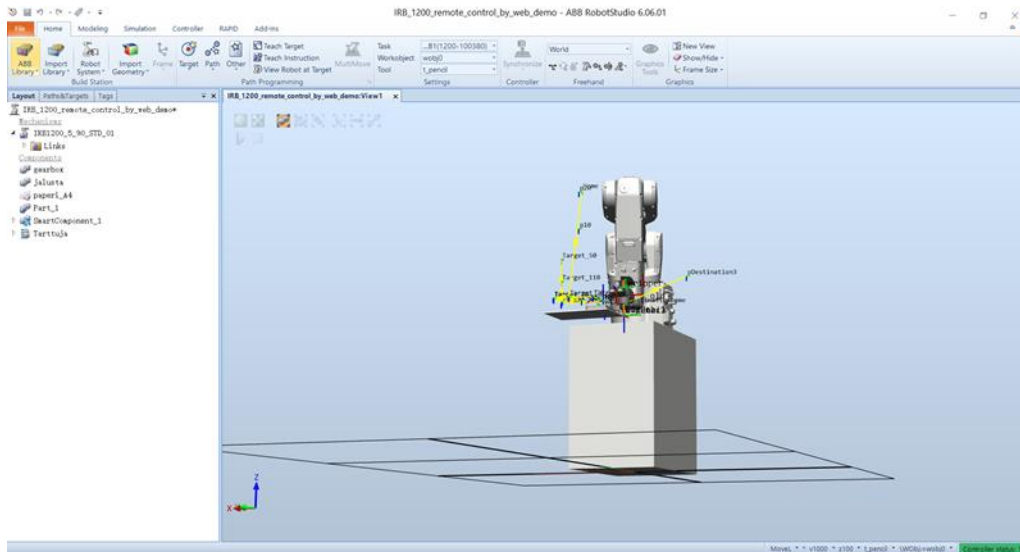


Figure 8. RobotStudio panels

1.6.3 ABB Robot web service

For robot task execution or data updating, the Robot Web Services, which are designed based on the "architectural style "REST, can support users with their own programs.

Representational State Transfer (REST), is an architectural style that defines a set of constraints and properties based on HTTP. Web Services that conform to the REST architectural style, or RESTful web services, provide interoperability between digital-enabled computer systems. REST-compliant web services provide requesting systems with access to and manipulation of textual representations of web resources through a uniform and predefined set of stateless operations.

The application protocol in Robot Web Services is HTTP, while the cornerstones of HTTP are URL's and Verbs. An URL is designed to identify something, such as a resource. A HTTP Verb defines the method to be executed on a resource. Each resource can support one or more HTTP verb. HTTP has multiple predefined verbs, the most important of which are:

- GET: Retrieve a resource
- PUT: Create or update a resource
- POST: Update a resource

- DELETE: Delete a resource

GET does not change a resource's status, whereas DELETE, PUT and POST do.

Robot Web Services highlight a set of web APIs that can any HTTP aware client can consume using any programming language. The APIs return data either as XML or JSON, which can then be parsed using standard XML/JSON parsers. The available Robot Web Service can be easily verified using a Web browser and typing the robot controller URL in the address field. It is possible to use the default user name and password, "Default User" and "robotics", so as to read most data. Meanwhile, in order to use the REST APIs, the client application should execute a HTTP request and parse the response.

Clients should not monitor for state changes in resources, because these are sent as events. Robot Web Services support the Websockets protocol, through which clients can subscribe to changes.



Figure 9. Robot web service/8/Architecture

2 OVERALL STRUCTURE

This chapter illustrates this project's architecture, describing each section before providing a flow chart which outlines the entire process.

2.1 Project structure

Taking the envisaged project functions into account, the project is divided into five components: the ABB robot module, Omron robot module, CMMO controller module, strategy module and Odoo module. These components cooperate in order to execute tasks within the limited time.

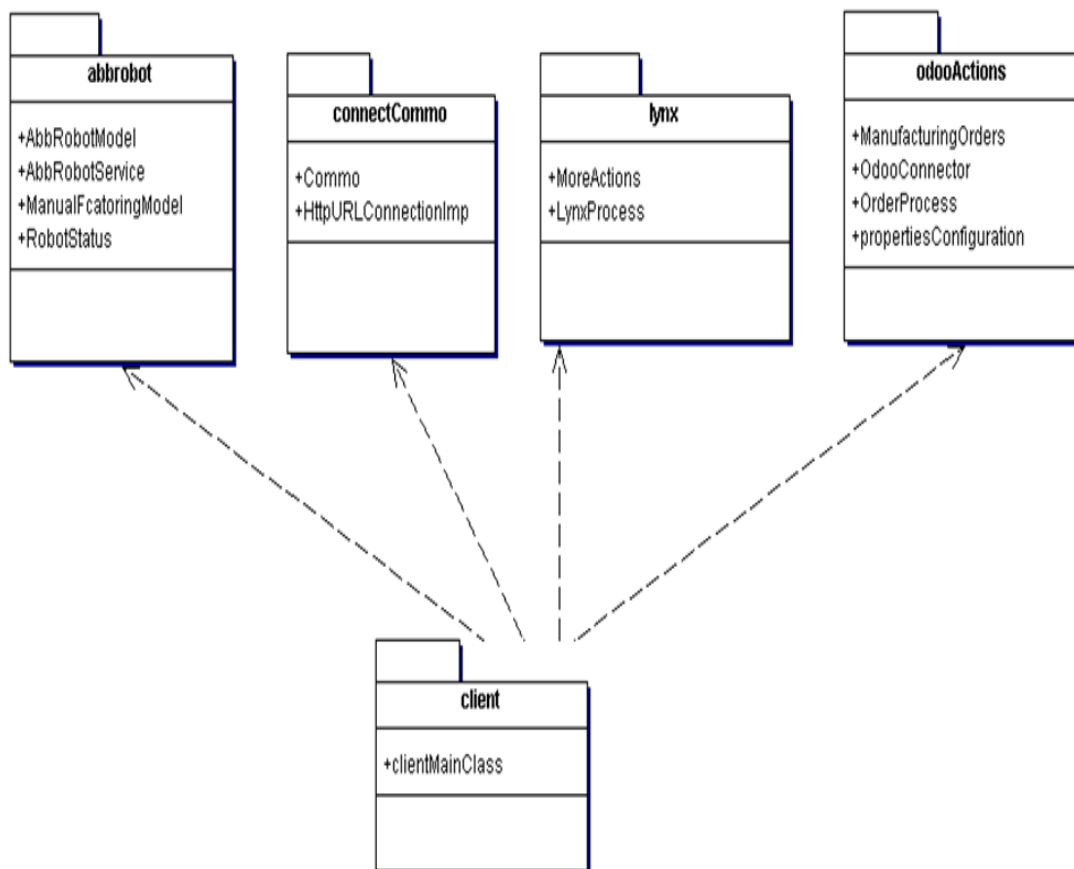


Figure 10. Whole project architecture

2.2 Module description

- ABB robot module

This module connects with the ABB robot, changing its parameters as the data which is received from Odoo. In order to realize this goal, the robot web service, which uses HTTP protocol and is based on the REST structure, is utilized with the file including the ABB robot control programs available to the robot controller—RobotStudio.

The application constantly polls so as to monitor which programs the robot has finished by checking whether the specified parameter has changed or not. Once a positive response is obtained, the parameter "readAmount" is changed and then recognized by this application so that the user is aware of the module status and can execute the rest of the work flow.

- Omron robot module

The Omron robot module connects with the Omron Adept Lynx robot, instructing it to execute a sequence of actions such as going to a specified area and realizing a certain position. In this part, related applications such as MobilePlanner and MobileEyes can assist the user in preparation work, while ACRL language is mastered and applied in the program. If the user starts this application, the telnet connection is created with Adept Lynx and the code, include correct ACRL command, is sent to the robot according to the map sequence.

- CMMO controller module

In the CMMO controller module, multiple connection methods are present, with the differences between them determining module performance effectiveness and convenience. After identifying the advantages and disadvantages of connection methods, the web service is selected so as to communicate with the CMMO controller, sending it to execute actions of the electronic cylinders. The CMMO controller data are saved in the parameter file, with configuration information fixed according to the parameter file when the application is running.

- Odoo module

This component is responsible for reading, searching and updating data in Odoo. In its process, the first confirmation made is Odoo having been installed and the database created following certain requirements. Next, because the target section is the

manufacturing module, the key words must be saved in the property file, so that when the XML-RPC methods are called, they can be realized without the parameters' names.

- Strategy module

This component serves as the main class for organizing other modules' functions. Furthermore, its innovative points, as future features, are the design of the complex event detection method based on the priority structure in the manufacturing schedule and the shortest path algorithm applied in route selection. Hence, the module produces the best solution for executing module performance.

2.3 Project flowchart

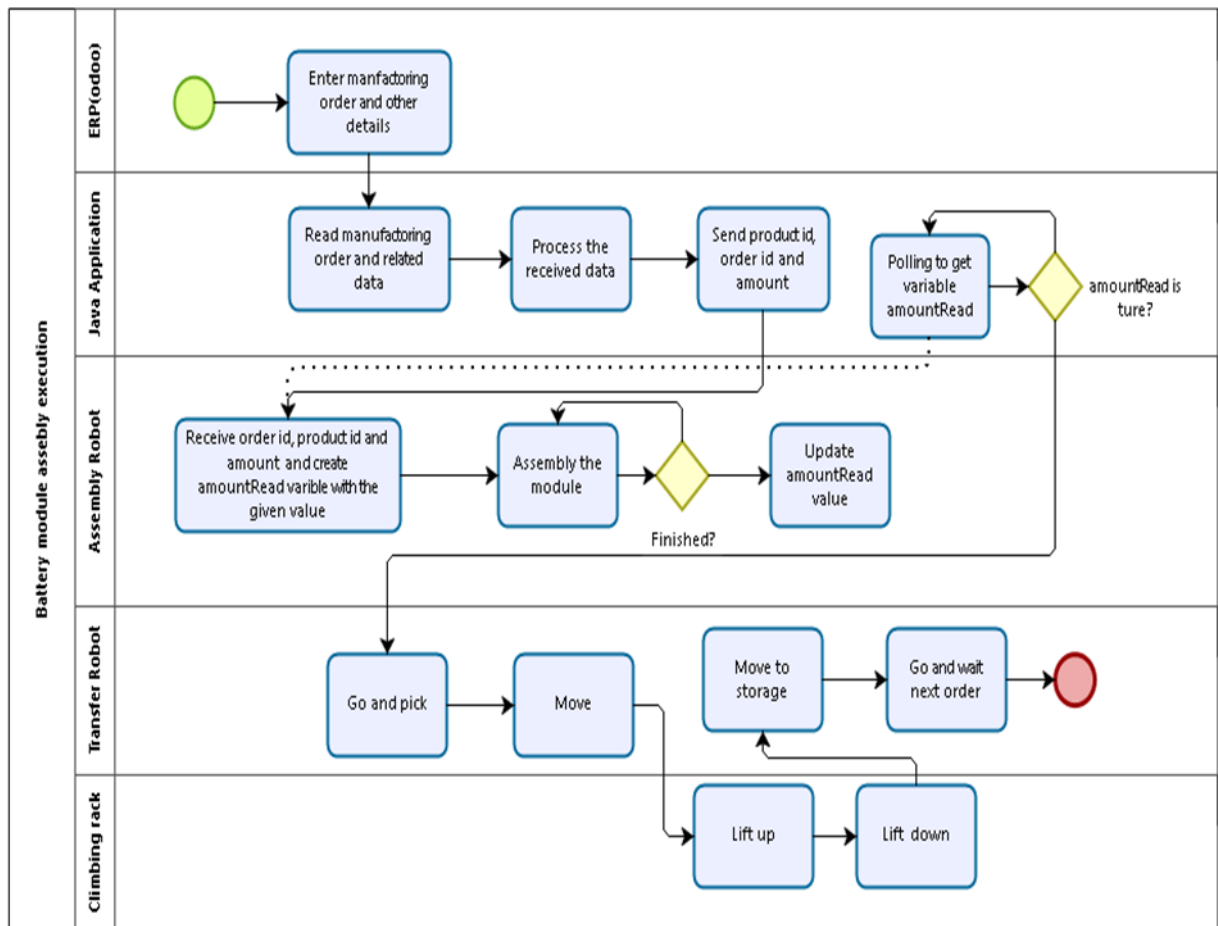


Figure 11. Whole project flowchart

In this flowchart, the Odoo module first gets the specified data according to user needs, then processes the manufacturing orders taking different conditions into consideration. The next step is to send the relevant information in a single order to the ABB robot. Once

the ABB robot completes the battery assemblage process, the value of the parameter “amountRead” is changed. If the Java application detects this transform, the Omron robot is enabled and called to take the shortest route to pick up the necessary materials. After the Adept Lynx arrives at its target location, the CMMO controller moves the electric cylinders up and down so as to lift the packaged battery. Then, the Omron robot moves again, taking the battery and waiting for the next order. Finally, the finished order’s status is converted to “done” so that every user has the latest data.

3 ODOO MODULE

This section introduces the Odoo service configuration, steps for importing Odoo Java APIs and the primary Odoo Java API methods applied in this project.

The initially prepared environment dictates how the project starts. Thus, the Odoo system configuration is fixed before programming, or more precisely, it is installed and run automatically. Using the task manager, the user can monitor current service status so as to ensure that the Odoo system is operating normally. Furthermore, the database and various modules designed by the individuals should be added. In order to make the system more closely resemble reality, information regarding products and manufacturing orders were collected for the product and manufacturing modules.

Through these efforts, the following actions are rendered more effective.

In the Odoo system with XML-RPC architecture and the ability to accept multiple languages and platforms, the developers can use Python, Ruby, PHP or Java on machines running Unix, Linux, Windows or Macintosh when programming web services using XML-RPC. Considering the further component development, Java is selected. Hence, the XML-RPC web service must first be mastered because it serves as the cornerstone of later development. Knowledge pertaining to Odoo Java API about concepts helps users adapt to library implementation instances.

Possessing a comprehensive command of Odoo Java API is insufficient for this project. Unlike other languages, the structure of Java applications is unique in its design. Thus, familiarity with the examples in the Java API guide is essential for those interested.

3.1 Environment configuration

3.1.1 Installation

Odoo can be installed through multiple methods, but not all are appropriate in every situation. Rather, the installation methods depend on the intended use. The following content describes each method and their key features.

1. Online: The easiest way to use Odoo in production or to trial it.

2. Packaged installers: Suitable for testing Odoo, developing modules and for long-term production subject to additional deployment and maintenance work.
3. Source Install: Provides greater flexibility such as running multiple Odoo versions on the same system. Good for developing modules and can be used as the basis for production deployment.
4. Docker: If the user typically uses the docker for development, an official docker base image is available.

