Co-funded by the
Erasmus+ Programme
of the European Union

TECHNICAL SCHOOL PIROT, SERBIA

In association with:

- **Scuola Di Robotica** – Genova, Italy, and;
- **NGO Osveženje** – Pirot, Serbia,

presents the following didactic material for VET schools and institutes:

INDUSTRIAL ROBOTICS – operating and programming

This material has been compiled and prepared for purposes of Erasmus project by:
***Dejan Aleksovski**, grad. mech. eng. - department of mechatronics, Technical School Pirot*
***Fiorella Operto** – M.Sc. in Philosophy, Scuola di Robotica*
***Luca Gillardi** – B.Sc in Physics, Scuola di Robotica*
***NGO "Osveženje"**, Pirot*

Pirot, Genoa, 2023/24

Co-funded by the Erasmus+ Programme of the European Union

TABLE OF CONTENTS

INTRODUCTION

Welcome to "INDUSTRIAL ROBOTICS - Operating and Programming." This handbook is designed to guide you through the fascinating world of industrial robotics, from foundational principles to advanced programming techniques. Whether you are a student, an engineer, or a seasoned professional, you will find valuable insights and practical knowledge to enhance your understanding and skills in operating and programming industrial robots. Join us on this journey as we explore the cutting-edge technologies and applications that are shaping the future of manufacturing and automation.

Introduction to "Robot Curriculum Development" project

This Handbook "INDUSTRIAL ROBOTICS - operating and programming"; is one of the results of an Erasmus+ Small Scale project, "Robotics Curriculum Development" Project N. KA210-VET-2C05C51F, coordinated by the Tehnička škola of Pirot (TSP – Pirot/Serbia), in Serbia and whose partners are Scuola di Robotica (SDR - Genoa/Italy) and the NGO Osvezenje (Pirot/Serbia).

The core of the project is the drawing of a curriculum dedicated to manipulative robotics and robotic arms for students from technical and vocational schools. The two courses (LTTA) held in Serbia (TSP) from 6th to 10th of November 2023 and in Italy (SDR) from 15th to 20th April 2024 and attended by Italian and Serbian teachers and students were preparatory to the drafting of this Manual, which includes various activities carried out by the participants, and other references and insights.

How to use this manual

This Handbook offers lessons derived from workshop activities focusing on both theory and Use Cases of Robotics, in order to prepare students in vocational education and training for their future work. The aim of this project is to create a completely new curriculum in robotics, together with appropriate teaching materials that will be jointly created by the project partners and that will help students in technical and vocational schools to adapt to their working environment as soon as possible.

Given the limitations of time and context, this Handbook cannot, and do not want to be a comprehensive manual – such as university-level courses in Systems and Control Theory for Automation. Our aim has been to provide participants in the project, and all those who will use it, with a general overview of the state of the art in industrial robotics, the use of robotic arms in industry, and beyond, so that students can plan their profession, or future course of study, with sufficient knowledge.

Keep in mind that the main target for the project are VET schools, so some concepts are voluntarily simplified or omitted. We tried to provide a general overview, and then some applications examples. In the example sections, since some real platform is necessary, as done during the LTTA we tried to provide a few example picked from the real-life robots that may be found in industry, and that can also be used inside the educational system. Remember that you may keep inspiration and adapt our manual to different robots brands.

The needs analyses results

SKA Analysis - The Analysis of Skills, Knowledge, and Attitudes for the Application of Robotics in Industry was conducted within the "**Robotics Curriculum Development**" project ("Project") under the Erasmus+ program Key Action 2: SMALL-SCALE PARTNERSHIP PROJECT **No 2022-2-RS01-KA210-VET-000097484**, carried out by the project partners: Technical School Pirot, UG Osveženje from Pirot, and the School of Robotics from Genoa. This field was chosen based on the trend and the growing need for the use of robots in industry, as well as the lack of suitable personnel in this field both in Serbia and Italy. Due to technological advancements and changes in the labor market, competencies that young people do not acquire through formal education at the appropriate level are increasingly required, making them unable to meet the needs of the economy.

This analysis aims to identify the knowledge, skills, and attitudes demanded by employers in the industry to adapt the existing robotics curricula at the Technical School, based on these observations, and to meet the industry's requirements.

Methodology

The SKA Gap methodology (Skills, Knowledge, Attitude method) was developed through the work of SFIVET (The Swiss Federal Institute for Vocational Education and Training) and represents a method for profiling occupations as a starting point for building a unique model for developing education/training programs based on its application and results. As an expert organization dealing with vocational education and training, SFIVET has developed a special methodology for job analysis and determining ways to profile future trends and potential deficiencies within professions, in the context of specific economic activities. Based on the results of the SKA analysis, an occupation and competency profile is created, from which a curriculum for training new employees for the given occupation is developed. The competency profile, as a result of the SKA analysis, includes a list of necessary skills, knowledge, and attitudes that employees in a specific occupation must possess to successfully perform the given job.

The SKA analysis was conducted to gather information and identify the skills, abilities, and attitudes required for successful job performance in the field of robotics. The SKA analysis was carried out following the basic principles of the SFIVET methodology, which was adapted for this project's needs.

Implementation Period of the SKA Analysis: The analysis was conducted from June 2023 to May 2024. A series of steps were taken for the analysis:

Defined the methodology for conducting the analysis with detailed instructions and working materials (questionnaire).

Translated the material into English and delivered it to partners in Italy.

Held workshops in Serbia and Italy.

Prepared a report on the SKA analysis.

The analysis was conducted through workshops with direct job performers in the field of robotics and with supervisors responsible for worker education within the companies. The analysis included companies from Serbia and Italy (from Serbia - Tigar Tires from Pirot and TERI ENGINEERING DOO BEOGRAD - OGRANAK TERI ENGINEERING PIROT, VIRS doo, Vertus Zemun doo, as well as KUKA and FANUC representatives in Serbia. In Italy, we visited the company Komau and the Technical School in Sestri Levante, which studies robotics in its program). We also had the opportunity to conduct an interview with Professor Žarko Čojbašić from the Faculty of Mechanical Engineering at the University of Niš, whose narrow scientific field is automation and robotics, as well as with representatives from the University of Kragujevac, who are developing an innovative robotic platform for waste sorting. We also used desk analysis of existing curricula from secondary vocational schools that study robotics and automation and qualification standards in the field of robotics in the Republic of Serbia in our research.

The analysis identified areas of competencies necessary for successful job performance, that is, the skills, knowledge, and attitudes an individual needs to be successful in the field of robotics as recognized in the analyzed sample and most prevalent in the market. Based on the conducted SKA analysis, input information was prepared for the development and supplementation of the Robotics curriculum, especially in the domain of practical knowledge.

SKA Analysis Report

I Workshop 18.08.2023. - TERI ENGINEERING DOO BEOGRAD - OGRANAK TERI ENGINEERING PIROT The first workshop was conducted with employees working in Technical Support for machine operations in the maintenance sector. The purpose of the workshop was for direct executors to describe their job as Robotics Technicians in detail. In this company, their task is to improve machine reliability and provide technical support for machines operating with predominantly KUKA robots.

II Workshop 31.08.2023. - Tigar Tyres The second workshop was conducted with employees of Tigar Tyres. The focus was on the maintenance of robotic arms in the production process. The purpose of the workshop was for direct executors to describe their job as Robotics Technicians, whose task is to maintain machinery that includes robotic arms from various manufacturers (Fanuc, Kuka, Yaskawa). Representatives from the human resources sector, engineers, direct executors of jobs at machine stations with robotic arms, and workers involved in the maintenance of machinery participated in the workshop.

First Workshop in Italy 13.07.2023. - COMAU in Turin The first workshop in Italy was held with representatives of COMAU, a company engaged in robot manufacturing. During the visit, we had the opportunity to gain direct insight into the production process and converse with their internal representatives, whose observations and information were invaluable for our further work. We used materials from their training center as literature for our research, comparing them with the existing curriculum of the Technical School and defining conclusions for further curriculum improvement.

Second Workshop in Italy 18.04.2024. - Technical School in Sestri Levante The second workshop in Italy took place during the LTTA in Italy, where we visited the Technical School in Sestri Levante and had the opportunity to speak with teachers and students about the knowledge and skills required for working with robots.

Other Workshops at the Fair of Technique and Technological Innovations (21-24 May) in Belgrade During this event, workshops were held with several companies:

- **VIRS doo** - A company shaping trends in automation and robotization of production processes for over 10 years, preparing comprehensive and customized technological solutions and implementing cutting-edge technologies in robotic welding and automation of production processes, using FANUC industrial robots.
- **Vertus Zemun doo** - Specializing in the distribution of equipment, tools, and consumables for metal processing, representing brands like Karnasch, Promotech, Bernardo, and others. They automate business processes mainly in welding.
- **KUKA Representation in Serbia** - We interviewed Ljuban Vasić, head of sales for KUKA in Serbia.
- **FANUC Representation in Serbia** - A global leader in industrial automation, FANUC provides high-tech products and services, offering a comprehensive range of leading products and services for industrial robotics, CNC systems, and factory automation solutions.
- **Faculty of Mechanical Engineering, University of Niš** - A workshop was held on 24.05.2024. at the Science and Technology Park Niš with Professor Žarko Čojbašić, whose primary scientific field is automation and robotics. He also works directly with companies on production process automation, bringing rich practical experience in addition to academic knowledge.

Job Descriptions - Occupational Profiles From the summarized information from all workshops, interviews, and desk analysis, we can conclude that robotics is a multidisciplinary and interdisciplinary science and technology. When talking about robotics, we generally mean industrial robotics. However, our interlocutors believe that by 2024, robotics has transcended industrial applications to include a broader range of uses (drones, humanoid robots, collaborative robots, etc.).

Currently, robots are used in almost all branches of industry, and it is nearly impossible to consider them in isolation from the machine station and production process. Depending on the industry, the complexity of the production process, and the capacity of the companies, all occupations in the domain of industrial robotics encountered in our analysis can be categorized into four types of robotics-related jobs:

- Robotics Technician (Production Operator)
- Maintenance Technician

- Systems Technical Designer - Robotics Engineer
- Production Systems Manager

In the following text, we will present job descriptions individually. Note that the job details vary depending on available information and the prevalence of the specific occupation in the analyzed companies.

JOB DESCRIPTION - Robotics Technician (Production Operator)

Robotics operators can work in a variety of industries. Depending on the specific job, robotics operators may also be responsible for troubleshooting issues that arise, performing repairs, and providing feedback to engineers about robot design. The role of a robotics technician, along with its responsibilities, is one of the most common professions in our area.

Job Profile: Robotics Technician (Production Operator)

Work activites	**Steps - tasks**
1. Preparation for work	1.1. Visual inspection of the machine site (the machine site includes a robotic arm that performs part of the process) 1.2. Defining parameters on the robot control unit as needed 1.3. Confirming the correctness of the parameters 1.4. Starting the machine in automatic mode
2. Monitoring operations and diagnosing deviations	2.1. Monitoring the operation of the robotic arm and the machine site 2.2. Receiving information from the operator regarding problems occurring in the functioning of the robotic arm 2.3. Visual inspection of the machine and the robot within it 2.4. Identifying the type of error based on the alarm or the problem notification on the panel 2.5. Diagnosing the cause of the stoppage in consultation with the engineer 2.6. If necessary, stopping the machine using the key to switch to manual mode
3. Work correction	3.1. Securing the site 3.2. Correcting the cause of the interruption by adjusting the program or physically replacing basic parts (replacing cables, motors, controllers, robot tools, sensors, lubricants)

	3.3. Returning the robotic arm to its original position using the control unit 3.4. Defining new starting positions for the robot 3.5. Confirming the correctness of the parameters 3.6. Restarting

Work activity: Preparation for work		
Skills	**Knowledge**	**Attitudes**
Able to distinguish parts of the robotic system (robot, controller, control unit, output devices, sensor-regulator-actuator) in order to perform a visual inspection of the machine site with the robot. Able to use the control unit and define parameters on it. Able to distinguish between correct and incorrect parameters. Able to start the robot based on specified parameters. Able to use the key to switch between manual and automatic modes.	Able to explain parts of the robotic system. Able to describe the function and operation of the robotic arm. Able to distinguish between different tools for the robotic arm. Has basic programming knowledge. Able to describe the procedure for defining parameters on the control unit. Familiar with the different modes of robot movement (linear, continuous, circular). Knows the rules for safe operation. Understands the function and use of the key for switching the robotic arm to a different mode. Knowledge of English for understanding instructions on the control unit (this should be included as a prerequisite requirement for training).	Responsibility, thoughtfulness, logical thinking, analytical skills, precision

Work activity: Monitoring operations and diagnosing deviations Work situation: Based on the alarm displayed on the panel, the worker identifies the malfunction either through knowledge or by using the manufacturer's guide (Manufacturer's instructions).		
Skills	**Knowledge**	**Attitudes**
Able to monitor the robot's operation Can detect irregularities in operation Understands information from the operator about the problem Understands the cause of the problem Able to perform a visual inspection of the robotic arm Able to use the key to switch the robot to manual mode Able to use the manufacturer's instructions Can diagnose the cause of the stoppage Able to use the control panel (graphical interface) Able to stop the machine if necessary	Understands basic robotic systems and their functioning Familiar with the basics of kinematics, dynamics, and the use of industrial robots Can explain the functional structure of the robot and the interconnection of structural elements as a whole Able to explain the function of the control unit Familiar with the defined operation of the robot and the machine line Knows the possible causes of robot stoppages Understands safety procedures Can describe the difference between automatic and manual modes of robot operation Understands English for comprehending instructions on the control	Analytical, precise, communicative, safety-conscious, detail-oriented, highly responsible, logical thinking, precision

	unit and manufacturer's guide Familiar with the coordinate system Knows the basics of programming	

Work activity: Work correction		
Skills	**Knowledge**	**Attitudes**
• Able to make small corrections in the program on the control unit • Can replace basic parts in the robotic cell (replacing cables, motors, tools on the robot, sensors, lubricants, controllers, etc.) • Able to return the robotic arm to its original position using the control unit • Can set new position parameters for the robotic arm • Able to create a report on the corrections made	· Familiar with basic principles: electronics, pneumatics, mechatronics, sensor operation, mechanics, servo motors, drive regulation, calibration · Knows the basics of programming · Familiar with the function of robots in flexible automation · Knows how to connect robots with other parts of the machine site · Able to stop the robot and change parameters depending on the environment · Able to operate the robot in both automatic and manual modes	Analytical, precise, communicative, safety-conscious, detail-oriented, highly responsible, logical thinking

	· Makes corrections to robot parameters based on the environment · Understands input and output parameters for system operation	

Robotics technicians can perform a wide range of tasks, from building and setting up robotic equipment to disassembling it. They can perform routine maintenance on robots and are often responsible for troubleshooting and correcting minor errors to ensure the machines operate correctly. In industries, their tasks mainly involve monitoring the operation of robots within the machine station and making small adjustments to their performance.

Our interviewees believe that a robotics technician cannot customize a robot for a specific task (program it) because this is more complex and not part of their role. However, even though robotics technicians do not engage in programming, they need to have a basic understanding of programming principles.

It is also important to note that our interviewees mentioned only closed cells in the industry when describing this job. However, today people in production work alongside collaborative robots (cobots), where robots and humans work together. Thus, the job description of a robotics technician should include working with collaborative robots.

Job Profile: Maintenance Technician

Since we did not encounter enough people involved in robot maintenance in our fieldwork and sample, the following text provides only a job description for a maintenance technician, rather than a detailed job analysis. This job description has been verified by our interviewees and can serve as a starting point for curriculum improvement.

A maintenance technician is a person who directly maintains robots and robotic systems. They maintain the electromechanical components of the robot, install software (SW) and hardware (HW) systems, and perform preventive maintenance on industrial robots and their systems. It is desirable for them to have an educational background in electrical engineering, to know and have experience in servicing, commissioning, and repairing installations, and to understand the basics of programming and automation. They need to have a good understanding of factory automation processes and functionalities.

In our practice, within factories, robotics technicians handle regular maintenance and minor repairs of robots and systems, while services and major repairs are the responsibility of the equipment manufacturers. Therefore, specific knowledge of the software used by certain manufacturers is necessary.

Job Profile: System Technical Designer - Robotics Engineer

A system technical designer - robotics engineer is generally a college-educated individual with a degree in Robotics, Computer Science, Electrical Engineering, or similar fields. A robotics engineer is capable of conducting feasibility research and designing new robotic equipment. They install and test new robotic systems to ensure proper operation, oversee technicians and robotics operators, review testing and repair data, troubleshoot robotic systems, and integrate robots with peripheral equipment.

Robotics engineers regularly research the latest trends and modern approaches in robotics (navigation, sensing, decision-making, robotic mechanism components, system functionality, etc.). They are responsible for designing, developing, testing, building, and servicing robots. The goal of their work is to construct robots that are productive and safe for various purposes. They often use computer-aided design (CAD) software and manufacturing software to create their plans.

Robotics engineers work closely with software developers to create highly sophisticated robots capable of performing specific tasks correctly. They may also communicate with market scientists to find the most cost-effective materials needed to build robots.

Communication, including active listening, is a key skill for robotics engineers. These professionals may be responsible for explaining ideas and robotic equipment to managers, some of whom may have little technical knowledge. Creativity and problem-solving skills are also advantages.

Work analysis

Work activtites	**Steps - tasks**
1. Preparation for work - design and development	1.1. Communication with clients 1.2. Understanding the needs and the industry where automation is being implemented 1.3. Conducting research on the feasibility and design of robotic equipment

2.Installing robotic systems	2.1. Constructing robotic systems 2.2. Programming robots in collaboration with software developers 2.3. Integrating robots with peripheral equipment
3. Testing and corrections of robotic systems	3.1. Worker/Robotics Operator Supervision 3.2. Software Testing 3.3. Error Identification 3.4. Troubleshooting 3.5. Commissioning

Work activity: Preparation for work - design and development of robotic systems		
Skills	**Knowledge**	**Attitudes**
Able to understand the needs of the industry for which automation is being implemented Able to combine existing knowledge and solutions with client requirements Able to communicate with clients regarding their requirements and possible solutions Able to develop solutions in a simulator Good at understanding, developing, and	Able to explain client needs Able to apply existing knowledge and solutions to meet client requirements Familiar with robots and robotic components Knows the basics of assertive communication Understands safety rules Good understanding of tools and environments for robotic simulation (IsaacSim, Gazebo, etc.)	Responsibility, thoughtfulness, logical thinking, analytical skills, precision, communicative abilities

implementing complex systems Able to distinguish parts of the robotic system	Knowledge of English	

Work activity: Installation of robotic systems		
Skills	**Knowledge**	**Attitudes**
Able to design and develop solutions for various types of robots (mobile robots, robotic arms, drones, industrial robots) Able to design robotic hardware and software according to customer orders Able to maintain system documentation Able to develop computerized systems that exist within robots Able to use computer coding and software design Able to implement developed technical solutions	Knowledge of the principles of robotics, control systems, and algorithms Familiar with coordinate systems and different types of robot movements Knowledge of programming languages used in robotics: C++, Python, ROS (Robot Operating System) Understands robotic perception, path planning, kinematics, and obstacle avoidance Understanding the principles of operation of robotic sensors and actuators, as well as control systems Demonstrating an understanding of HW/SW architecture design Familiar with SLAM (simultaneous localization and mapping) Well-versed in algorithms for path planning, obstacle avoidance,	Responsibility, thoughtfulness, logical thinking, analytical skills, precision, communicative abilities, creativity, innovation.