In this project, the packaged installer meets the requirements. Hence, it is selected because its features support development in other aspects.

Odoo provides packaged installers for Windows, deb-based distributions (Debian, Ubuntu,) and RPM-based distributions (Fedora, CentOS, RHEL, ...) for both the Community and Enterprise versions. Thus, the packaged installer for Windows is easy to use and apply.

Installation steps:

- Download available version/5/: The Community version package was downloaded.
- Check the components in relation to user needs during installation. Because the user must save their own data in the Odoo system, the SQL Database must be installed.

/6/

3.1.2 Component Extension

The Odoo system allows users to design its functions and features according to their requirements. Therefore, real-world conditions should be researched, recorded and logged into the Odoo system.

Component Extension steps:

- Add the new database: The specified database should be created and basic information, such as database name, email and password, preserved because data can be applied during user programming.

Create Database
×

Odoo is up and running!
Fill out this form to create a new database. You will install your first app afterwards.

Database Name

Email

Password

Language **Country**

Load demonstration data (Check this box to evaluate Odoo)

Continue

Figure 12. Adding the new database

- Adding the specified module: Odoo supplies the user with various applications which can satisfy their individual needs (Figure 13). For this project, the manufacturing module is required.

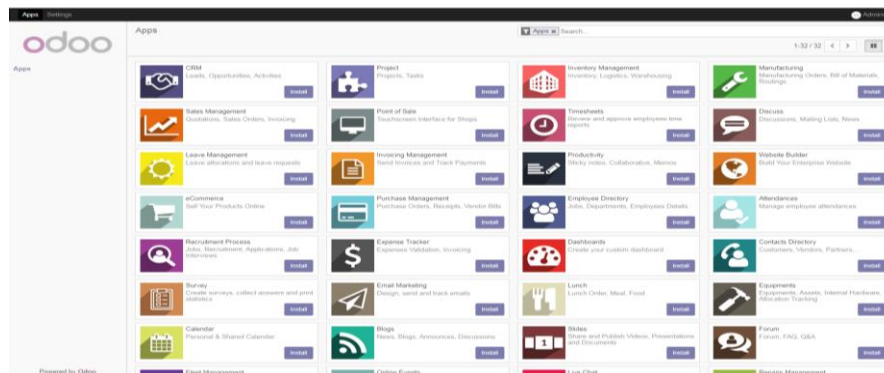


Figure 13. Adding the specified module

- Adding product materials: Given that the battery module is assembled from multiple elements such as battery cells, the battery connector, the battery frame, entries for these products are created and updated with amount, price and so on.

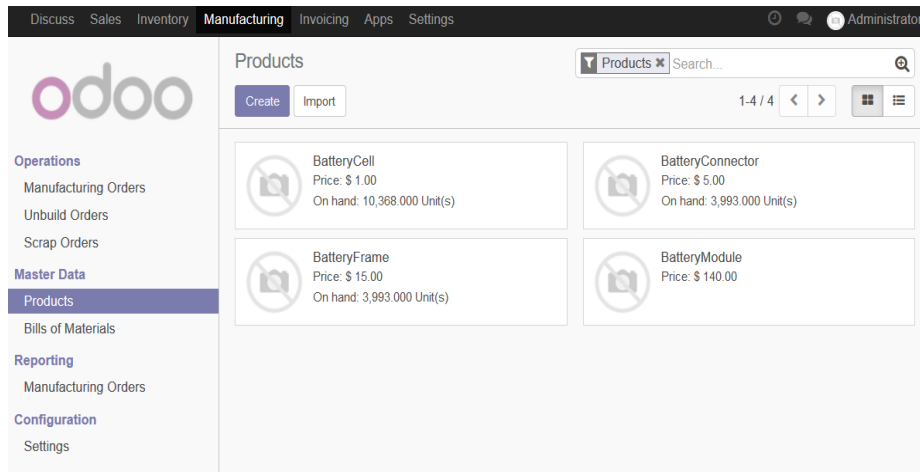


Figure 14. Adding product materials

- Adding the manufacturing orders. After the materials are prepared, manufacturing orders can be created one by one. The reference, material availability and order status are produced automatically by Odoo, while the deadline, product choice and quantity are edited by the administrator.

Reference	Deadline Start	Product	Quantity	Materials Availability	State
<input type="checkbox"/> MO/0002	04/28/2018 18:23:06	BatteryModule	3.000	Available	Confirmed
<input type="checkbox"/> MO/0004	04/27/2018 20:33:57	BatteryModule	4.000	Available	Done
<input type="checkbox"/> MO/0008	04/19/2018 13:52:22	BatteryModule	10.000	Available	Done
<input type="checkbox"/> MO/0007	04/15/2018 13:37:47	BatteryModule	10.000	Available	Done
<input type="checkbox"/> MO/0003	04/04/2018 16:30:21	BatteryModule	1.000	Waiting	Done
<input type="checkbox"/> MO/0001	03/23/2018 10:32:09	BatteryModule	8.000	Available	Done
			36.000		

Figure 15. Adding the manufacturing orders

3.2 Odoo Programming

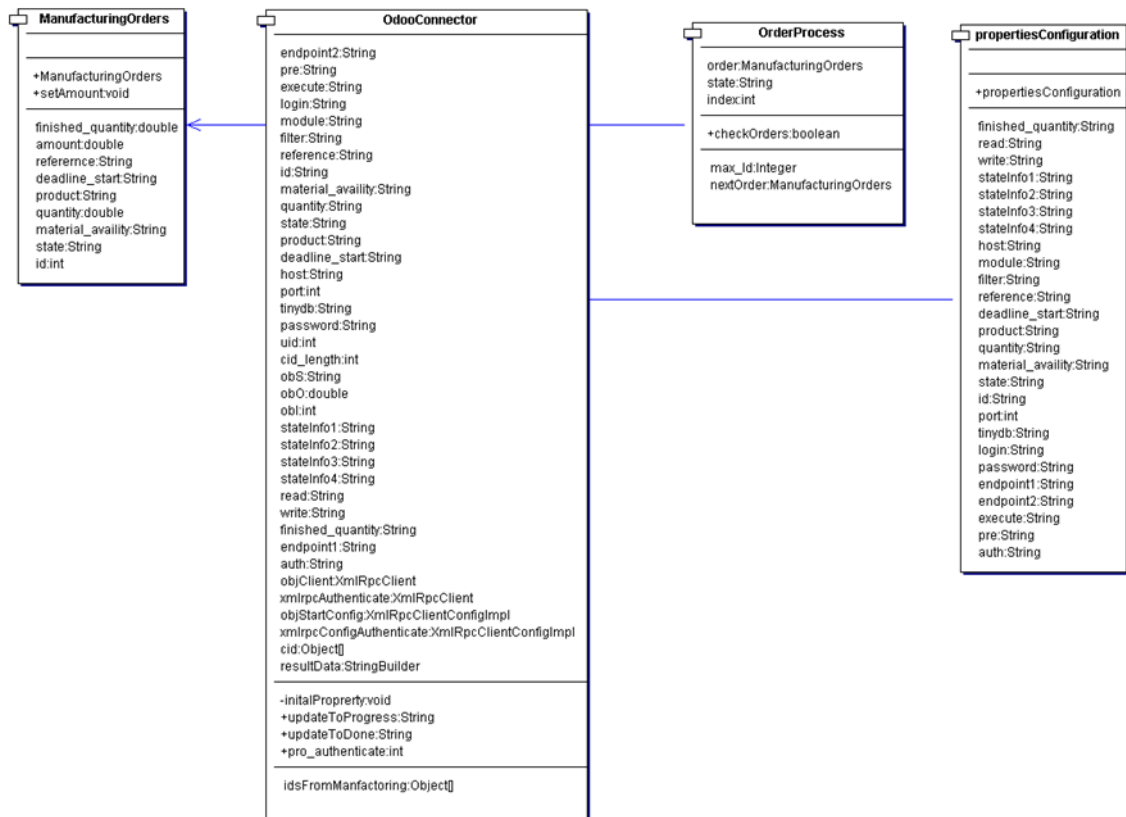


Figure 16. Odoo module architecture

During programming, code segments are divided into various classes according to their functions so as to elucidate the underlying logic. The class “PropertiesConfiguration” collects all the key words in Odoo. “ManufacturingOrder” initializes the order produced by Odoo. “OdooConnector” calls the Odoo java API to read, search and update manufacturing order data. “OrderProcess” verifies whether orders are available and calls the next order according to the latest deadline.

3.2.1 Odoo Java API

To achieve the “OdooConnector” class functions, the Odoo Java API available through XML-RPC should be applied. In order to master this technology, the user must grasp several points which are outlined below.

- Configuration

In the Odoo Java API configuration, certain elements must be logged. Firstly, the server URL, which is the instance's domain (e.g. <https://mycompany.odoo.com>). Secondly, the database name, which is the name of the instance (e.g. mycompany). Finally, the username, which is configured as the user's login shown by the Change Password screen.

```
final String url = <insert server URL>,
    db = <insert database name>,
    username = "admin",
    password = <insert password for your admin user (default: admin)>;
```

Figure 17. Configuration code

To make this part available, the configuration content was organized in the property file.

```
#url
pre=http
host=localhost
port=8069
#db
database=BatteryModule
#username
dbuser=867251552@qq.com
#password
dbpassword=ugUshe4r
```

Figure 18. Configuration in the property file

- Logging in

Odoo requires API users to be authenticated prior to data querying.

The “xmlrpc/2/common” endpoint provides meta-calls which don't require authentication, including version information retrieval. The authentication itself is carried out through the authenticate function, which returns a user identifier (uid) in the authenticated calls instead of the login. Moreover, the user identifier is applied for calling other methods.

Prior to authentication and identifier retrieval, the key words endpoint and authentication should be written in the property file.

```
endpoint1=/xmlrpc/2/common
auth=authenticate
```

Then, in the certain method, the project's related content was organized.


```
xmlrpcConfigAuthenticate.setServerURL(new URL(pre, host, port, endpoint1));
xmlrpcAuthenticate.setConfig(xmlrpcConfigAuthenticate);

int uid = (int)xmlrpcAuthenticate.execute(
    xmlrpcConfigAuthenticate, auth, Arrays.asList(
        tinydb, login, password, Collections.emptyMap()));
```

Figure 19. Authentication in the property file and code

- Calling methods

In the previous points, the configuration and log-in were introduced. However, when the user wants Odoo to execute more complex functions, then calling methods are first utilized. This is done through the second endpoint, “xmlrpc/2/object”, which is applied to the Odoo model to call methods via the “execute_kw” RPC function.

Each call to “execute_kw” takes the following parameters:

- the database used, a string
- the user id (retrieved through authentication), an integer
- the user's password, a string
- the model name, a string
- the method name, a string
- an array/list of parameters paced by position
- a mapping/dict of parameters to pace by keyword (optional)

At first, the key word endpoint should be written in the property file. Then, two variables are created, XmlRpcClient and XmlRpcClientConfigImpl. Next, they are initialized and parameters set with the exited data.

```
endpoint2=xmlrpc/2/object
```

```

XmlRpcClient objClient;
XmlRpcClientConfigImpl objStartConfig;

objClient = new XmlRpcClient();
objStartConfig = new XmlRpcClientConfigImpl();
objStartConfig.setServerURL(new URL(pre, host, port, endpoint2));
objClient.setConfig(objStartConfig);

```

Figure 20. Preparation of the calling methods in the property file and its code

- List records

In examining every item in the specified module, the method search () can be used to produce a perfect solution, helping the user solve the issue. In other words, records in the certain field can be listed and filtered via search().

In regards to its protocol, search () takes a mandatory domain filter (possibly empty), and returns the database identifiers for all records with which there is a match.

To start, the key word related data should be written in the property file as the figure.

```

execute=execute_kw
module=mrp.production
filter=search

```

Figure 21. Parameters of the searching method in the property file

Then, all the manufacturing orders should be listed.

```

configList.add(tinydb);
configList.add(uid);
configList.add(password);
configList.add(module);
configList.add(filter);

paramList.add(condiState);
configList.add(paramList);
cid = (Object[]) objClient.execute(execute, configList);

```

Figure 22. Searching method code

- Read records

Although the ID array was saved after filtering, other order fields remained unknown. In order to obtain the information needed, the read () method is next applied.

Record data is accessible via the read() method, which takes a list of IDs (as returned by search()) and optionally a list of fields to retrieve. By default, all the fields the current user can read are retrieved, typically creating a large quantity of results.

To start, the key word related data is written in the property file.

```
read=read  
reference=name  
deadline_start=date_planned_start  
product=product_id  
quantity=product_qty  
material_availability=availability  
state=state  
id=id  
finished_quantity=qty_produced
```

Figure 23. Parameters of the reading method in the property file

Then, all the other manufacturing order fields are listed.

```

cid = this.getIdsFromManufactoring();

for (int i = 0; i < cid.length; i++) {
    argsList.add(cid[i]);
}

fieldList.add(reference);
fieldList.add(id);
fieldList.add(deadline_start);
fieldList.add(product);
fieldList.add(quantity);
fieldList.add(material_availability);
fieldList.add(state);
fieldList.add(finished_quantity);
fieldMap.put("fields", fieldList);

paramList.clear();
configList.clear();
paramList.add(argsList);
configList.add(tinydb);
configList.add(uid);
configList.add(password);
configList.add(module);
configList.add(read);
configList.add(paramList);
configList.add(fieldMap);

cid_length = cid.length;

for (int i = 0; i < cid_length; i++) {
    records.add((Map<Object, Object>) ((Object[]) objClient.
        execute(execute, configList))[i]);
}

```

Figure 24. Reading method code

- Update records

When the manufacturing order is finished, its status must be updated using the write () method. Records can be updated using write(), which takes a list of records and updates them by mapping the updated fields to the values similar to create().

Multiple records can be updated simultaneously, but the same values are obtained for the fields being set. Thus, multiple data actions should be handled carefully by the user.

To start, the key word related data is written in the property file.

```
write=write  
stateInfo1=confirmed  
stateInfo2=progress  
stateInfo3=done  
stateInfo4=cancel
```

Figure 25. Parameters of the writing method in the property file

Then, manufacturing order status for the specified IDs is changed.

```
configList.add(tinydb);  
configList.add(uid);  
configList.add(password);  
configList.add(module);  
configList.add(write);  
  
paramMap.put(state, stateInfo3);  
  
List<Object> id = new ArrayList<Object>();  
id.add(idTochange);  
paramList.add(id);  
paramList.add(paramMap);  
configList.add(paramList);  
  
objClient.execute(execute, configList);
```

Figure 26. Writing method code

4 ABB ROBOT MODULE

The main solution in the ABB robot module involves applying the robot web service in order to update the manufacturing order in the ABB robot's program, thus optimizing and serializing the manufacturing process. In other words, when a group of tasks are marked as increasing or decreasing in orders, the operators identify them in the shortest time.