Independent test execution Commissioning and validation testing of hardware and software	holonomic and non-holonomic motion planning, path planning for robotic arms (MoveIt) Experienced in working with Bayesian/Kalman filters and sensor fusion (LiDAR, IMU, Visual, Odometry, Radar, GPS, etc.) Knowledge of electrical engineering and programming Knowledge of automation Proficient in computer operation Understanding of factory automation processes and functionalities Familiar with all programmable logic controls such as PLC, SCADA Understands safety rules	

Work activity: Testing and corrections of robotic systems		
Skills	**Knowledge**	**Attitudes**
Able to independently conduct testing of robotic systems in a simulator and real environment Able to initiate and perform validation testing of hardware and software Able to communicate with employees regarding necessary corrections Able to identify hardware and software errors and	· Familiar with software testing methodologies and quality assurance · Knowledgeable about robotic automation tests · Familiar with the robot manufacturer's software · Able to make necessary corrections to software and hardware · Knowledgeable about robots and robotic components · Has a deep understanding of testing, continuous integration, building, deploying, and monitoring complex systems	Responsibility, thoughtfulness, logical thinking, analytical skills, precision, communicative abilities.

resolve them according to procedures	· Familiar with the basics of assertive communication · Understands safety rules · Knowledge of English for understanding instructions on the control unit (this should be included as a prerequisite requirement for training) · Understands safety rules	

JOB DESCRIPTION - Production Systems Manager

In our analysis, both in the field and the sample we processed, a strict distinction cannot be made between the roles of robotics engineer and production systems manager. The knowledge and skills required for both positions are very similar and intertwined, with the manager placing greater emphasis on communication with clients and workers, process management, interpersonal relationships, and quality control. The production systems manager organizes and manages projects from start to finish, taking responsibility for various phases—from translating needs into functional and technical specifications to user acceptance and even initiating production.

The dynamic world of industrial robotics not only requires technical knowledge but also specific expertise in project management to ensure precise execution with minimal risks and within deadlines. For the entire process to be high quality, the production systems manager must understand the complete automation process, so the knowledge and skills are almost identical to those of a robotics engineer. In practice, robotics engineers with years of experience and strong communication skills often become process managers. Therefore, the following text does not provide a detailed job analysis but rather a description of the role of the industrial systems manager.

The production systems manager is responsible for project planning in accordance with client demands and industry requirements, managing the project to proactively mitigate risks and efficiently allocate resources. They must also manage deadlines and oversee the budget while controlling costs for optimal financial management. The person in this role must possess strong communication skills to effectively interact with all stakeholders regarding project progress. They must perform quality control and testing to ensure the highest standards are met, as well as consult with collaborators for innovative, efficient, and safe solutions.

Additionally, they manage client relationships and conduct evaluations after project implementation to ensure continuous improvement and sustainable development.

Conclusion

Robotics is a multidisciplinary and interdisciplinary science and technology. The application of robots in industry aims to increase productivity, maintain a constant level of product quality, enhance flexibility, and humanize work. Robots can be found in almost all branches of industry, and in industrial production, it is nearly impossible to consider them isolated from the machinery and production processes.

The goal of this analysis was to use a specific methodology (SKA analysis) in communication with companies and workers who directly work with robots to understand the knowledge, skills, and attitudes required by the industry from employees. Based on these insights, partners from the Technical School can adapt the existing robotics curriculum to meet the industry demands. The SKA methodology (Skills, Knowledge, Attitude method), developed by SFIVET (The Swiss Federal Institute for Vocational Education and Training), serves as a method for profiling occupations and has been tailored to the specificities of this project. Besides desk research, the analysis also included field research in Serbia and Italy.

The analysis was conducted through workshops with direct executors of jobs and superiors involved in employee education within companies. Companies from Serbia and Italy were included (from Serbia: Tigar Tyres from Pirot, TERI ENGINEERING DOO BEOGRAD - BRANCH TERI ENGINEERING PIROT, VIRS doo, Vertus Zemun doo, as well as KUKA and FANUC representatives in Serbia. In Italy, we visited the company Comau and the Technical School in Sestri Levante, which includes robotics in its program). Additionally, we conducted interviews with Professor Žarko Čojbašić from the Faculty of Mechanical Engineering, University of Niš, whose primary scientific field is automation and robotics, and representatives from the University of Kragujevac, developing an innovative robotic platform for waste sorting. We also utilized desk analysis of existing curricula from secondary vocational schools that study robotics and automation, as well as the qualification standards in robotics in the Republic of Serbia.

This analysis identified the areas of competence required for successfully performing jobs in robotics, i.e., the skills, knowledge, and attitudes individuals need to be successful in robotics-related jobs. The results of the SKA analysis provided input for developing and supplementing the Robotics curriculum, particularly in the domain of practical knowledge.

Depending on the industry, the complexity of the production process, and the capacity of companies, four types of robotics-related occupations were identified:

- Production Operator (Robotics Technician)
- Maintenance Technician
- Systems Technical Designer
- Production Systems Manager

Different knowledge and skills are required for workers depending on the job position and production complexity. In our research, we most frequently encountered the Robotics Technician and Systems Technical Designer - Robotics Engineer.

The Robotics Technician, with their tasks, is the most common occupation in our region. The Robotics Technician cannot adapt the robot for a specific task (program it) as this is more complex and done by the Robotics Engineer. However, even though the Robotics Technician does not engage in programming, they should master some basic level and simple tasks.

The analysis also included the description of the Maintenance Technician's job, which cannot be clearly separated from the Robotics Technician in the field and possesses very specific knowledge at different levels depending on the industry. The Maintenance Technician focuses on the mechanical parts of the robot and machinery and their maintenance.

The competencies of the Production Systems Manager are generally similar to those of the Robotics Engineer, so a detailed job analysis, which would be very similar to that of the Robotics Engineer, is not provided.

The presented job analyses and descriptions should serve as input parameters for determining the robotics curriculum.

Detailed analyses and defined knowledge, skills, and attitudes for the Robotics Technician and Robotics Engineer, with recommendations for module outcomes and content, are provided in the analysis. In addition to the knowledge, skills, and attitudes presented, the following should be considered:

- Differentiate between robotics and industrial robotics. It is considered that what we today call robotics and consider as such is actually industrial robotics, as we want students, primarily technically oriented, to be trained for industrial robotics. However, by 2024, robotics has surpassed this (drones, humanoid robots, cobots - collaborative robots).
- The recommendation is that the technical school, in addition to industrial robotics, should also address a broader picture of contemporary robotics in the curriculum, which can be more relatable and interesting to students (robotic vacuum cleaners, drones, autonomous vehicles, etc.).
- The analysis participants believe that the practical part should focus on and be conducted with industrial robots and programming industrial robots, focusing on programming the robot that the school possesses.
- The issue of programming languages and programming is very complex, according to the analysis participants, as everything is changing rapidly, and the use of artificial intelligence in programming has significantly altered things. Knowing the basics of programming, with an emphasis on the programming language of the robotic arm manufacturer that the school possesses, is very important.
- LEGO kits are also very good for practicing programming.

- The recommendation from the participants is that when designing the curriculum, basic kinematics and dynamics of robots should be covered, including industrial robot parts, but also considering cobots with different safety systems and cells.
- The recommendation is that programming should be considered in the domain of industrial robotics and that practical exercises should focus on tasks such as pick and place, palletizing, tracking a given path, picking and sorting specific elements based on characteristics and using sensors, writing names using a robotic arm, welding, painting, etc.
- The recommendation is to acquire, for example, a robotic vacuum cleaner and practice navigation, memory, room planning through practical exercises.
- Participants note that there are no factories in our environment that develop robotic solutions; rather, robots are purchased and production problems are solved in some robotic cells. Therefore, they believe that robot design does not make much sense to be specifically addressed in the curriculum.
- On the other hand, it is necessary to emphasize the application of existing robots to students, as part of the curriculum - the application of industrial robots.
- It is necessary to use other programming systems (LEGO, 3D printing).
- In addition to industrial robotics, it is highly recommended to consider other segments of robotics that are also advancing (humanoid robots), distribution centers using drones - managing drones in warehouses, responding to problems, recharging batteries, coordinating drones; autonomous vehicles - recharging cars, coordinating, servicing, changing batteries, tracking; precision agriculture - autonomous tractors; medical - maintaining hospital beds automatically; hygiene maintenance - autonomous vacuum cleaners and cleaners - servicing them, setting, maintaining, recharging batteries, changing batteries, etc.
- Since robotics will continue to develop and future occupations cannot be predicted, participants believe that it is good to provide students with fundamentals that they can build upon.
- Chapters can be added on the application of robots in transport technology, medicine, agriculture, providing practical tasks if possible (a broader view of robotics - other areas of application, not just industrial robotics).

Regarding the results of our SKA analysis, they were presented to the professor of the Faculty of Mechanical Engineering, who agreed with the analysis and provided active recommendations for curriculum development included in the above recommendations. It is also recommended to consider the existing curricula of other schools in Serbia.

By comparing the existing curriculum of the Technical School Pirot with the findings of the analysis and curricula of schools in Italy, we believe that the existing curriculum should be improved, especially in the area of students' practical knowledge to ensure they can apply the acquired theoretical knowledge in practice. Participants in the analysis suggested four modules:

2. Introduction to Robotics
3. Kinematics and Dynamics of Robots
4. Robot Control and Programming

5. Robotics Application

The first module, "Introduction to Robotics," and the second module, "Kinematics and Dynamics of Robots," are mainly focused on theoretical foundations, and the topics are generally covered by the existing curriculum. The emphasis of this analysis and recommendations for curriculum improvement is on the third module (Robot Control and Programming) and the fourth module (Mechanical Systems with Robotic Arms) and their practical implementation. This involves more hours of practical work in the proposed topics within modules 2, 3, and 4.

To conduct this type of teaching, the school needs resources, which include trained staff and robotic arms from various manufacturers commonly found in the industry. It is also desirable to develop learning materials (manuals) that detail the process of conducting practical exercises and are used by students and teachers in teaching.

CHAPTER 1

Welcome at the beginning of this chapter, devoted to understanding the basic concepts about robotics, industrial robotics, and the role that both robots and humans/users have in the industrial sector.

Introduction to robotics

Before defining and presenting industrial robots, we consider strictly relevant to introduce – more generally – what a robot is and commenting the role of robots in the education system.

Giving a formal and unique definition of robot is not that simple. Many different robots exist, and this broad class of devices makes it difficult to give a clear definition of it. The term "robot" means different things to different people. Even roboticists themselves have different notions about what is or isn't a robot. (Guizzo, 2018).

As an example, Merriam-Webster (n.d.) defines it as a "machine that resembles a living creature in being capable of moving independently … and performing complex actions" or "a device that automatically performs complicated, often repetitive tasks", so "a mechanism guided by automatic controls".
Even if not of these definitions are exhaustive – for example, not all robots resemble living creatures, or what a complicated task is? -, they all share the concept that a robot is a machine, that may perform tasks.
Another definition that may be found is "A robot is a machine which is programmed to move and perform certain tasks automatically" (HarperCollins, n.d.). This definition adds to the previous one the concept of movements, and – which is a point of main importance – the concept of the ability to do something in an autonomous way.

Further, The Oxford English Dictionary gives the following definition: "A machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer.".

Ben-Ari and Mondada (2018) show that automation in carrying out activities is a key element in robotics. But, about the concept of complex actions they write "The difference between a robot and a simple automaton like a dishwasher is in the definition of what a "complex series of actions" is … there are machines that are at the boundary between automata and robots.". Further, they focus the attention on the concept of programmable by a computer and writes that it is another key element in robotics. They furthermore note that few definitions of robots include the concept of sensors, but robots – instead of simpler automata – have it and use it to adapt their actions to the environment.

Putting all these elements together, we can move near to the definition given by Guizzo (2018): "A robot is an autonomous machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.".

In this manual, then, we will assume that a robot is a computer programmable machine, capable of acting in the real world, someway autonomously, because of the sensing of the environment.

Educational Robotics is then a branch of education, which uses robots as didactic tools to support the teaching/learning approach, and some robots – such as LEGO EV3, LEGO WeDo, Bee-Bot, Cubetto, Thimyio, … - has been explicitly designed to be used as teaching instruments (EARLY, n.d.).

Sant'Anna School of Advanced Studies writes "Educational Robotics is an integrated system using robotics and technology as a method to increase the quality of scientific and technical education in Public School … it increases involvement, develops problem-solving skills, promotes an interdisciplinary approach and supports the development of teamwork". Angel-Fernandez and Vincze (2018) note how educational robotics may involve several uses of robots, like a direct object to be used, a tool to support learning, or as a learning aid. Then they try to give a definition of educational robotics independently from the role of the robot, and they suggest the following: "Educational Robotics is a field of study that aims to improve the learning experience of people through the creation, implementation, improvement, and validation of pedagogical activities, tools (e.g. guidelines and templates) and technologies, where robots play an active role and pedagogical methods inform each decision."

The main theories behind Educational Robotics are constructivism and constructionism. Piaget (1974) discusses the importance of manipulating objects and artifacts, expressing how it is a key element in children's learning. Papert (1980) added the idea that knowledge construction happens especially effectively in a context where the learner is consciously engaged in constructing a public entity. This entity could be any physical object, such as a sandcastle, or a bricks construction, but in this field, we will refer to technological and robotics artifacts.

The e-Media project Consortium (2019) writes that children learn quickly and easily if they can use concrete and physical objects, like robots – 3D devices – are. Further, robots act as facilitators for learning and can evolve into a rich experience for children, who will take different skills and competencies from different subjects.

According to Alimisis (2013), educational robotics can really improve the learning approach, but the solution is not to only introduce robots in didactics. He notes how learning robotics is not the goal to achieve in educational robotics. Instead, a didactic curriculum update, to include robots as tools, must be made. He writes "the role of Educational Robotics should be seen as a tool to foster essential life skills … through which people can develop their potential to use their imagination, to express themselves and make original and valued choices in their lives. Robotics benefits are relevant for all children …".

In this manual we will provide a technical vision about the robots, since the goal of the document is to foster industrial robotics skills. However, keep in mind that robots can also be used as teaching tools, to support the education process, and should be intended not as the goal to reach, but as an instrument to obtain other skills.

Introduction to industrial robotics

The manufacturing landscape has undergone a significant revolution with the integration of industrial robots. These machines are no longer confined to pure science fiction; they have revolutionized industrial production lines, performing tasks with matchless precision, speed and endurance. Indeed, industrial robots are designed to perform tasks quickly, accurately and efficiently, often outperforming humans in terms of speed, precision and consistency.

Industrial robots are sophisticated machines that are programmable and capable of performing various tasks with precision and autonomy within industrial environments. These robots are equipped with sensors, actuators, and controllers, enabling them to perceive and interact with their surroundings. They play a crucial role in modern manufacturing processes, offering efficiency, flexibility, and reliability.

According to the definitions given by International Organization for Standardization (ISO) in 2021, industrial robots are automated and programmable manipulators. This means that they are mechanism consisting of an arrangement of segments, jointed or sliding relative to one another. They consider a manipulator as an industrial robot if it can move on three or more axes/directions, and if it is used in an industrial environment. The manipulators can be multipurpose, due to the use of different end effectors.

Definition and characteristics of a robotic arm

A robotic arm, a subset of industrial robots, is a type of mechanical arm that is similar to a human arm. It has various segments which closely resemble the shoulder, elbow, and wrist. A robotic arm is programmable and can be directed to perform a variety of functions just like the human arm can (Flynt).

This design allows robotic arms to perform a wide range of tasks, including:

- Material handling: Precisely picking, placing, and transporting objects of various shapes and sizes.
- Assembly: Meticulously assembling complex components with consistent accuracy.
- Welding: Performing intricate welds with exceptional control and repeatability.
- Painting: Applying paint finishes with a smooth, even coat, eliminating human error
- Machine tending: Loading and unloading machines, ensuring continuous operation.

These are just a few examples, and the potential applications of robotic arms continue to expand as technology advances.

Robotic arms are used in various industries and applications where precision, repeatability, and efficiency are paramount. They are commonly deployed in manufacturing processes such as automotive assembly, electronics production, and warehousing logistics. Robotic arms are preferred for tasks that require high accuracy, consistency, and the ability to operate in hazardous or challenging environments. Additionally, they enhance productivity by reducing cycle times and labor costs while improving overall quality and safety (Wegener et al, 2015; Meir et al, 2024).

There are several types of industrial robots, each designed for specific tasks (Universal Robots, 2022; Internetional Federation of Robotics, 2022), for example:

- Articulated Robots: They look like a human arm, which is why usually we refer to Articulated Robots just calling them "robotic arms" or "manupulators".
- Cartesian Robots: Also called rectilinear or gantry robots, they have three prismatic joints for the movement of the tool and three rotary joints for its orientation in space.
- SCARA Robots: These manipulators have two parallel rotary joints to provide compliance in a selected plane.
- Delta Robots: These manipulators have links which form a closed loop structure.
- Collaborative Robots (Cobots): These are designed to work near humans. Certain safety features allow for significant risk reduction in hybrid work environments.

Inside this manual we will refer to industrial robots or manipulators considering articulated robots, so arms capable of moving someway like human arms.

Digressions about industrial robots outside industrial environment

We introduced, and we will discuss, about industrial robotic arms. However, robots are widely used also outside of industrial environment, and even classic robotic arm are suitable for that purpose.

Developments over the past decades in the fields of Automation and Control, Mechatronics, Machine learning and AI, vision systems, sensor technology and computational power and data sets have made it possible to realise robotic arms that, having emerged from isolation and industrial segregation, can collaborate with humans without restriction. Thus, in addition to industrial uses, we will find robotic arms employed in various tasks and even as limbs of humanoid robots.

The manipulation, precision and speed capabilities of a robotic arm are used today in tasks that were unthinkable years ago. There are cooperative robotic arms that can assist the human surgeon in delicate nano-surgery operations, equipped, as they are, not only with higher degrees of freedom than the surgeon, but also with information from Artificial Intelligence: there are exoskeletons equipped with robotic arms that allow the human to hold and move heavy weights and perform other functions easily.