This module is divided into two components. The first component involves deploying a demo for RobotStudio prior to programming. A suitable environment ensures the module program works well and produces the correct response.

The second component deals with mastering the robot web service and further developing a series of actions responsible for sending, receiving and verifying data between the program and the code in RobotStudio.

4.1 Environment configuration

4.1.1 Demo deployment

In RobotStudio, which is downloaded from the ABB website and installed on the user's computer, the exited demo is first opened.

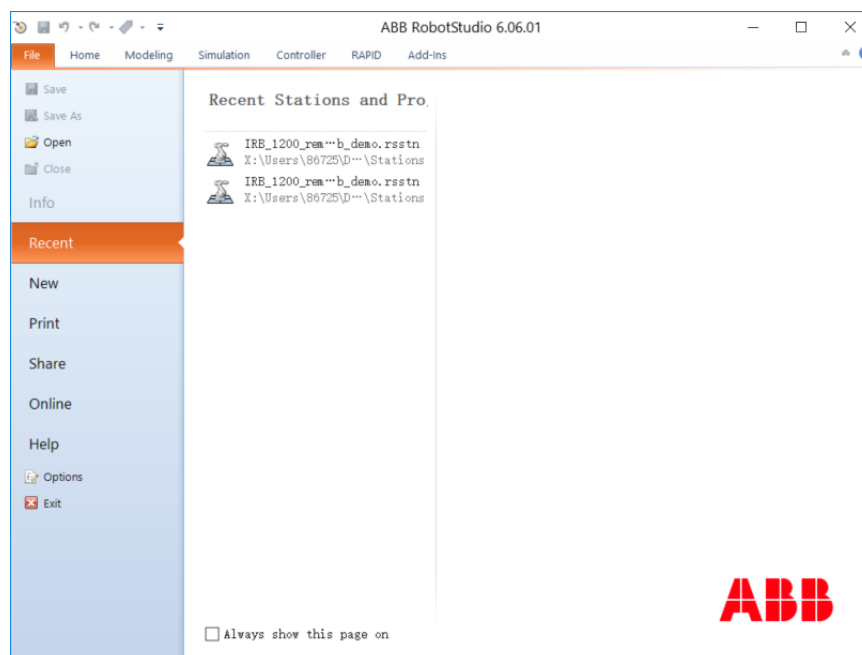


Figure 27. Recent file in RobotStudio

4.1.2 Parameter creation

Then, in the demo, the moving program “mMoverob” is located.

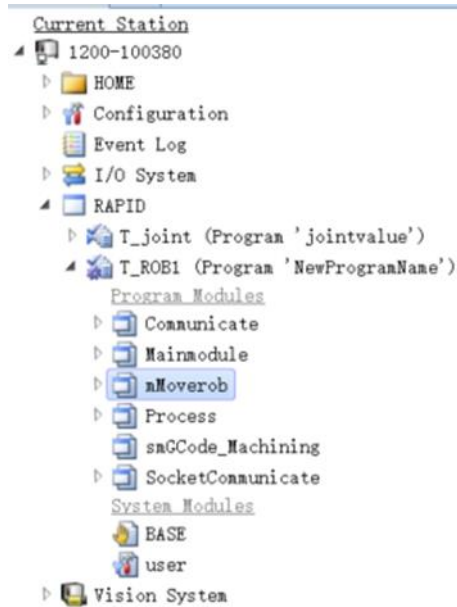


Figure 28. Current program structure

In this program, the required parameters are created and initialed at zero. The parameter “ordered” refers to the ID produced. The value of the parameter “productID” should be the same as the manufacturing order ID sent to the ABB robot. The parameter “quantity” represents the quantity of the prepared manufacturing order, which is calculated according to the supposed and finished product amounts. Finally, the parameter “amountRead” checks whether the ABB robot’s tasks have been finished.

```
PERS num orderId:=0;
PERS num productId:=0;
PERS num quatity:=0;
PERS num amountRead:=0;
```

After running this application, the parameter is changed as shown in Figure 29, in which every variable has been updated in accordance with the actual situation and the manufacturing orders received from Odoo.

```
PERS num orderId:=1;
PERS num productId:=8;
PERS num quatity:=9;
PERS num amountRead:=1;
```

Figure 29. Current program order information

4.2 ABB robot programming

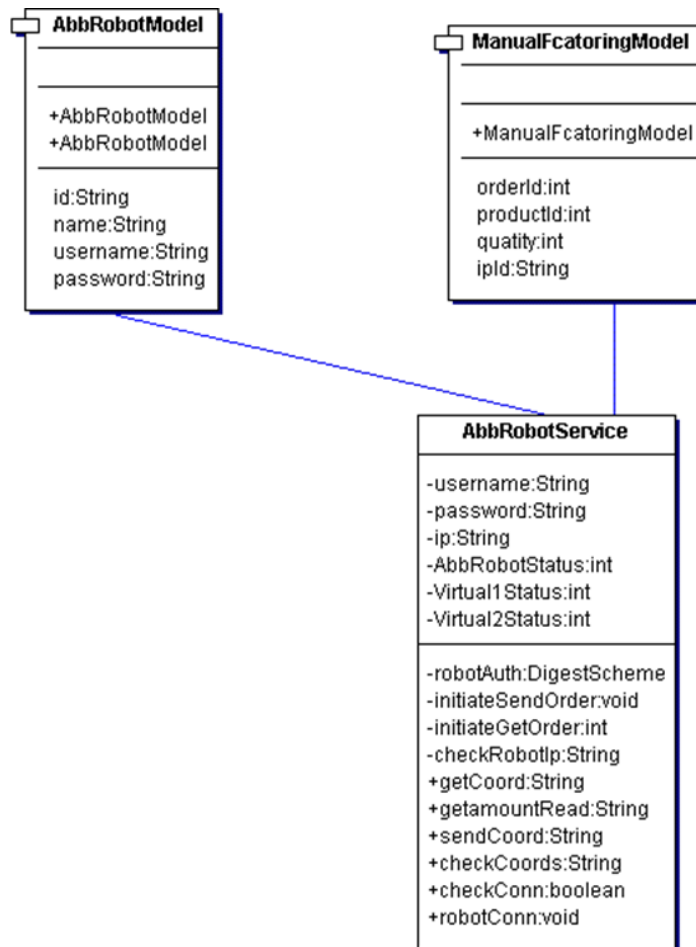


Figure 30. ABB robot module architecture

For this module’s programming, the four classes contribute towards achieving the common goal of updating the ABB robot’s parameters. The class “AbbRobotModel” collects the data related to the ABB robot, creating a model for it. “ManualFcatoringModel” plays a similar function as above, except that it creates a model available for manufacturing. “AbbRobotService” organizes the data processing methods and can send, receive data with the ABB robot in RobotStudio.

4.2.1 ABB robot web services

The class “AbbRobotService” serves as the core module component for the Robot Web Services, which require time and energy in order to be researched and mastered. Moreover, they consist of numerous services, each of which may have additional services or

resources, making the target resources difficult to identify. A subset of available services and resources is given:

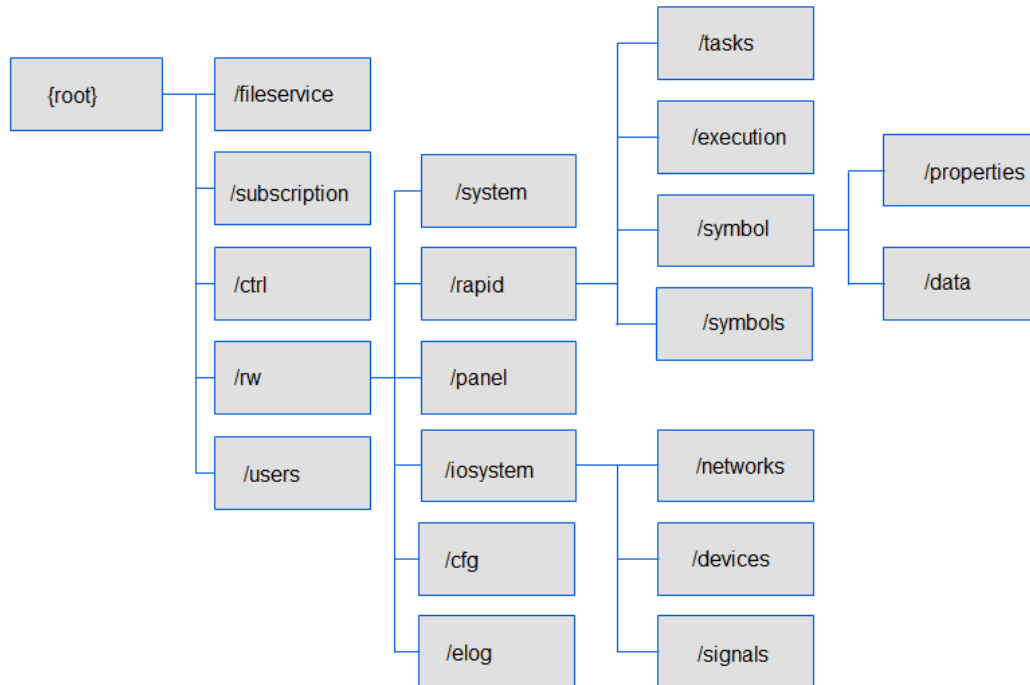


Figure 31. The subset of services and resources in robot web services

- Fileservice: Provides remote access to files and directories. Handles the transfer, creation, removal and renaming of files and directories (similar to FTP service.)
- Subscription service: Handles resource subscriptions and sends events when subscribed resources are updated. Subscription service uses "Websockets" for events.
- Ctrl service: Handles robot controller global functionality, such as access to the controller clock, controller identification and restart execution.
- Users service: Handles the registration of connected clients.
- RW Handles RobotWare services, such as IO, RAPID, E-log and CFG.

4.2.2 RobotWare services

As a component of Robot Web Services, the RAPID System Service resource map is shown in Figure 32. In this project, rapid symbol data and its updating are sought after, so that the rapid data URL can be determined according to:

“/rw/rapid/symbol/data/{symbolurl}”. Moreover, the “symbolurl” here refers to “RAPID/T_ROB1/” and the specified variable. Thus, the parameters in the prepared demo can be determined.

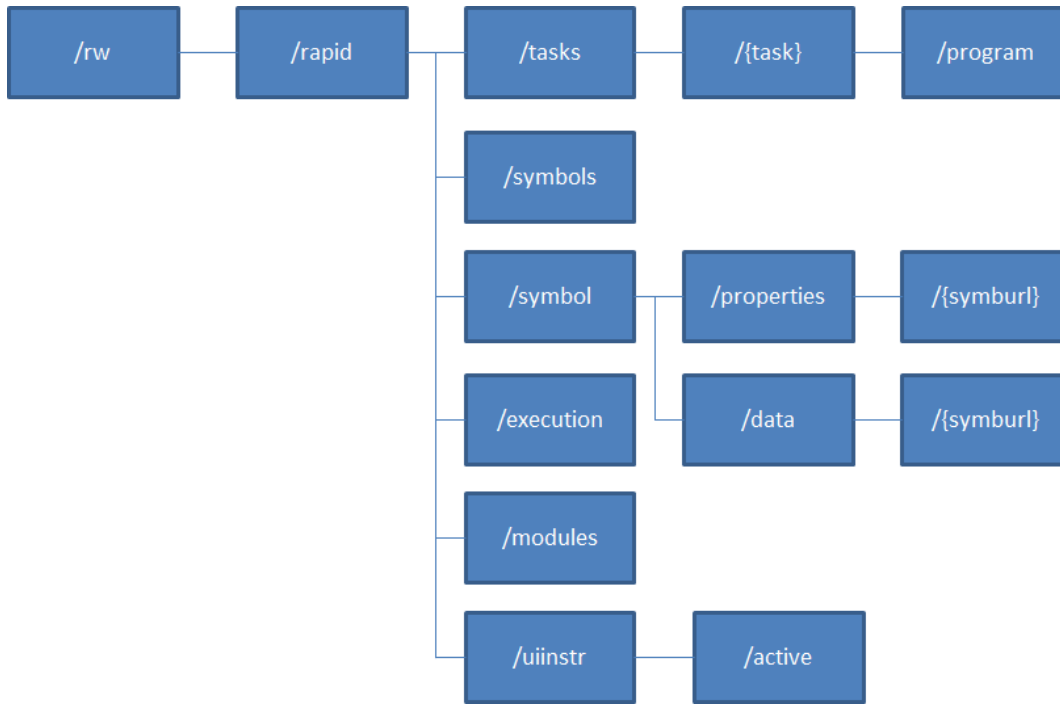


Figure 32. The subset of services and resources in RobotWare services

4.2.3 Operations on RAPID data

Two operations are necessary in rapid symbol data processing. The first operation involves receiving rapid symbol data. Because a resource is executed by the method defined by a HTTP verb in the HTTP protocol, and each resource supports one or more HTTP verb, the HTTP verb and its related method are assessed accordingly. During rapid symbol data retrieval, the GET verb serves as support.

Given below is an excerpt of the code for the first operation.

```

CloseableHttpClient client = HttpClients.createDefault();
//set the URL for the specified variable
URL url = new URL("http://"+checkRobotIp(ipId)+"/rw/rapid/symbol/data/RAPID/T_ROB1/"+manufacturingName);
HttpHost targetHost = new HttpHost(url.getHost(),url.getPort(),url.getProtocol());

HttpClientContext context = HttpClientContext.create();

CredentialsProvider credPro = new BasicCredentialsProvider();
credPro.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials(username,password));

AuthCache authCache = new BasicAuthCache();

authCache.put(targetHost, robotAuth());

context.setCredentialsProvider(credPro);
context.setAuthCache(authCache);
//use the GET verb
HttpGet httpGet = new HttpGet(url.getPath());
//execute and get the response
CloseableHttpResponse res = client.execute(targetHost, httpGet,context);

```

Figure 33. GET rapid symbol data

The second operation refers to updating a RAPID variable. In this process, the URL “action=set” is added to the original URL so as to carry out updating. Next, the HTTP verb is POST. Given below is an excerpt of the code for the second operation.

```

CloseableHttpClient client = HttpClients.createDefault();
//set the URL for the specified variable and suitable action parameter
String url = "http://"+checkRobotIp(ipId)+"/rw/rapid/symbol/data/RAPID/T_ROB1/mMoverob/"+manufacturingName+"?action=set";
//use the POST verb
HttpPost post = new HttpPost(url);

List<NameValuePair> Param = new ArrayList<NameValuePair>();
Param.add(new BasicNameValuePair("value", ""+value));
post.setEntity(new UrlEncodedFormEntity(Param));

HttpHost targetHost = new HttpHost(new URL(url).getHost(),new URL(url).getPort(),new URL(url).getProtocol());

HttpClientContext context = HttpClientContext.create();

CredentialsProvider credPro = new BasicCredentialsProvider();
credPro.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials(username,password));

AuthCache authCache = new BasicAuthCache();

authCache.put(targetHost, robotAuth());

context.setCredentialsProvider(credPro);
context.setAuthCache(authCache);
//execute and get the response
HttpResponse response = client.execute(targetHost,post,context);

```

Figure 34. Updating a RAPID variable

5 OMRON ROBOT MODULE

The Omron mobile robot, Adept Lynx, serves as the battery carrier. In order to make it move, the manual for the relevant software should be read and memorized. Moreover, the properties of ARCL (Advanced Robot Command Language) must be learnt and mastered.