The need to have a so-called "third arm" is one of mankind's dreams: we humans who, by putting on an exoskeleton equipped with several arms, can perform various object functions at the same time, like Doctor Octopus from Spider-Man. More realistically, let us imagine a surgeon engaged in a delicate operation requiring his expertise and steady hands. While his two biological hands manipulate the surgical instruments, a third robotic limb attached to his torso plays a supporting role. Or, in another field, a construction worker employs a robotic arm to support a heavy beam. And what about music written for a pianist who has twelve fingers for his piano.

The concrete need for user-friendly collaborative robotic arms relates to the fact that human life spans have lengthened worldwide. According to UN figures, in 2020 about 13.5 per cent of the world population was at least 60 years old and, according to some estimates, this figure could rise to almost 22 per cent by 2050. Advanced age can lead to cognitive and/or physical difficulties, and as more and more elderly people potentially need assistance, advances in technology can provide the necessary help. Mostly, these arms are controlled by joysticks or are relatively autonomous in the sense that they have been trained to perform several functions with simple commands. There are also advanced projects involving the brain-controlled robotic arms of people with tetraplegia, who are often paralysed from the neck down. In these projects, volunteers with tetraplegia have accepted an implant in their brain to control the arm with their thoughts, issuing mental commands that cause a robotic arm to lift a drink to their lips or help with other simple activities of daily living. These systems fall into the category of brain-machine interfaces (BMI).

For more general uses, there are now commercially available lightweight robotic arms that can be mounted on wheelchairs, tables and assist people for feeding, personal hygiene, medication management, personal safety and so on.

Another field where knowledge on intelligent robotic manipulation has been employed is the development of prosthetic upper limbs. In all the cases mentioned, the challenge for is the effective control of the many possible degrees of freedom that, in robotic arms, is greater than in human arms. The robotic art must integrate with human possibilities and capabilities, not the other way around.

Some use cases involve one or more robotic arms being mounted on a torso to simulate a humanoid robot. Among these, a recent project is a collaboration between researchers at Spain Carlos III University and the manufacturer Robotnik. The team developed the Autonomous Domestic Ambidextrous Manipulator (ADAM), a robot for the elderly that can assist people with basic daily functions. ADAM is a mobile indoor robot that is equipped with a vision system and two arms with grippers. It can adapt to homes of different sizes to ensure safe and optimal performance. It respects the user personal space, helping them perform household tasks and learning from their experiences through an imitation learning method.

ADAM's role is to perform tasks within the kitchen, preparing and delivering food or water to Robwell, who would then supply it to the users.

Manipulating objects that constantly change density and shape is among the most anticipated, ambitious and difficult challenges in robotics; these have been studied and realised in a unique project developed by the team of Bruno Siciliano, head of the Department of Electrical Engineering and Information Technologies at the Federico II

University in Naples. RoDyMan, an advanced dynamic robotic handling project, is a service robot that is able to replicate human activities with a high level of dexterity and mobility. Among its tasks is one of the most ambitious in the field of robotics, non-prehensile manipulation, manipulating deformable objects. And what could be more deformable than pizza dough? A unique exercise in dexterity and skill, an excellent training ground for future applications in the medical-surgical field, in particular for prostheses in orthopaedics, textiles and others. RoDyMan has been instructed by one of the most famous pizza makers in Naples on the dough grip, the precision grip in the vaulting phase in which one has to grasp and release the disc of dough while also assessing its consistency, the movements between palm and thumb, the bimanuality.

Robotic arms can also be used for recreational and entertainment purposes. Kuka has made a Rollercoaster with one of its large arms as a rotor. You can find a demo of it here:
https://youtu.be/ck8y1sl97BY

Also from the University of Naples Federico, with RodyMan, comes the use of a robot equipped with two cooperative robotic arms that conducted the Instrumental Ensemble of the Giuseppe Martucci Conservatory, complete with batons and score (https://youtu.be/38RIn-1gPgQ).

In conclusion, robotic arms have profoundly changed work in sectors such as automotive, logistics, and packaging and, coming out of factories, they are demonstrating a versatility that makes them one of the mainstays of collaborative robotics, and may be found even outside the industry. In this manual we will focus – both for ease of use, logical and historical reasons – to classical industrial tasks, but keep in mind that the same industrial programming may foun application outside the industries.

Configuration of a robotic arm system

A robotic arm system consists of several parts, interlinked with each other. We are going to present some of the necessary parts:

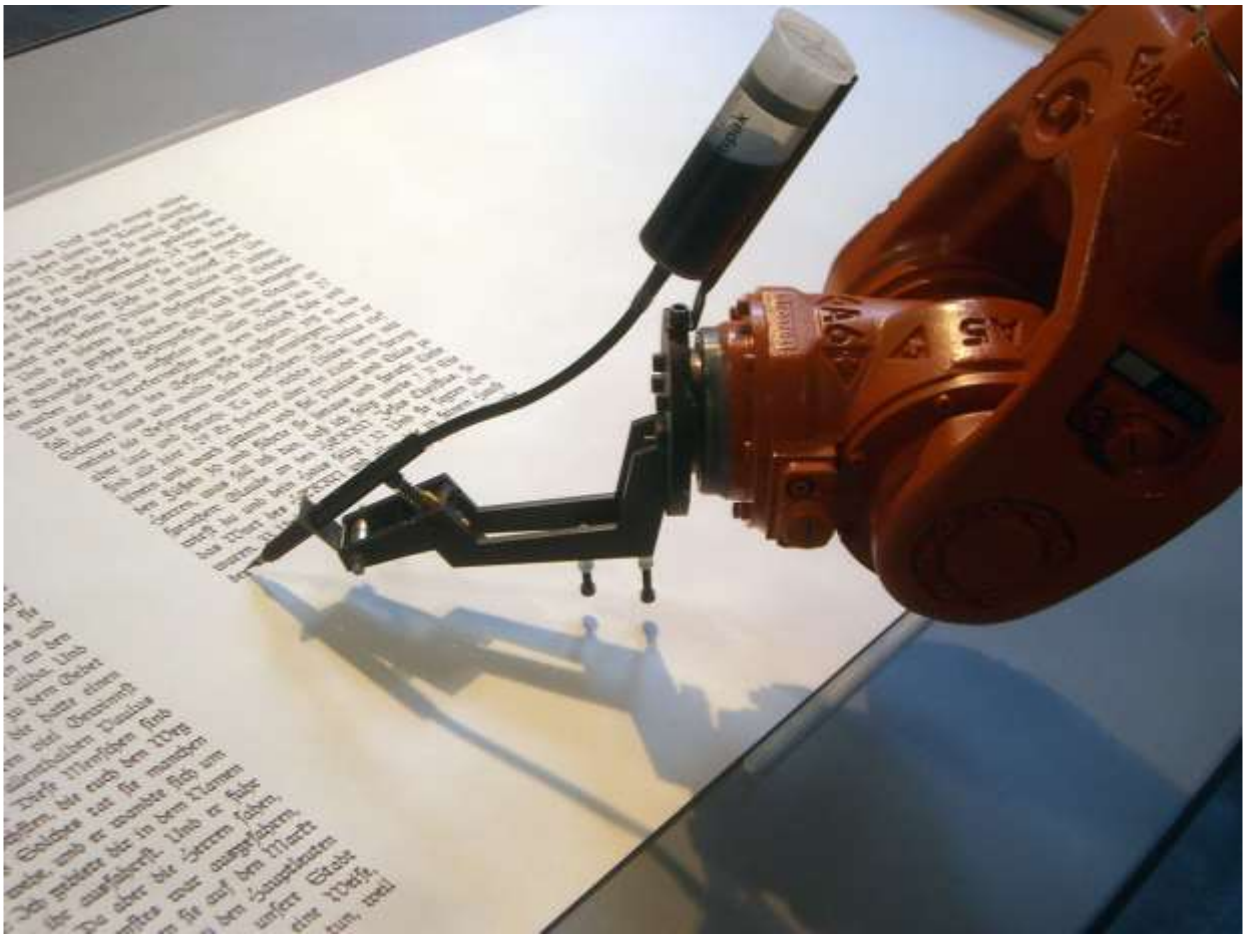
- The robot: obviously it is necessary to consider the robot itself, defined as in the previous section, as a set of linked mechanical segments and joints.
- Robot Controller/cabinets: The robot itself must be programmed, and it needs a control system. Thus, robot controller is the “brain” of a robot. It is a sophisticated piece of hardware, often accompanied by complex software, that governs the robot’s actions. It is a computer system that connects to the robot to control the movements of the industrial robot arm.

In the previous image ((c) Dependability, released under the Creative Commons Attribution-Share Alike 4.0 International license) you can see robots from the electrotechnical company ASEA (from 1988 ABB) exhibited in the ASEA Historical Collections in Västerås, Sweden. From left: IRB 6, IRB 2000, ABB IRB 3000. At the extreme right of the image, you can see the ABB Control cabinet S3 (for IRB 2000 and 3000).

- Robot Teach Pendant: Usually industrial manipulators include teach pendants, handheld devices used to program industrial robots3. It may contain several buttons or switches or feature a touchscreen display3. It allows the owner to make any changes or additions to the programming function of robot.

Image courtesy of KUKA AG, that shows a robot controller in background, and a teach pendant.

- End Effector and flange: The same Industrial robots may perform different tasks, mounting different tools or end effector. Industrial robots usually have a flange. It is the interface where tools, grippers, or other end-effectors are attached for performing specific tasks. The flange typically provides mechanical connections, such as mounting holes or a quick-change mechanism, as well as electrical and pneumatic connections for power and control signals. An end effector is a device attached to the end of a robot's arm to help it interact with the surrounding environment. It is a peripheral device that attaches to a robot's wrist, allowing the robot to interact with its task.