This module is developed in regard to two aspects. Initially, the environment is scanned into the new map, with a list of target goals marked on the map in MobilePlanner. After obtaining the raw map, certain modifications should be carried out, thus rendering the map clearer and easier to follow. In order to make controlling the mobile robot more effective and the commands simpler, the concepts of macros and routes should be understood. A group of actions can be defined in one macro and several macros can form one route. Using these reduces the amount of command code.

The second aspect involves establishing the telnet connection and sending control orders in the program. In this section, the software putty can verify program correctness and test ARCL commands through its console. Then, Java helps to create the telnet client and apply the tested commands prior to communication with Adept Lynx.

5.1 Environment configuration

5.1.1 Initial map creation

In order to describe this module, the software MobilePlanner should first be introduced. This software package serves as the "control center" of the Adept Motivity software suite, providing the tools to accomplish all major mobile robot activities, such as creating and editing map files, creating and editing goals and routes, driving the robot and configuration.

With the Motivity software, including MobilePlanner, autonomous robots know where their position and drive from one place to another without a human operator. The only requirement is that the mobile robot has a prepared map of the operating environment static features. With MobilePlanner, making a map for the robot, and even an entire fleet of robots, is fast and easy.

For map creation purposes, this project's definition of a map should be determined because the map can only be planned using MobileEyes and MobilePlanner. The map in

these software packages represents the floorplan of the mobile robot's operating space, including the environment's static features, such as walls, doors and permanent shelving. Moreover, the map contains the route information, goals, docks, and tasks for the robot. The name of the map always has an extension.

As previously discussed, the Adept mobile robot is familiar with its environment, that is it has a map for navigation. In order to create a map, the user must first press scan in MobilePlanner and scan the workspace. Then, the environment can be scanned while the mobile robot navigates. At the same time, certain points, such as the charging station-dock, can be marked with the joy-stick, facilitating accurate and precise goal indication.

Although the scan file comes from the robot, the user map is saved on the local PC until it is saved elsewhere. Depending on file size, the conversion process can take a considerable amount of time and memory. Once the scan is complete, the user can transform it into a map so as to assign the mobile robot tasks to execute.

5.1.2 Editing maps by adding objects

This final section discusses the process of converting a scanned file into a map file so that the robot can navigate its environment. However, the map is not yet completed. Editing maps by adding objects such as goals, docks and forbidden areas is necessary.

At first, the Eraser tool can be used to make the map appropriate in more situations by providing a clear outline. Its main task is to remove map features which are temporary or moveable, such as chairs and forklifts. As for the objects in different ranges, eraser size can be adjusted between 5 to 1000mm in order to accept them.

Once basic map features satisfy the user, drawing tools can be applied so as to add goals, docks, forbidden lines and areas, as well as more advanced lines and areas. These features must be added so that the mobile robot can successfully navigate the workspace.

Thus, their definitions can be determined. Goals are virtual destinations to which the mobile robot drives in its environment. Goal properties include the x-coordinate, y-coordinate and the heading degree, which plays an important role in robot rotation. Figure 35 displays a list of goals marked on the map.

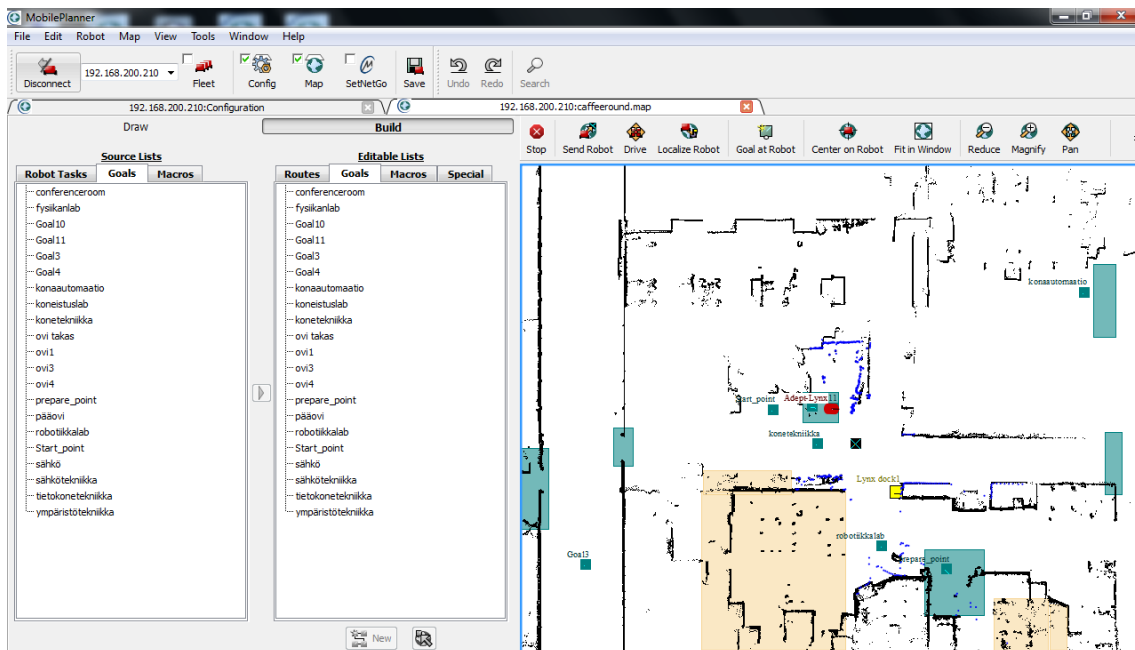


Figure 35. Marking the map goals

Advanced lines, points, and areas include features such as closed doors, measuring sticks, switchable forbidden lines or areas and one-ways. These advanced features can also be added to the map, controlling traffic flow in certain conditions.

Forbidden lines and areas are frequently applied in the maps. If the user wants to restrict mobile robot movement, adding forbidden lines and areas can help. These areas typically correspond to a portion of the operating environment which the mobile robot is not allowed to enter.

IgnoreSonar area, which refers to areas on the map where sonar sensors are disabled, should also be added. This can be useful for crossing known thresholds so as to avoid activating the foot sensor, which thus reduces the risk of mobile robot immobility in the presence of obstacles.

Figure 36 presents the menu for draw tools which can add advanced areas and lines.

sonar sensors would detect obstacles and halt the robot. Thus, the robot has to move backwards, with its distance value given as negative in Figure 37.

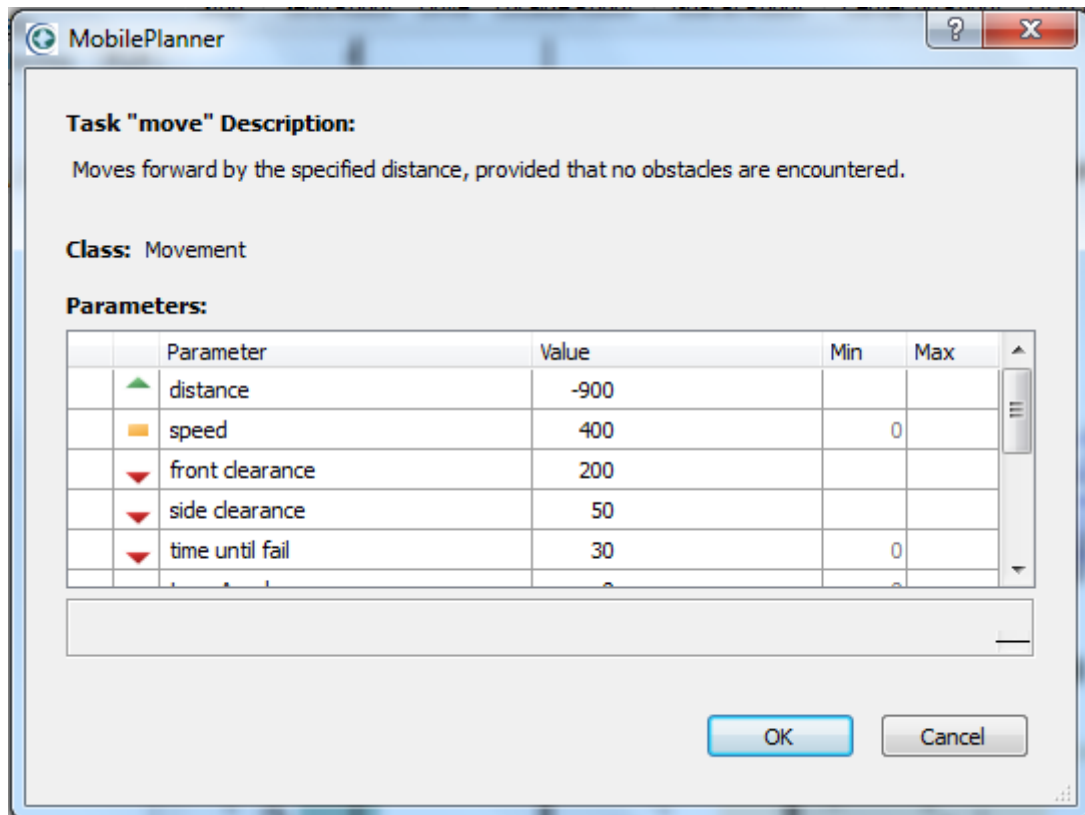


Figure 37. Move task details

The setHeading task is another necessary task which belongs to the category of non-instant tasks. This task supports the user in turning the robot towards a specified heading. This parameter represents the number of degrees the robot must turn and is especially significant when the robot comes to the door and wants to go backwards. Robot rotation can be altered using this method.

Speech and sound tasks are those that control robot audio. As the mobile robot navigates the operating space, a sound can be played during driving so as alert those around it of its presence or its next intended task. In this project, the speech task is programmed to initiate when the mobile robot begins patrolling the route and attracts user attention.

Because of the complex actions, compromise is necessary for task groups. In order to solve this problem, macros can be applied so as to combine goals and tasks together. Macros allow the user to save combinations of goals/tasks within one or more routes,

meaning macros are "reusable". For example, rotation and movement can be incorporated into a macro for efficiency.

The sum of all goals, tasks and macros should ultimately led to the route(s) being applied. In fact, a route is essentially a "to do" list for the mobile robot, consisting of a series of ordered tasks, goals, and macros for completion. In this project, two routes are created: PickRoute and LeaveRoute. The PickRoute collects all data related to going to the operating space. Firstly, the robot is called to go to prepare_point goal, which it acknowledges by saying "I am going to pick up a battery". This goal is located in the opposite direction of the door. Thus, the robot executes the macro to move backwards and wait. After picking up the battery, the LeaveRoute is called, initializing the macro for rotation and moving backwards as accompanied by the robot saying, "I am going to leave". Then the robot goes to another goal.

Finally, the finished map is saved to the robot so that it has all the information necessary to execute its tasks.

5.2 Omron robot programming

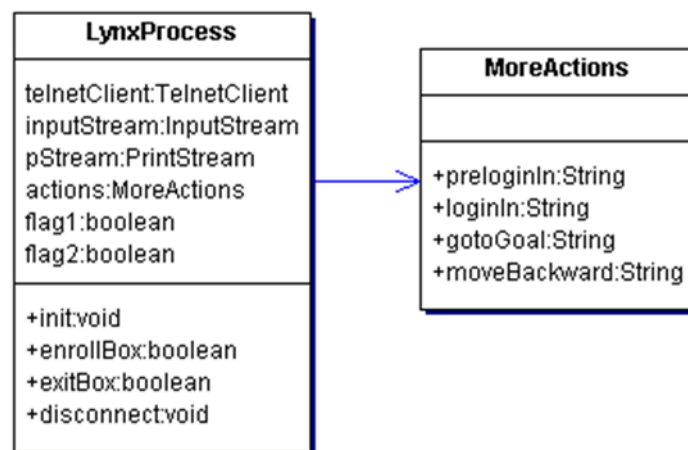


Figure 38. Omron robot module architecture

Figure 28 shows the two classes which contribute to module program. The LynxProcess class is used to build the telnet connection with the mobile robot-Adept Lynx and call the methods in the MoreActions class for controlling the mobile robot in performing certain functions.

The MoreActions class contains the methods for writing ARCL commands in the telnet console. These methods include actions such as logging in, reading instructions, patrolling the PickRoute and patrolling the LeaveRoute. All these actions help the robot finish the movements required for this project.

5.2.1 Working with Putty

The software introduced in the previous chapter provides the finished map with defined tasks and routes. However, it is not sufficient for this project's programming. The code in this program should be capable of controlling the robot.

Although applying ARCL can resolve this issue, the connection must first be identified and ARCL commands tested prior to implementation.

ARCL supports multiple client/server connections through the TCP/IP socket. According to the Adept Lynx manual, the robot can communicate through the telnet, which is a protocol used on the internet or on local area networks. Telnet provides a bidirectional interactive text-oriented communication facility using a virtual terminal connection, in which user data is interspersed in-band with Telnet control information for a 8-bit byte-oriented data connection over the Transmission Control Protocol (TCP).

The Putty software can be applied in the initial stages in order check command correctness. Putty is a free and open-source terminal emulator, serial console and network file transfer application which supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. Moreover, it can connect to a serial port.

Following the steps below, a connection ARCL can be established:

1. On the Windows-based PC, open Putty
2. Start Telnet using the ARCL server address and port number specified in ARCL Server Setup Parameters. Figure 39 shows where the port (7171) and server address (192.168.200.210.) can be found.

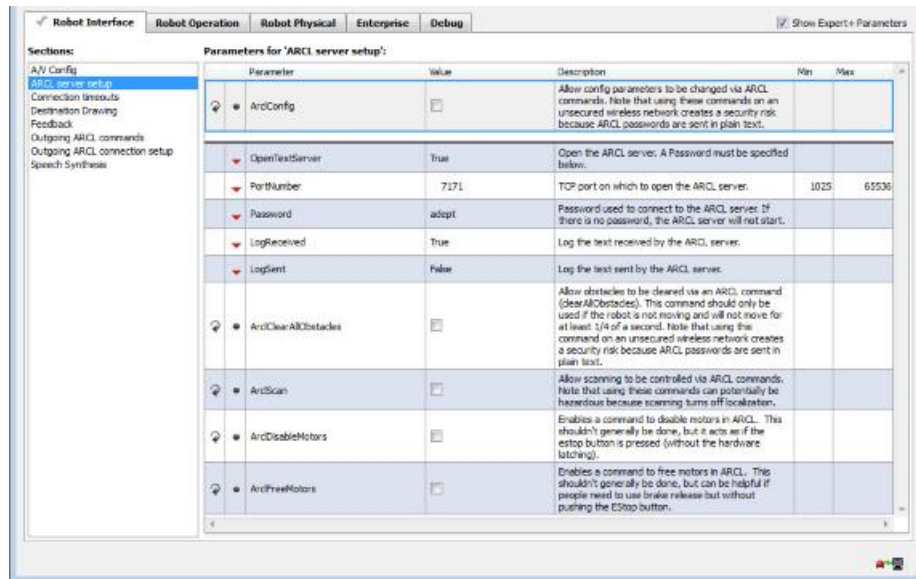


Figure 39. Server Information in MobilePlanner

3. Enter the password according to the instructions. After the user has successfully logged in, the server responds with a list of supported obstacles commands, each with a brief description. An excerpt of the content is displayed in Figure 40.

```

Enter password:
Welcome to the server.
You can type 'help' at any time for the following help list.
Commands:
getDateTime      gets the date and time
help             gives the listing of available commands
payloadQuery     Queries the payload for this robot
payloadSlotCount Queries for number of payload slots
queueCancel      Cancels an item by type and value
queuePickup      Queues a pickup goal for any appropriate robot
queuePickupDropoff Queues a pickup dropoff goal pair for any appropriate robot to do
queueQuery       Queries the queue by type and value
queueShow        Shows the Queue
queueShowRobot   Shows the status of all the robots

```

Figure 40. Content in putty

4. If needed, the echo off command can be executed to prevent input written by the user from echoing in
5. When the user finished their operation, the quit command can properly close the connection.

After the user connects to ARCL, available ARCL commands can be examined.

5.2.2 ARCL Implementation

After establishing a connection to the ARCL server and monitoring the mobile robot using ARCL commands designed through putty, implementation can be realized using Java.

Before organizing the command code, the telnet should first be created as in Figure 41.

```
//initialize a TelnetClient object
TelnetClient telnetClient = new TelnetClient();
//define the variables of writing and reading
InputStream inputStream=null;
PrintStream pStream=null;
```

Figure 41. Initialization of the related variables (1)

The client setting and initialization of the related variables are organized as shown in Figure 42.

```
public void init() throws SocketException, IOException {
    //set the length of time out for client
    telnetClient.setDefaultTimeout(50000);
    //set the address and port
    telnetClient.connect("192.168.200.210", 7171);
    //initialize variables
    inputStream = telnetClient.getInputStream();
    pStream = new PrintStream(telnetClient.getOutputStream());
    actions = new MoreActions();
}
```

Figure 42. Initialization of the related variables (2)

In executing connection work, ARCL commands can be written using Inputstream, the responses to which are then returned by PrintStream.

Although the getRoutes command is not applied in this Java application, it is useful for verifying which routes the robot should save. More precisely, it displays the list of route names found on the current map.

The method required for implementation is driving the mobile robot to pass through the routes. For this purpose, three methods which meet the requirements are identified, patrol command, patrolOnce command and patrolResume command. The patrol command initiates the named route's continuous patrol, while the patrolResume command calls the robot to continue the current route's navigation. As the two methods are applied, the stop command is applied, making the process more complex. Thus, the patrolOnce command, which calls the robot to patrol the specified route once, is the best choice for the user.

The code for patrolling the PickRoute route once is given in Figure 43.

```
pStream.println("patrolOnce PickRoute"); // write the command
pStream.flush(); // push the command to Telnet Server
```

Figure 43. Code for patrolling the route once (1)

For patrolling another route, the route name must be changed.

```
pStream.println("patrolOnce LeaveRoute"); // write the command
pStream.flush(); // push the command to Telnet Server
```

Figure 44. Code for patrolling the route once (2)

They all require the rely to be checked, with one such action displayed in Figure 45.

```
if (sBuffer.toString().trim().endsWith("Finished patrolling route PickRoute")) {
    break;
}
```

Figure 45. Code for patrolling patrol the route once (3)

6 CMMO CONTROLLER MODULE

This module administers the connection with the CMMO controller, thereby organizing its function. The CMMO controller version is CMMO-ST-EA-SY. This device can be charged using three connection types: FCT, CVE and Web servers as shown in Figure 46.
/9/

Connection	Changeover of device control
FCT	Device control can be assumed by all other connections (device control: enable FCT). When the device control is disabled, master control is restored to the I/O interface.
CVE	Device control can be assumed by all other connections and can assign device control to an existing active connection (object #3).
Web servers	Can take over device control from the I/O interface. When the device control is disabled, master control is restored to the I/O interface.

Figure 46. Three connections with the CMMO controller

In comparing the advantages and disadvantages of these three methods, it is clear that the Web server is outstanding. After deciding on the connection, the user can assume device control using the I/O interface. Hence, the parameter file which was changed in the FCT can be downloaded to the CMMO controller, assigning the Web server master control.

When the Web server assumes device control, the queries controlling the CMMO device require the user to identify and select certain requirements and efficiency for consideration. When homing method configuration is set according to user needs, the combination of homing and stop can control the electronic cylinders in lifting the battery through the CMMO controller. In order to avoid exceptions during execution, the homing query time length should be limited, so that the CMMO controller can order the electronic cylinders to lift successfully.

6.1 Environment configuration

6.1.1 Device configuration

Although the CMMO controller connection is determined using the web server, the FCT must nonetheless be applied so as to configure the parameter file. Thus, the components containing the CMMO controller and electric cylinders should be connected and configured in the FCT. Figure 47 shows the current motor controller identified and selected.

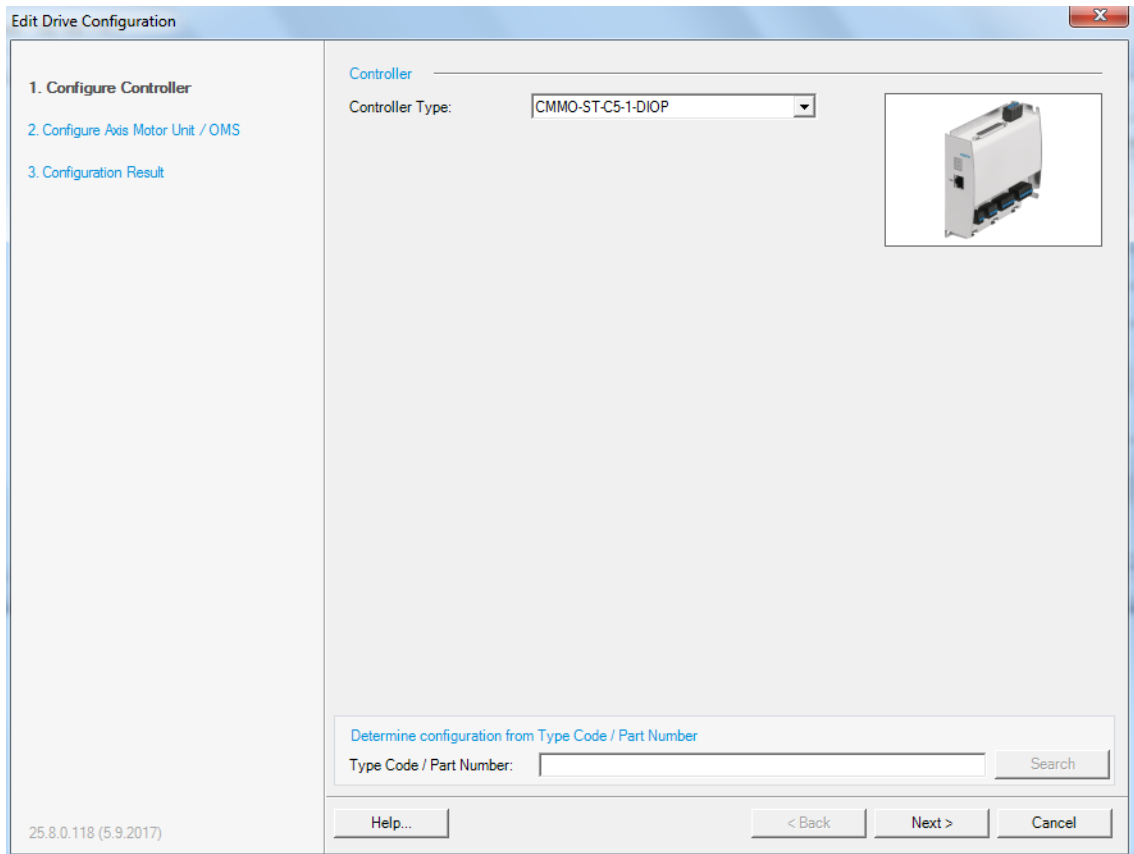


Figure 47. Configure controller

In Figure 48, the current Axis motor unit is identified and selected.

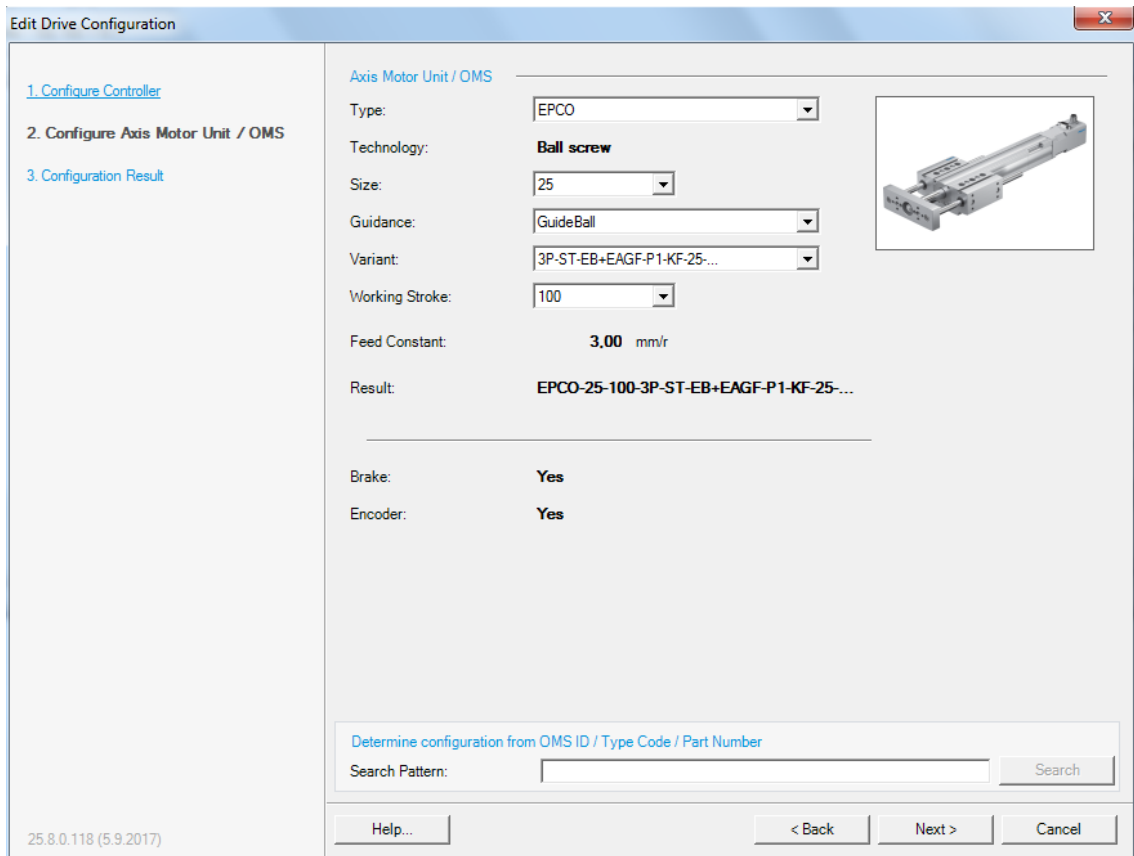


Figure 48. Axis Motor Unit configuration

In Figure 49, the correct control profile selected is valve profile rather than binary profile.

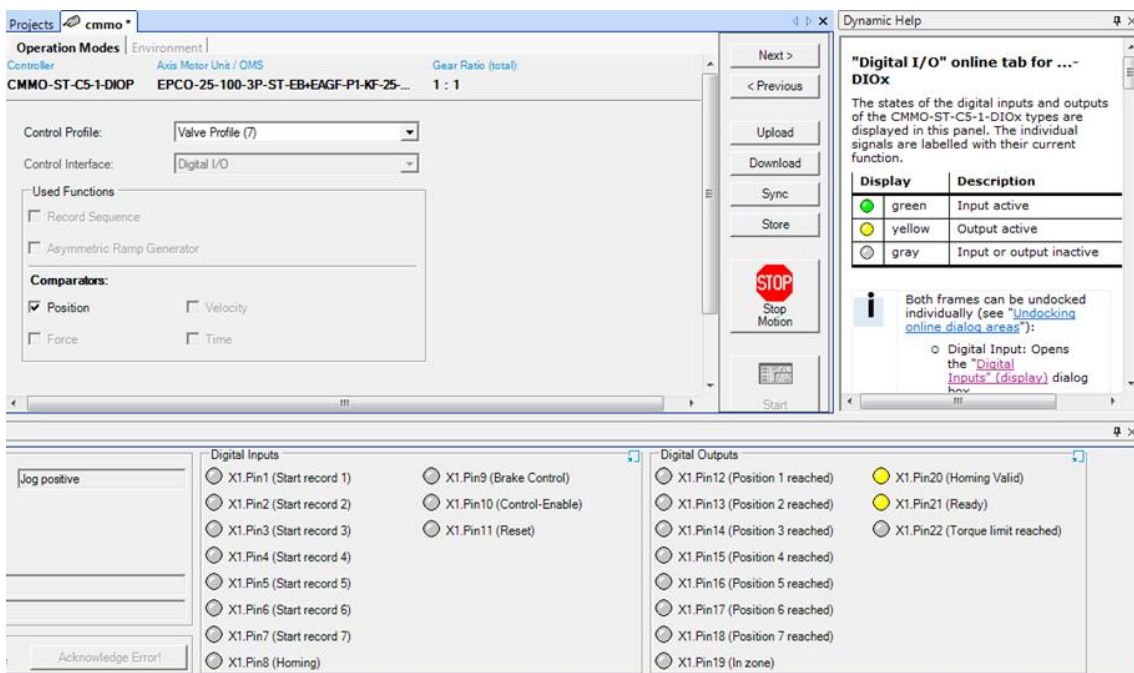


Figure 49. Valve profile selected

6.1.2 Parameter setting

For further functions, it is necessary to better understand homing action. After research, it is known that homing includes multiple methods.