In the previous image (Mirko Tobias Schaefer, licensed under the Creative Commons Attribution 2.0 Generic license) you can see an industrial robot with an ink pen mounted as end effector on the flange.

- Peripheral Devices: Peripheral devices are non-essential devices or equipment connected to the robot controller, in order to extend its capabilities. In the context of robotics, peripheral equipment refers to all types of equipment that provide functions the robot is missing or execute functions more cost-effectively.

- Workpiece: Even if it is not strictly related to the robot, it is mandatory to consider the existence of the workpiece, that is the object manipulated by the robot. The workpiece may be external respect to the robot, and the tools mounted on the robot may work on it (or carry it, if we are talking about grippers), but does also exist option where the workpiece is connected to the flange, and the tools are mounted externally.

CHAPTER 2

In this second chapter we will introduce the basics of a theoretical framework, necessary to fully understand and manage the ideas proposed in the next sections. This framework may also be used as a reference, to wonder how industrial robotics may be used to discuss standard didactics topics in school lessons using an industrial robot.

Robotic field

A widly used term inside industrial robotics is "the field". It refers to the environment where robots and automated systems operate. This encompasses the physical space, machinery, and processes involved in industrial activities such as manufacturing and assembly. Understanding the field is essential for optimizing robotic operations, ensuring safety, and achieving efficiency and productivity.

The field in industrial robotics includes the physical environment, which consists of the factory layout, lighting, temperature, and potential contaminants. It also involves the machinery and equipment that interact with or are operated by robots, such as conveyor belts and assembly lines. Additionally, it includes the specific industrial processes and workflows that the robots are integrated into, such as welding, painting, or material handling.

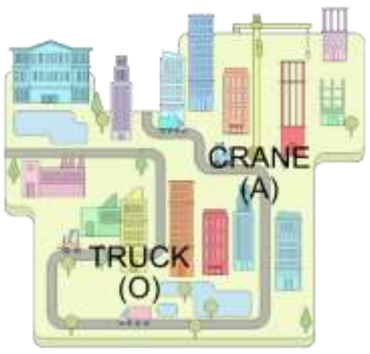
The field is crucial because it directly impacts the performance and efficiency of robotic systems. A well-designed field ensures smooth operations, minimizes downtime, and enhances safety for both robots and human workers. Proper understanding and management of the field lead to better integration of robotics into industrial processes, ultimately improving productivity and product quality.

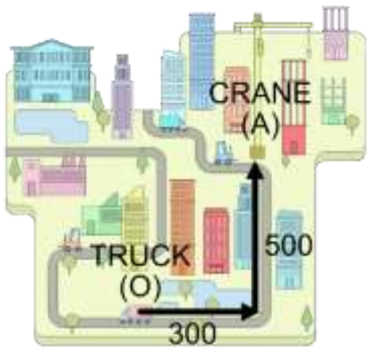
During the next chapters we will focus on a few elements from the field, and we will discuss key points in field robotics.

Coordinates

In several moments connected to the use of a robot it is important to define where an element is located in space.

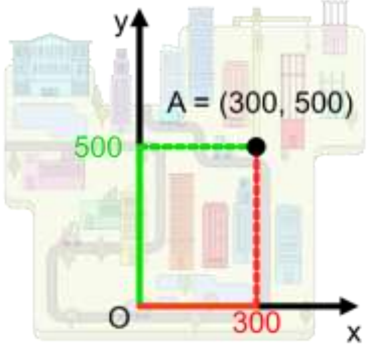
Let's start thinking it in two dimensions. Suppose we are driving a truck in a city, and call its starting point O . Suppose we would define the location of a destination point, for example a crane, that we will call A. We just defined two points, that locate an element on the map. We may now state that the point A is located, for example, 300 meters east and 500 meters north respect to us.





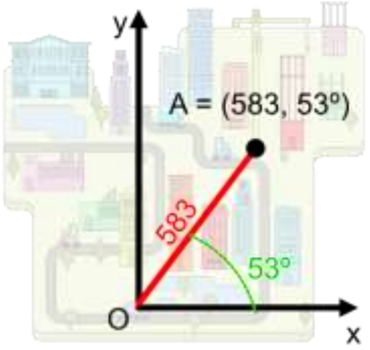
So, what did we just do? We defined the position of the point A, giving relative information related to a defined origin point (that is why we called the truck point 'O'). Talking more scientifically - and according to Encyclopedia of Mathematics, anyone can define a Cartesian Coordinate System.

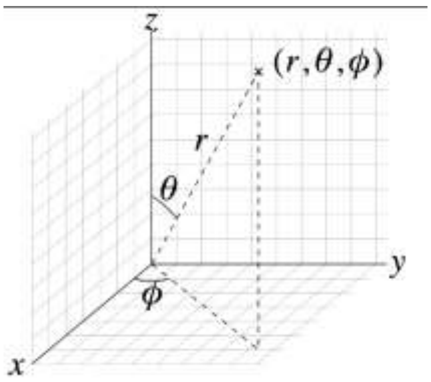
This is a combination of two orthogonal lines, called coordinate axes, on each of which a positive direction has been chosen and a segment of unit length has been specified. The point of intersection of the coordinate axes is said to be the origin of the coordinate system. Once this has been defined, it is possible to define any point in the plane giving its coordinates, that is the distance along the defined directions from the origin.



A point may be defined as a couple of ordered numbers (x, y), representing the distance. The first number (the abscissa) is equal to the magnitude of the orthogonal projection of the directed segment OA on the abscissa axis, the second one (the ordinate) being the orthogonal projection of the directed segment OA on the ordinate axis. The same approach may be used in three dimensions, adding a third direction and the third coordinate 'z'.

A different approach to define a second point starting from an origin is called polar coordinates, in this way you need only to define the distance from the origin to the destination, looking at the direction, but then you need to define which direction is that. Usually, the polar coordinates are defined as $A = (\rho, \theta)$ where ρ is the distance from the origin to the point, and θ is the angle between the x axys and the direction from the origin to the desired point. The same approach may be used, again, in 3D adding a second angle (usually it is φ) and sometimes calling the system as "spherical" instead of polar.





Vectors

As seen before physics – and so robotics - deals with a great many quantities that have both size and direction, and it needs a special mathematical language, using vectors.

Some quantity may be expressed using only one number, for example the temperature in a room or a mass (i.e. 23 ºC, or 150 kg). This is called 'scalar'. On the other hand, does exist quantities that need to be expressed in a different form, for example movement, speed, acceleration. Imagine trying to define the speed of an airplane. You may say that it is

$$v = 1000\frac{\text{km}}{\text{h}} ,$$

but this is only the magnitude of the speed, since you also need to define the direction of the movement. (Walker, 2014).

Vectors are typically represented by arrows, where the length of the arrow indicates the magnitude and the arrowhead points in the direction. Mathematically, vectors are often expressed in coordinate systems using components, such as $\vec{v} = (v_x,\ v_y,\ v_z)$ in a three-dimensional cartesian coordinate system. These components represent the projection of the vector onto the x , y , and z axes, respectively. They indicate how much of the vector's magnitude is directed along each of these axes. For example, if $\vec{v}$, its components would represent the velocity in the x , y , and z directions.

Position, speed, acceleration

We noticed in the previous section that we can define the position of a point in space using its coordinates. One general way of locating a particle (or particle-like object) is with a position vector often called $\vec{r}$, which is a vector that extends from a reference point (usually the origin) to the desired point: $\vec{r} = (x,\ y, z)$. Suppose now to observe a moving particle. Its position vector changes so that the vector always extends to the particle from the origin.

The speed of the movement is defined as the variation of space within the time, so how fast the displacement occurs:

$$\overrightarrow{v_{avg}} = \frac{\Delta\vec{r}}{\Delta t}.$$

We used the subscript 'avg' to note that we are talking about mean speed, if the Δt is finite. The same way we can define a mean acceleration, so how fast the speed changes over time:

$$\overrightarrow{a_{avg}} = \frac{\Delta\vec{r}}{\Delta t}.$$

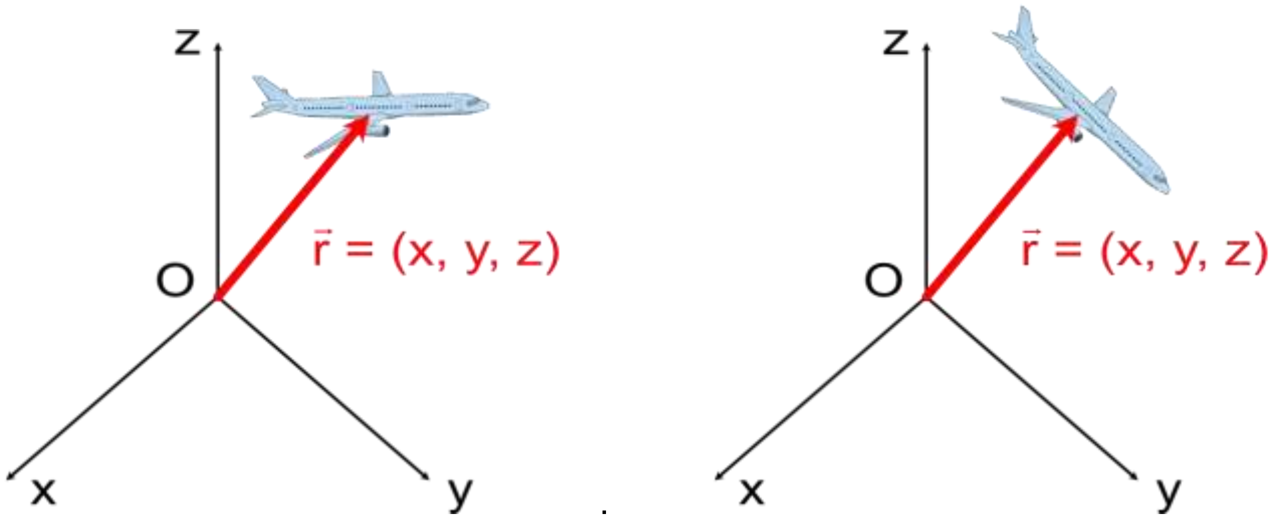
Note that if Δt is approaches zero we are considering the instantaneous speed (or acceleration), and we can use the mathematical tool of derivatives to formally manage it (Walker, 2014).