In FCT (Festo Configuration Tool), the homing method can be configured in the Axis options panel and carried out on the “homing” tab using the set parameters.

The homing method containing "Target" and "Direction" is selected. The result is displayed in stylized form next to the selection as follows. Meanwhile, the methods available in the list depend on several conditions, meaning the user should select the most suitable method from the list.

Target:

- Block: Homing searches for a mechanical stop. When detected, the parameters are specified.
- Condition: A motor with an encoder operates in "controlled" mode.
- Homing Switch with Index: Only for motors with an encoder. Homing searches for a homing switch which end at the encoder's next index. The homing switch is parameterized and the motor with the encoder operated in "controlled" mode.
- Homing Switch without Index: Homing searches for a homing switch with parameterization.
- Position Actual Value: The current position becomes the homing position, with no movement required. This target is always within possible limits.

Direction:

When the target equals "Block" or "Homing Switch", its direction is determined as follows.

For linear axes:

- Negative: A spindle is "retracted", a slide moves in the motor's direction.
- Positive: A spindle "extends", a slide moves away from the motor.

For rotary modules (rotation axes) looking towards the drive shaft:

- Negative: The motor shaft turns anti-clockwise.
- Positive: The motor shaft turns clockwise.

In order to activate the homing switch, the homing switch with index negative method is applied. Here, the electronic cylinders belong to the linear axes, meaning they lift down and then up in executing a single cycle. This method description is shown in Figure 50.

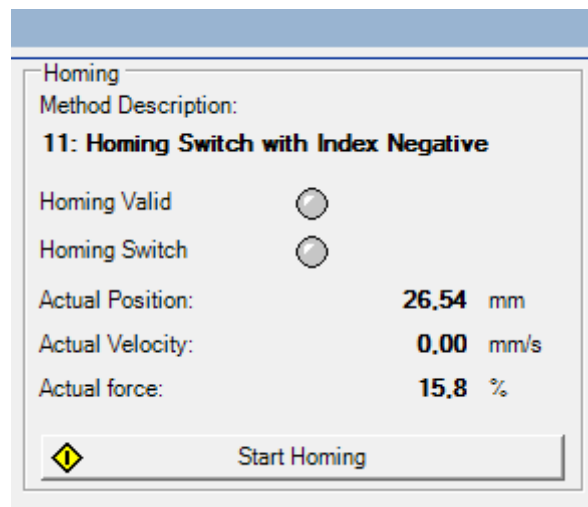


Figure 50. Homing Switch with index negative

6.1.3 Enabling the Web server

When the homing setting had been modified in FCT, the new parameter file is downloaded to the device, with the parameter uploaded to the PC from the web server page (192.168.178.1). Thus, next time the user can simply download the exited "Parameter.fpf" file directly to the device without changing FCT settings. The controlling panel, including uploading and downloading functions, is displayed in Figure 51.

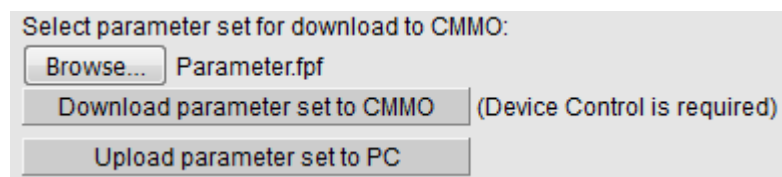


Figure 51. Uploading and downloading settings

Device Control (master control), which is an exclusive access right, ensures that one connection can control the actuator at any given time. Simultaneous control by multiple connections could cause serious problems for the actuator. There is always an approval signal for the connection controlling the device at any given time. After switching on the motor controller, the I/O interface has sole control over the device.

To facilitate the Web servers in assuming control over the current device using the I/O interface, control by other methods must first be disabled. Moreover, by enabling CMMO control through the web service as in Figure 52, a HTTP connection with the CMMO web service can be constructed.

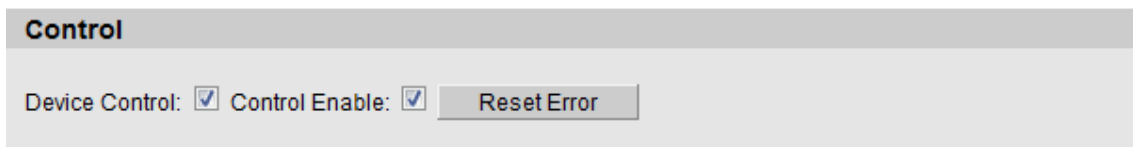


Figure 52. Device control enabling

6.2 CMMO controller programming

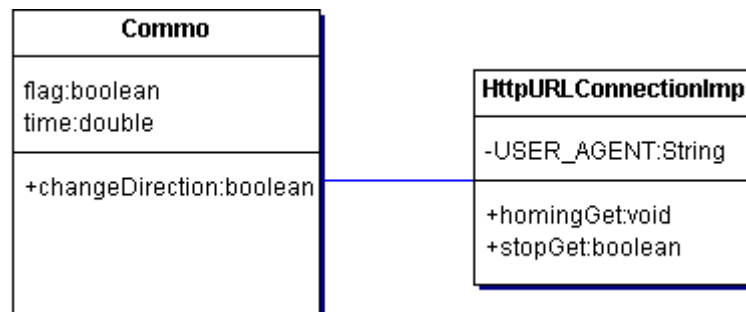


Figure 53. CMMO controller module architecture

This module contains two classes: HttpURLConnectionImp class and Commo class. The first class consists of two basic methods, the first of which, homingGet, builds the HTTP connection with the CMMO Web server, sending the homing query to the server by GET request. The other method, stopGet, also engenders HTTP communication with the CMMO Web server, forwarding the stop query to the server, again using GET.

In the CMMO class, the changeDirection method performs the functions of lifting electric cylinders. The two methods in the HttpURLConnectionImp class form part of the

changeDirection method, because when the homingGet is applied as part of changeDirection, the thread including the stop action waits until the cylinder lifts.

6.2.1 Web server queries

During homing query execution, the HTTP GET request with the URL "http://192.168.178.1/query?homing" retrieves the result and controls the motor so as to execute its actions. In the current IDE, the code is shown in Figure 54.

```
//send homing query
String url = "http://192.168.178.1/query?homing";

URL obj = new URL(url);
HttpURLConnection con = (HttpURLConnection) obj.openConnection();

// optional default is GET
con.setRequestMethod("GET");
```

Figure 54. Homing query code

Moreover, the corresponding web page is displayed in Figure 55.

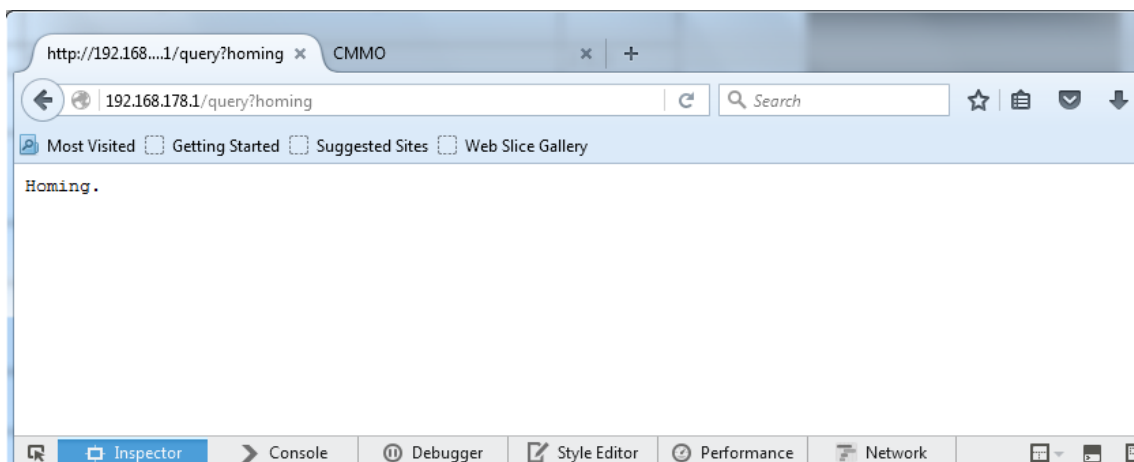


Figure 55. Homing Web page

Then, according to time counted, an HTTP GET request with the URL "http://192.168.178.1:80/query?stop" is sent to the server in order to stop the motor. In the current IDE, the code is given in Figure 56.

```
//send stop query
String url = "http://192.168.178.1/query?stop";

URL obj = new URL(url);
URLConnection con = (URLConnection) obj.openConnection();

// optional default is GET
con.setRequestMethod("GET");
```

Figure 56. Stop query code

The corresponding web page is displayed in Figure 57.

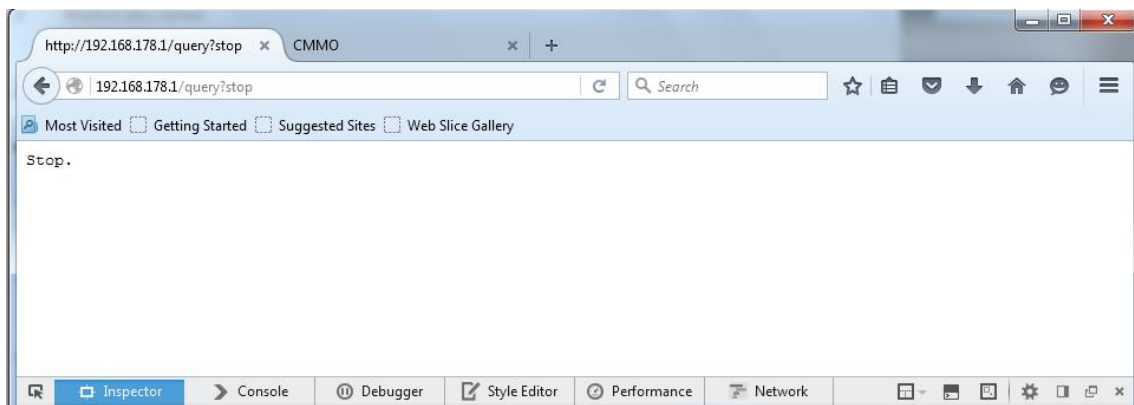


Figure 57. Stop Web page

7 STRATEGY MODULE

This module plays the most important role in the whole project, importing all the other modules and possessing the ability to call every method in their packages. Thus, it organizes the methods needed based on the workflow present in the overall structure, so that once the application runs, the program follows the sequence and completes the tasks.

Furthermore, there are two creative points which were brought to attention by my supervisor, the detection method based on the priority structure in the manufacturing schedule and the shortest path algorithm applied in route selection. In this chapter, the research regarding the two methods is outlined. These methods merit further analysis and development in the future so as to improve this application.

7.1 Initial programming



Figure 58. Strategy module architecture

In the main class initial version, although the code is simple, it can control the different modules and coordinate cooperation so as to finish the designated work line. In the main method, the “to do” list is obtained from the Odoo actions package before the first manufacturing order is produced. This order serves as the run method’s parameter. In the run method, several while loops are present because the main thread must wait for one robot to finish its activities before instructing another robot to run.

7.2 Future features

7.2.1 Scheduling Algorithm Based on Priority Table

In implementing this project, the priority table applied is the earliest deadline first (EDF). In other words, this EDF policy assigns the highest priority to the task with the earliest deadline. Similar algorithms include the rate-monotonic (RM) algorithm, least slack first (LSF for short), highest value first (HVF) and maximum value density first (HVDF).

However, it is not sufficient for priority to be determined solely by a certain characteristic parameter. Although these algorithms are optimal under normal system load, when overload occurs, the system cannot guarantee that all tasks meet the deadline. Thus, algorithms like EDF experience drastic performance degradation, even succumbing to the domino effect. Moreover, deadlines and short idle tasks are not necessarily the most critical, but even during overload, the system must guarantee the timely completion of critical tasks. This thus supports system performance, causing system failures and even crashes. Considering this risk, EDF is not the best algorithm for this project.

Many studies have demonstrated the importance of priority-driven scheduling algorithms expressing task criticality and deadline by using two independent characteristic parameters. One algorithm, the best effort (BE) algorithm, is based on task value density. However, experimental results show that BE has a substantially large system overhead in the event of system overload. Certain articles have mentioned the criticality-deadline first (CDF) algorithm, according to which each task is assigned a priority based on relative deadline and criticality when it arrives. Period or criticality, as feature parameter algorithms, can greatly improve overall system performance. Unfortunately, this algorithm still carries certain risks.

In summary, the above algorithm has the following shortcomings: (1) Since the task deadline and value are two completely different concepts with distinct value rangers, they cannot be weighted in a simple manner; (2) Since the operating system or scheduling system usually only supports a limited priority, the CDF priority must be converted to the system's priority level, causing different tasks to have the same priority; (3) Heuristic algorithms, such as BE, have large system overheads, especially during system overload, with the higher overload processing overhead affecting system performance more severely. In order to overcome these problems, a real-time scheduling algorithm is

proposed, for which task priority is assigned by ordering which fully reflects the sequence relationships between task characteristic parameters. This algorithm not only serves in synthesizing task value with two deadline feature parameters, but also in any other tasks with two feature parameters involved in the scheduling of comprehensive consideration. Moreover, this algorithm focuses on the priority table design method for synthetic task deadline and value. Two scheduling algorithms, EDV (earliest deadline value) and VED (value earliest deadline), are proposed. In this project, because every product-battery module is labelled with the same price, the value comparison does not differ from the amount comparison in their findings.

7.2.1.1 Deadline/value priority table design

The objective of the deadline/value priority table design method is to comprehensively consider task deadline and value during task scheduling so that the system may gracefully collapse under overload while avoiding a domino effect. First, the principles of algorithmic scheduling are clarified:

- The earlier the task's deadline, the greater its value and the higher its priority;
- For those tasks with identical deadlines and values, the first-arrival takes priority.

Figure 60 shows the prioritization schedule based on deadlines/values, where the arrows indicate task priority. In this priority assignment method, the task deadline sequence goes in ascending order, starting with the earliest deadline. Originally, the task was more straight forward, with the task value sequence ranked in descending order, starting from the greatest value tasks. For tasks with identical deadlines, the higher its value, the higher its priority. Meanwhile, for tasks with identical values, the earlier its deadline, the higher its priority. In the priority table shown in Table 1, each task has a priority level.

$$P = i + j \quad (1)$$

Where i and j denote the position of task deadline and value in their respective sequences. The tasks identified on each diagonal line in the figure have the same priority level P , but for tasks with identical priority, the scheduling order must be displayed in different directions according to arrow direction. Task priority, p , is calculated as follows:

$$p = (i + j - 1) * (i + j - 2)/2 + i \quad (2)$$

The smaller the p-value, the higher the priority. For tasks with identical priority, the scheduling algorithm is inclined to priorities tasks with earlier deadlines. This priority design pattern provides a deadline/value scheduling algorithm for integrated tasks. These algorithms are called EDV algorithms.

Similarly, another design pattern is present in task priority table, as shown in Figure 60. For tasks with identical priority, the scheduling algorithm prioritizes larger tasks. According to this model, task priority is calculated as follows:

$$p = (i + j - 1) * (i + j - 2)/2 + j \quad (3)$$

Scheduling algorithms based on this pattern are called VED algorithms.

Although task priority is assigned according to the priority table, this method does not conduct calculations in advance. Instead, the priority table for the dynamic organization task at runtime is used, achieving priority-based task scheduling.

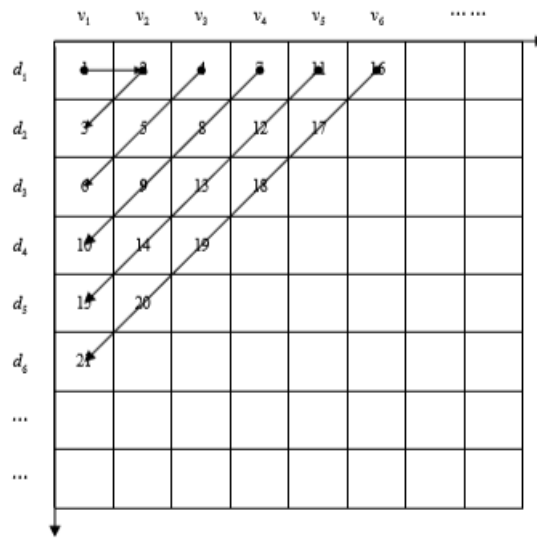


Fig.1 EDV priority table design

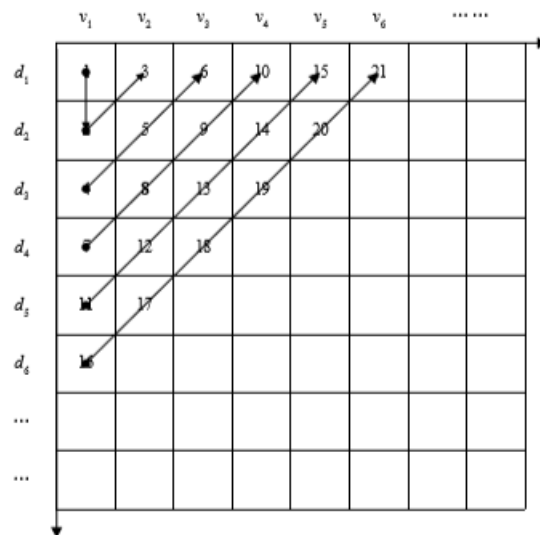


Fig.2 VED priority table design

Figure 59. Priority table design of EDV and VED

7.2.1.2 Deadline/value priority table design implement

This section uses EDV as an example of implementing a scheduling algorithm based on the priority table. Based on the priority table’s characteristics, the task deadline sequence and value sequence are realized as a deadline-based task linked list Qd and value-based respectively. The task chains Qv, Qd, and Qv are bidirectional circular linked lists with free head nodes, together forming a logical priority table. The reason behind the use of a doubly linked list is the facilitation of task node insertion and removal. The structure and list of TaskNodes are defined as shown in Figure 60:

```

typedef struct TaskNode{
int a; //Arrival time of the task
int c; //The remaining execution time of the task
int d; //Deadline of the task
int v; //The value of the task
int i,j; //The task's row and column subscripts
int p; //The priority of the task
struct TaskNode *pdprior, *pdnext; //Deadline based task list pointer
struct TaskNode *pvprior, *pvnext; //Value-based task list pointer
}TaskNode, *TaskLink
TaskLink dhead;
TaskLink vhead;
TaskNode *pactive

```

Figure 60. Code for EDV table design (1)

When a new task arrives, the system runs the priority algorithm, inserting the task into Qd and Qv, and calculating task priority so as to determine whether it must preempt the task currently being executed.

The task receiving strategy algorithm is described as follows:

- 1) Let pnew point to the new task, with an initially Null pointer field;
- 2) Insert the task pnew into Qd

```

(a) pd=dhead->pdprior; // Define the pointer, point to the end of Qd Point
(b) i = pd-> i + 1; // initialize the new task in the priority table row subscript
(c) while ((pd!= Dhead) and (pnew-> d <pd-> d)) { //Determine where the new task is in Qd
(d) pd->i++; pd=pd->pdprior; i--}
(e) pnew->i=i; //Set he row index of new task in priority table
(f) insert(pd,pnew); //after inserting pnew into pd

```

Figure 61. Code for EDV table design (2)

- 3) Insert the task pnew into Qv

```

(a) pv=vhead->pvprior; //define the pointer to point to the tail of Qv
(b) j= pv->j+1; //initializes the new task in the column of the priority table subscript
(c) while ((pv!=vhead) and (pnew->v>pv->v)) { //Determine the position of the new task in Qv
pv->j++; pv=pv->pvprior; j--}
(d) pnew->j=j; //Set the new task under the column in the priority table
(e) insert(pv,pnew); //after inserting pnew into pv

```

Figure 62. Code for EDV table design (3)

- 4) Calculate task priority and determine whether it should preempt the task currently being executed

```
(a) pnew->p=(i+j-1) *(i+j-2)+ i;
(b) if (pnew->p>pactive->p) { //The new task pnew preempts the current task
    pactive; pactive=pnew;}

```

Figure 63. Code for EDV table design (4)

The algorithm searches backward from the two linked lists' tail nodes so as to determine the new task node's position and priority. In the worst-case scenario, Qd and Qv must be rescanned, making algorithm complexity equal $O(2n)$, where n is the number of tasks in the current priority table.

When a task has been executed, or fails before the deadline expires, the system must invoke the task completion/frustration strategy in order to remove the completed or aborted task node from the priority list, adjusting subsequent tasks' rows and columns accordingly. Next recalculated tasks are marked for their priority, with the highest priority task selected for execution. The algorithm for the task completion/birth strategy is described as follows:

- 5) Remove the currently completed task from the priority list

```
(a) // From the Qd list Remove the task node
pd=pactive->pdprior->pdnext=pactive->pdnext;
pactive->pdnext->pdprior=pactive->pdprior;
(b) //Remove the task node from the Qv list
Pv=pactive->pvprior->pvnext=pactive->pvnext;
pactive->pvnext->pvprior=pactive->pvprior;
(c) delete pactive; pactive=NULL;

```

Figure 64. Code for EDV table design (5)

- 6) Subscript all subsequent tasks to the Qd list minus 1

```
while (pd!=dhead)
|{ pd->i--; pd->p=(pd->i+pd->j-1) *(pd->i+pd->j-2)/ 2+i;
| pd=pd->pdnext;}

```

Figure 65. Code for EDV table design (6)

- 7) Subscript for all subsequent tasks in the Qv list minus 1

```
while (pv!=vhead) {
    pv->j--; pv->p=(pv->i+pv->j-1) *(pv ->i+pv->j-2)/2+i;
    pv=pv->pvnext;}

```

Figure 66. Code for EDV table design (7)

8) Scan the head node of the Qd list in order to determine the highest priority task

```
(a) pd=pactive=dhead ->pdnext;
(b) hp=0; //Record the highest priority of the currently scanned task
    if (pactive!=dhead) {
        hp=pactive->p; pd=pd->pdnext;}
(c) While(pd!=dhead) {
    if(pd->p>hp) {hp=p; pactive=pd;}
    pd=pd->pdnext;
}
(d) if (hp!=0) //task pactive occupies the processor and begins execution
```

Figure 67. Code for EDV table design (8)

The algorithm first removes the given task node from the linked list Qd and Qv, updating the task's row and column indices based on the next node of the removed node so as to recalculate task priority. In difficult cases, Qd and Qv must be scanned once. The process of determining the highest priority task involves scanning the Qd linked list (or alternatively the Qv linked list), with time complexity $O(n)$. Therefore, total algorithm complexity is $O(3n)$, where n is the number of tasks in the current priority table.

7.2.2 Going through all points in the shortest path

Initially, the mobile robot was not responsible for delivering all materials. However, product transfer necessary part in factory work lines. Thus, the process of passing through all points in the shortest path is now discussed.

After considering the relevant problem, the main issue is requiring the mobile robot to visit a group of points with minimal effort. Thus, the problem bears similarities with the Travelling Salesman Problem (TSP). Moreover, the difficulty faced can be solved using the solution to the TSP.

7.2.2.1 TSP problem

The Travelling Salesman Problem (TSP) can be translated into the traveling salesman problem and salesman problem. It is one of the famous problems in mathematics. Suppose a travel businessman wants to visit n cities. He must choose the path to be taken. However, he can only visit each city once and must return to the city from which he started. The path selection goal is minimizing the required path distance.

The TSP problem is a combinatorial optimization problem, demonstrable using NPC computational complexity. The TSP problem can be divided into two categories, Symmetric TSP and Asymmetric TSP. All TSP problems can be outlined graphically:

```
V = {c1,c2,...,ci,...,cn}  i = 1,2,...,n
//this is a set of all cities. Ci represents the i city, n is the number of cities;

E = {(r,s): r,s ∈ V}
//this is a collection of connections between all cities;

C = {crs: r,s ∈ V}
//this is the cost measure of the connection between all cities (usually the distance between cities);

If crs = csr, then the TSP problem is symmetric, otherwise it is asymmetric.
```

Figure 68. Solution to TSP problems

A TSP problem can be expressed as:

Solving the traversal graph $G = (V, E, C)$, all nodes are visited once and the starting node is returned to, minimizing the path cost.

7.2.2.2 Greedy algorithm

The greedy algorithm is a common and simple algorithm for solving the optimization problem. The greedy algorithm always makes the best choice for now, with every action seeking to maximize current utility. However, it is worth noting that the greedy algorithm cannot obtain an overall optimal solution for all problems. The chosen greedy strategy must have a no-failure effect, meaning the process after a certain status has no effect on the previous state.

1. The basic idea behind the greedy algorithm

Starting from an initial solution triggers the gradual approaching towards a given goal so as to identify a better solution as quickly as possible. The algorithm stops when a certain step can no longer be advanced. The general steps are as follows:

- 1) Establish a mathematical model for describing the problem;
- 2) Divide the problem into several sub-problems
- 3) Solve each subproblem
- 4) Combine subproblem solutions so as to fully solve the original problem

2. Greedy algorithm implementation framework

The greedy algorithm does not have a fixed framework, but the key to its design is the choice of the greedy strategy. The premise of the greedy strategy is that the local optimal strategy can lead to a global optimal solution.

Start with an initial solution;

```
While (toward a given total goal before further)
{
    Using feasible decisions, find a solution element of a feasible solution;
}
A feasible solution that combines all solution elements into a problem;
```

Figure 69. Greedy algorithm solution

3. Greedy algorithm problems

- 1) It is not guaranteed that the final solution obtained is optimal;
- 2) It cannot be used to calculate maximum and minimum solutions;
- 3) It can only be used under specific conditions, such as greedy strategies having no repercussions.

7.2.2.3 Greedy algorithm for solving TSP problems

In order to initialize the TSP problem values, every node to be visited should be indexed, with the distances (c_{rs} and c_{sr}) between these points kept as the item of the two-dimensional array, for which the index represents the specified goal. The distance between ARCL commands can be used using the Omron mobile robot so as to detect the distances between points. Then, the value of these distances can be assigned to c_{rs} and c_{sr} .

Using the Greedy strategy, the mover traverses all reachable next nodes, selecting the nearest node as the next node. The basic idea is to traverse all reachable next nodes from a node by selecting the nearest node as the next node. Then, the nodes passed are marked and the next node selected as the current node. Thus, the greedy strategy is repeated until the solution is realized. Once all nodes are marked as visited, the whole problem ends and the final path array becomes the result.

8 RESULTS

8.1 Preparatory work

Prior to implementation, the materials and environment should be configured and prepared. In other words, the machines should be charged and the data, including the map, parameter file and ABB robot program, should be opened and operated.

8.2 Implementation

8.2.1 Starting the project

In order to execute this application, the Java application should first be run as shown in Figure 70. In the code-containing eclipse, the main class should be opened and run as the Java application.