However, to our purposes, we do not need this kind of formalism, and it is enough to remember that speed is variation of position, and acceleration is variation of speed. Anyway, keep in mind that we are considering vector quantity, so for example if we wonder about a point moving horizontally at a certain speed, that suddenly changes direction (but keeping the same magnitude) we have a change of speed, since the direction has changed!

Rigid body

Since now we only discussed about points and moving points. However, real life objects are not points. In some cases, we can schematize them as points, but sometimes it is necessary consider the whole shape of the body. Another extremely important theorization, then, is the rigid body approach. A rigid body is a solid body, that cannot be deformed.

The position of a rigid body may be defined using the vector approach, usually considering the centre of mass of the object as position point. This solution, however, lead to a problem: you may define where an object is, but you are not evaluating the orientation of the object.



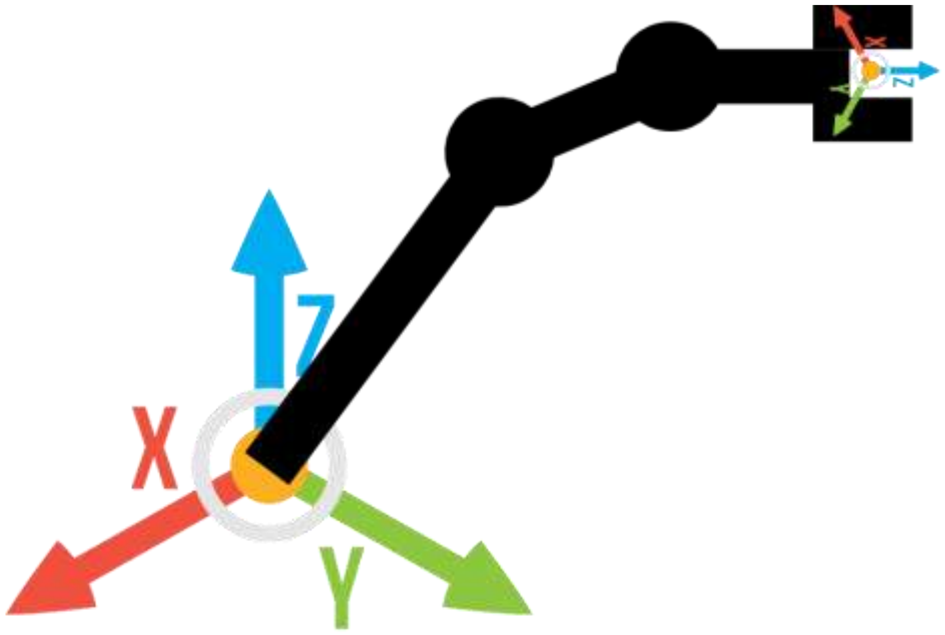
To define the orientation of the object does exist different approaches, we will later introduce the Euler Angles idea that is widely used in the robotics sector.

A manipulator can be schematically represented as a group of rigid bodies (links) connected by joints. This combination in called kinematic chain. One end of the chain is constrained to a base, while an end-effector is mounted to the other end. The resulting motion of the structure is obtained by composition of the elementary motions of each link respect to the previous one (Siciliano et al, 2008).

Euler angles

As seen in the previous section, if you move a solid body in space, to define unambiguously its pose it is not enough to define its coordinates, but it is important to determine the orientation of the piece.

One approach is to consider a second cartesian coordinate system, that is integral with the piece, and then define the orientation of the second system, over the reference frame.



Otherwise, Euler Angles may be introduced. Euler angles are a set of three angles that describe the orientation of a rigid body in three-dimensional space relative to a fixed coordinate system. They are typically used to represent rotations by sequentially applying rotations about three distinct axes, and are defined by specifying three successive rotations around fixed axes. These rotations can be applied in any order, but conventionally they are represented as rotations about three orthogonal axes.

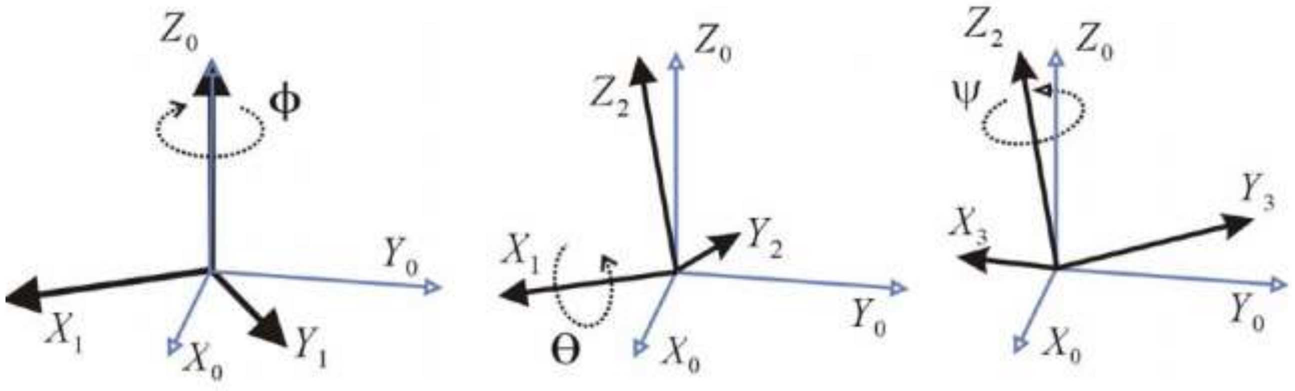


Image by Prtn mmk, released under the Creative Commons Attribution-Share Alike 3.0 Unported license. It shows how three Euler Angles (here called Φ, Θ, Ψ) can be defined.

Several (27) possible combinations of rotations exist to define Euler Angles. As example, we can consider the XYZ rotation to define Euler Angles:

- Rotate the reference frame by the angle φ around x axis;
- Then rotate the current frame (previously rotated) by the angle θ about 'new' axis y (called y')
- Finally rotate the current frame by the angle Ψ about axis z''

(Siciliano et al, 2008).

Right hand rule

The right-hand rule is a convention and a mnemonic, utilized to define the orientation of axes in three-dimensional space. A Cartesian coordinate System follow the right-hand rule. To use this rule, you should assign to your right thumb-index-middle finger three different axis, and point them mutually orthogonally. Below a description of the method, and an explanatory image.

The thumb represents the X axis, point your thumb in the direction of the positive x-axis. Then the index finger – that represent the Y axis – should be pointed - perpendicular to your thumb - in the direction of the positive y-axis. At the end if you move your middle finger in a position that is perpendicular both to thumb and index you will obtain wich is the Z positive direction. By following this rule, you establish a consistent orientation for the axes. This convention ensures uniformity in describing positions and directions in three-dimensional space.

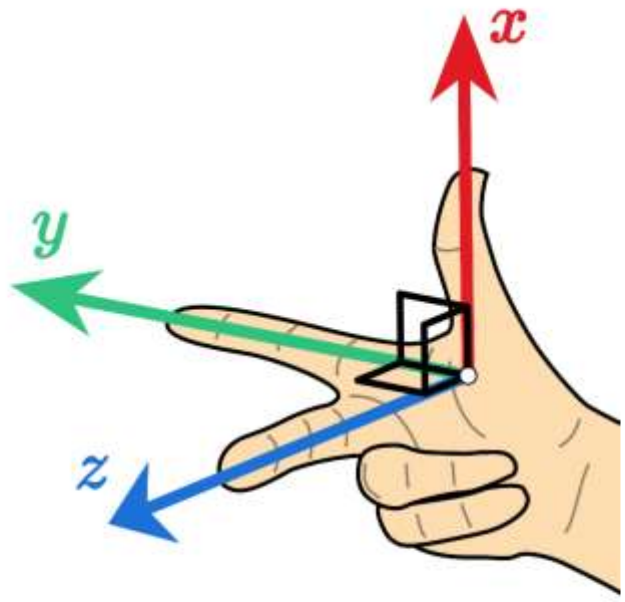


Image from Acdx, cmglee - released under the Creative Commons Attribution-Share Alike 4.0 International license.

The typical axis orientation (in the base frame, so referencing to the base of the robot), looking in front at the robot is:

- Y axis pointing at you
- X axis pointing to the left
- Z axis pointing upward

Degrees of freedom

In robotics, the degree of freedom (DoF) refers to the number of independent movements a robot arm can make. Each degree of freedom corresponds to a single axis of motion, so to and indipendent actuator. A typical robotic arm can have several degrees of freedom, allowing it to move in various ways.