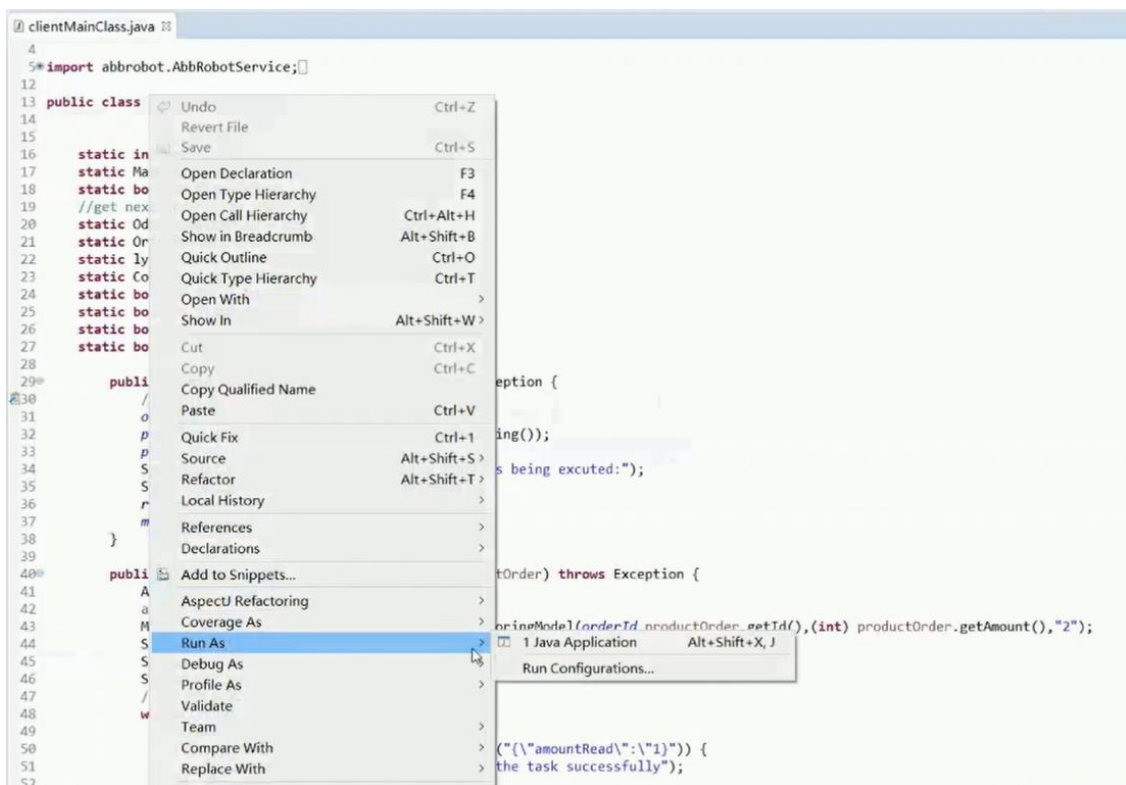
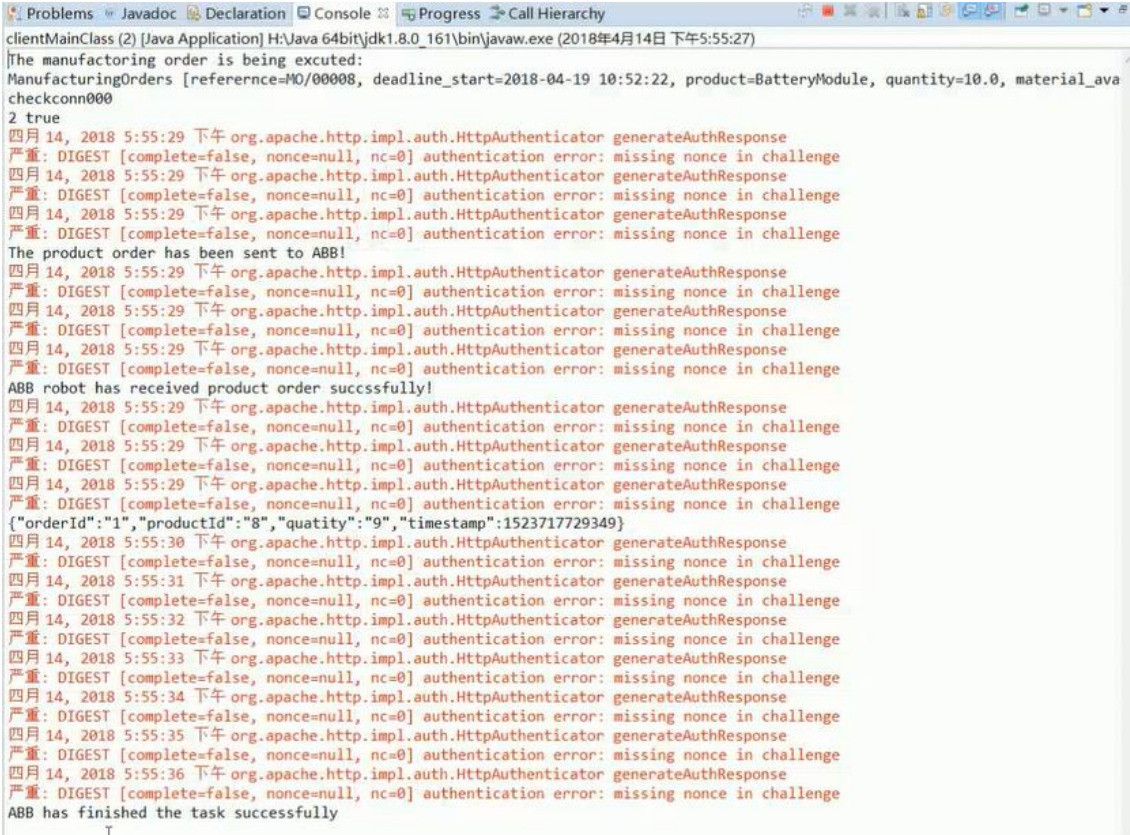


Figure 70. Starting the project

8.2.2 Project Process

After application initialization, the manufacturing order data is captured and processed. In order to provide the information for every status, the console logs the current process,

sending the next order to the ABB robot. The robot's actions of robot are shown in Figure 71.



```

clientMainClass (2) [Java Application] H:\Java 64bit\jdk1.8.0_161\bin\javaw.exe (2018年4月14日 下午5:55:27)
[The manufacturing order is being executed:
ManufacturingOrders [reference=MO/00008, deadline_start=2018-04-19 10:52:22, product=BatteryModule, quantity=10.0, material_ava
checkconn000
2 true
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
The product order has been sent to ABB!
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
ABB robot has received product order succssfully!
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:29 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
{ "orderId": "1", "productId": "8", "quantity": "9", "timestamp": "1523717729349" }
四月 14, 2018 5:55:30 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:31 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:32 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:33 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:34 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:35 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
四月 14, 2018 5:55:36 下午 org.apache.http.impl.auth.HttpAuthenticator generateAuthResponse
严重: DIGEST [complete=false, nonce=null, nc=0] authentication error: missing nonce in challenge
ABB has finished the task successfully
  
```

Figure 71. Result capture (1)

Then, the robot program information is updated according to the order data received by the ABB robot as shown in Figure 72.



```

IRB_1200_remote_control_by_web_demo:View1 1200-100380 (Station) x
T_ROB1/mMoverob x
1 MODULE mMoverob
2 PERS num XCoordinate:=0;
3 PERS num YCoordinate:=0;
4 PERS num ZCoordinate:=0;
5 PERS num orderId:=1;
6 PERS num productId:=8;
7 PERS num quantity:=9;
8 PERS num amountRead:=1;
9 VAR num torque;
  
```

Figure 72. Result capture (2)

After the ABB robot is assigned the new order and has finished its work, the mobile robot moves according to the application's control. The route followed is provided in Figure 73. The red point on this map, scanned from the real laboratory environment, represents the mobile robot and changes in real time.

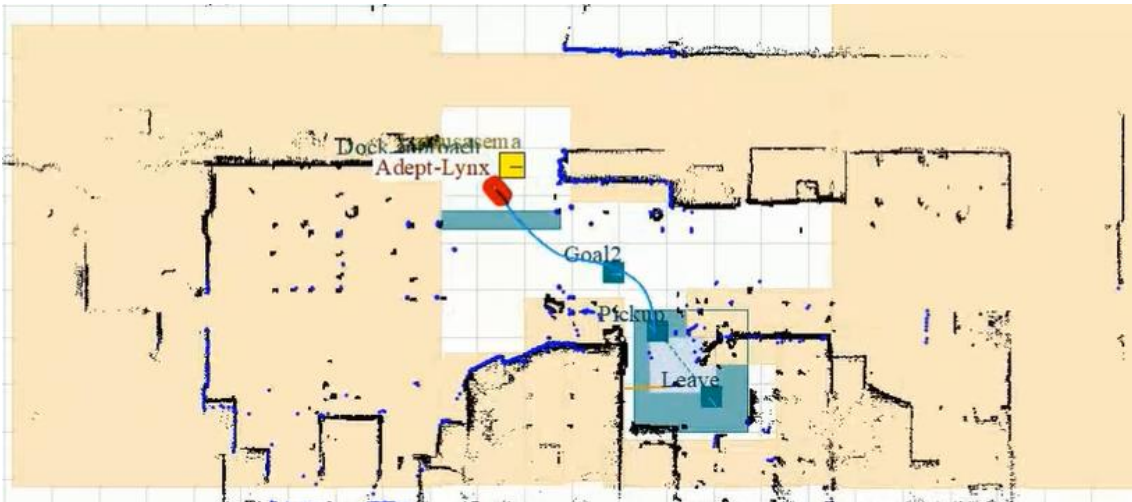


Figure 73. Mobile robot environment

After the ABB robot is assigned the new order and has finished its work, the mobile robot moves according to the application's control. The route followed is provided in Figure 73. The red point on this map, scanned from the real laboratory environment, represents the mobile robot and changes in real time.

Figure 73 shows the Omron robot's reaction during the experiment.



Figure 74. Omron robot reaction

In the application, the program calls the method to connect with the CMMO controller and control the electric cylinder. The information in the console is given in Figure 75, while the CMMO controller module reaction is shown in Figure 76.

```

-----
The adept lynx robot is controlling to pick up!
Interrupted: Parking
Patrolling route PickRoute once
WaitState: Waiting 5 seconds with status "Waiting"
WaitState: Waiting completed
Finished patrolling route PickRoute
-----

Control electric cylinder to move up!

```

Figure 75. Result capture (3)

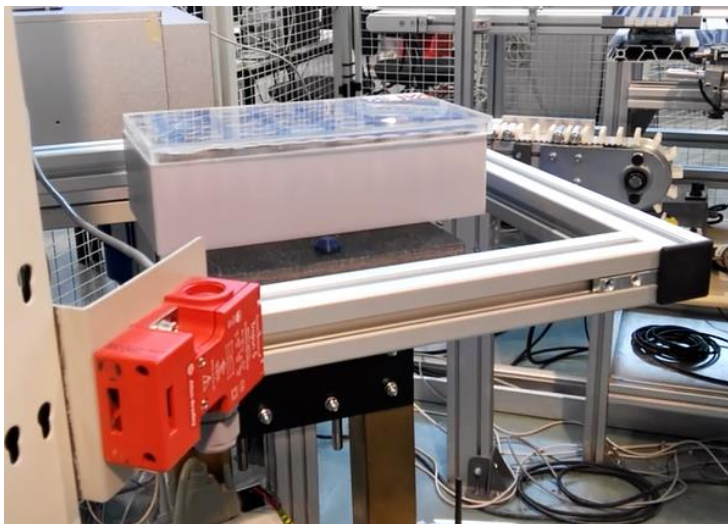


Figure 76. CMMO controller reaction

When the electric cylinder finishes a single cycle and has picked up the battery module, the mobile robot executes its tasks again by moving out of the operating space. The record of this in the console is provided in Figure 77.

```

-----
Control electric cylinder to move up!
-----
Control electric cylinder to move down!
-----

The adept lynx robot is controlling to leave!

Patrolling route LeaveRoute once
Finished patrolling route LeaveRoute
-----

```

Figure 77. Result capture (4)

Finally, the manufacturing order, which was previously selected for execution, is updated as given in Figure 78. Thus, factory staff can identify which order should first be dealt with.

The screenshot shows a web interface for Manufacturing Orders. The main title is "Manufacturing Orders / MO/00008". There are buttons for "Edit", "Create", "Print", "Action", "Lock", "Inventory Moves", "Confirmed", "In Progress", and "Done".

Order Details:

- MO/00008**
- Product:** BatteryModule
- Quantity To Produce:** 10.000
- Bill of Material:** BatteryModule
- Deadline Start:** 04/19/2018 13:52:22
- Responsible:** Administrator
- Source:**

Table of Consumed Materials:

Product	To Consume	Reserved	Consumed
BatteryConnector	10.000	10.000	1.000
BatteryFrame	10.000	10.000	1.000
BatteryCell	880.000	880.000	88.000

Figure 78. Selected order status

9 SUMMARY

Nowadays, applications for optimizing factory work flow are becoming increasingly popular. The main idea behind this thesis is the design of a system for controlling different robot types for completing work lines. As the middle level, the special MES (manufacturing Execution System) functions like a bridge connecting the ERP system, Odoo, and the execution module, the robots.

Prior to the Odoo module's development, the Odoo server and related database must first be prepared. Then, the required materials must be logged in the manufacturing module and an order list organized. In this application, the program can obtain the next manufacturing order according to the deadline through the XML-RPC web services.

After order acquisition, the execution components are initialized. In the ABB robot module, order information is assigned to the ABB robot's program deployed in the RobotStudio based on the REST web services. The Omron mobile robot then goes to the specified place under the application's control according to the map scanned from the lab. When the mobile robot finishes its tasks and stops, the CMMO controller lifts the electronic cylinders up and down to pick up the battery module. After the mobile robot leaves the specified place, the manufacturing order in Odoo is updated as done.

When this application's basic program was mostly finished, the supervisor Dr. Liu provided valuable suggestions regarding the strategy module. Taking his insights into consideration, the scheduling algorithm based on priority table as well as going through all points in the shortest path were researched and added as future features.

During content analysis, greater algorithm knowledge was acquired. Furthermore, the details for several robot types were documented, allowing the user to master certain skills involved in controlling them. It is thanks to the accumulation of these materials that this project could be realized.

Finally, the thesis as well as the final project worked out well. This process provided the user with an unforgettable experience.

10 REFERENCES

/1/ Odoo Web API. Accessed 1 April 2016

https://www.odoo.com/documentation/10.0/api_integration.html

/2/ XML RPC web services in Odoo

https://doc.odoo.com/6.0/developer/6_22_XML-RPC_web_services/#id1

/3/ Enterprise resource planning. Accessed 24 December 2016

https://en.wikipedia.org/wiki/Enterprise_resource_planning

/4/ Diagram showing some typical ERP modules. Accessed 25 August 2013

https://en.wikipedia.org/wiki/Enterprise_resource_planning#/media/File:ERP_Modules.png

/5/ Odoo download page.

<https://www.odoo.com/page/download>

/6/ Odoo install page

<https://www.odoo.com/documentation/11.0/setup/install.html>

/7/ ABB Robot manual

http://developercenter.robotstudio.com/blobproxy/devcenter/Robot_Web_Services/html/index.html

/8/ Diagram showing ABB Robot Web Services. Accessed 25 August 2013

http://developercenter.robotstudio.com/blobproxy/devcenter/Robot_Web_Services/html/images/rws.png

/9/ CMMO Controller manual. Accessed June 2015

https://www.festo.com/net/SupportPortal/Files/423889/CMMO-ST-EA-SY_2015-06b_8039016g1.pdf

/10/ Features of the Omron robot. Accessed 2018

<https://industrial.omron.us/en/products/ld-series#features>

/11/ Adept Introduces Lynx Autonomous Mobile Platform. Accessed 23 Jan 2013

<https://spectrum.ieee.org/image/MjIwNjAxNA.jpeg>

/12/ Adept Lynx Platform User's Guide. Accessed January 2015

http://www1.adept.com/main/KE/DATA/Mobile/Lynx_UG.pdf

/13/ XML-RPC Architecture of Odoo

https://doc.odoo.com/doc_static/6.0/images/tech_arch.png

/14/ Diagram showing CMMO Controller. Accessed June 2015

https://www.festo.com/net/SupportPortal/Files/423889/CMMO-ST-EA-SY_2015-06b_8039016g1.pdf#page=14