An anthropomorphic robotic arm usually uses rotational movements, to moves. Classic robotic arms may have 6 or 5 DoF, however different DoF may exist.

Sometimes an arm has auxiliary movements (such as a translation along an axis trough a belt or a linear actuator), and we refer to this as auxiliary axis.

Direct and inverse kinematics

Suppose to have a 6 axis robot, so a robot with 6 different moving joints. In this case you may know the exact position of every independent joint. In this case you know 6 different values, related to the joint positions. So, you know the information

$$q = (q_1, q_2, q_3, q_4, q_5, q_6)$$

define the pose of the robot in space, if q_n are the position of the joint number n. This approach is called working in the "joint space". Even if it is a used approach, it is pretty hard to imagine where the end-effector of the robot will land just knowing the position of the joints. It is possible to move to a different "space", called cartesian space or task space, where the position of the end effector is given using cartesian coordinates and orientation.

$r = (x, y, z, \Phi, \Theta, \Psi)$

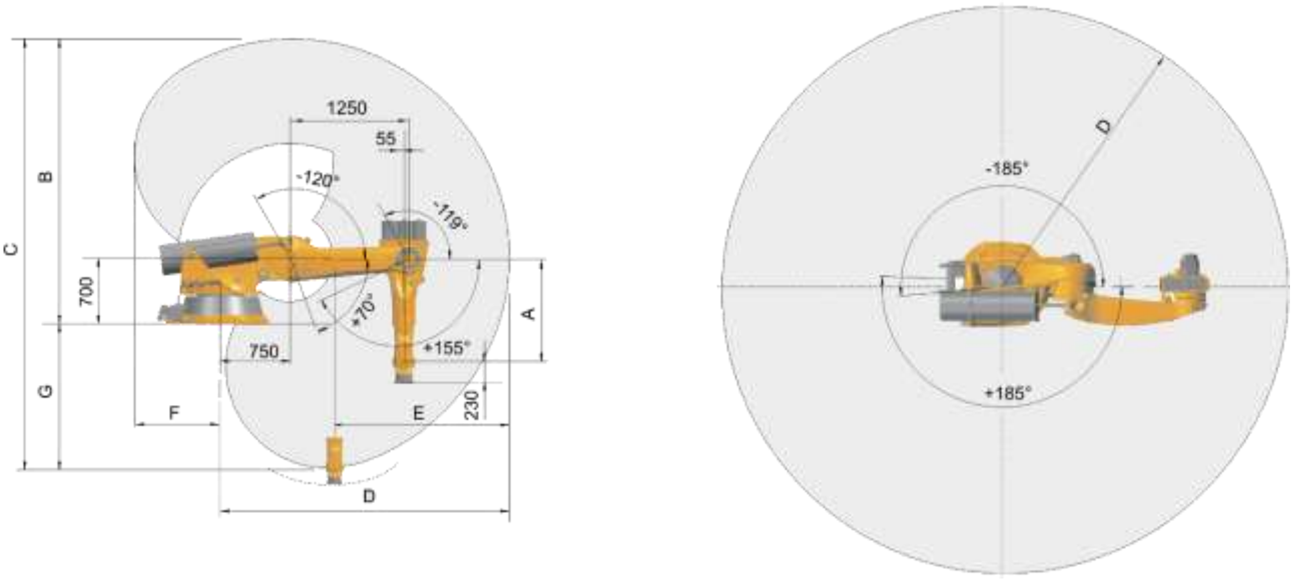
The mathematical approach to calculate and move from the joint space to the cartesian one is called "direct kinematics", while sometimes may be necessary to do de opposite, using the "inverse kinematic" (De Luca, nd).

The solution of the kinematic chain is complex, and out of the purpose of the manual, anyway it is important to understand that in robotic arms we have a set of solid bodies interconnected trough joints, and that if at the end of the last one is connected a tool, it is possible to calculate the final position of the tool knowing the physical parameters of the robot (i.e. length of each link) and orientation of them (given by the position of every joint). However, if you are only a robot user, don't be frightened by this theory, usually these calculations - which are also quite onerous - are automatically performed by robot software.

Reachable Points and singularities

Using a robotic arm is fundamental to understand which are the gettable point into space, since the robotic arm cannot move in every location in space. First of all, it should be considered the physical shape and construction of the robot. For example, it obvious that it is impossible to move the tool of the robot further than the length of the robotic links itself. Another evident problem is related to the mechanical end of stroke, or the presence of path limit switches, that will limit the possible position that every joint may reach.

Due to the physical limits of the robots does exist an area that limits the achievable points, while points out of this area will never be reached. The physically reachable point usually is referred as the working envelope.

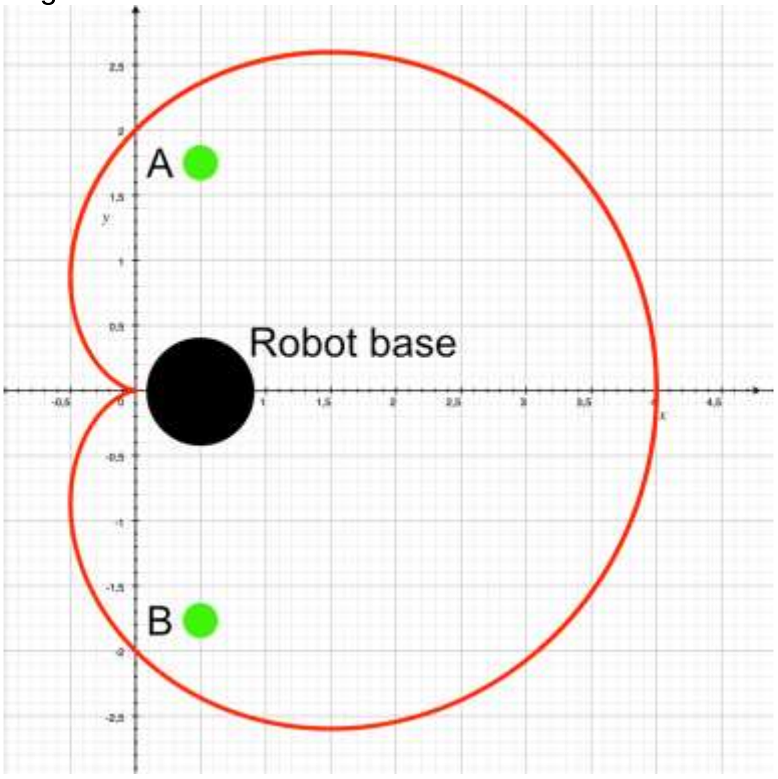


The image shows the working envelope of the KUKA KR 210-2K - Courtesy KUKA AG
Source: https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/pf0031_kr_2102_k_en.pdf

A bit different, and more insidious problem, depends on points that are in the working envelope, but cannot be reached from a defined starting configuration and/or using defined restrictions.

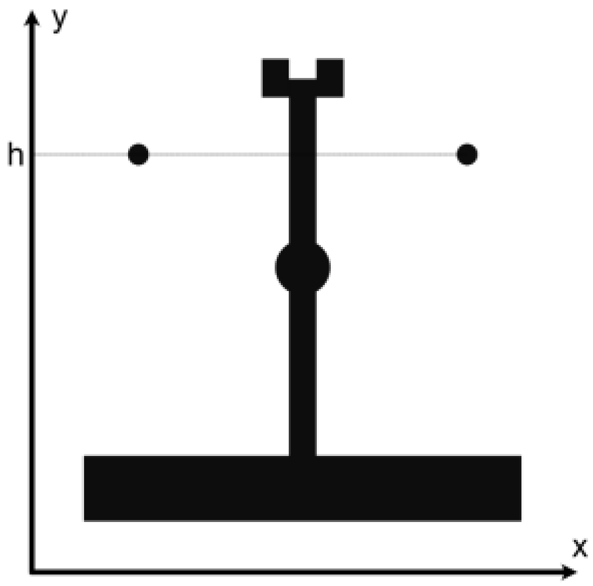
Let's imagine a hypothetic 2D working envelop like a cardioid (figure that is quite reasonable), as shown in the next image. At the centre of the cardioid there is the base of the robot itself. Imagine two points – A and B – located as seen in the image. Both are in the reachable area, so the end effector could suit it. Imagine, now, to try to move with a linear movement, parallel to the y axis, from A to B. Obviously, this cannot be done, since the base of the robot limit the possible path, so the point B can be reached, but not accordingly the request of a linear movement.

Sometimes it is important to add some external restrictions, such as explicitly force a kind of movement, or maintain a defined direction, or use a fixed orientation of a specific joint, to accomplish a movement. Other time it may not be mandatory to have a defined restriction, but we may add it to better manage the robot calculation of the path. However, not every time this can be simply done, and sometimes some clever idea should be used to reach the requested goal.



A related question, is the problem of singularities. As previously seen, robots have physical limitations. However, they are controlled by algorithms and mathematics, which have no physical limitations. For example, mathematically, it’s valid to have a joint speed of “infinity.” If the solution of the equations to compute the movement conflicts with the feasible movement, you reached a singularity. Usually you will obtain an error, or a weird (and sometimes hazardous) movement, or the robot only will stop. A few types of singularities exist, and this moves out of the purpose of this guide, however, just to understand the basics of the problem, will point at an example.

Imagine a 2D robot, with only 1 joint. Suppose that the robot is in a configuration where the link 1 is aligned to the link 2. Imagine you would like to lower the position of the tool to a desired position $y = h$. As you can see, two different solutions do exist: one turning the joint clockwise, the second one turning it counterclockwise. If two solutions exist, how can the robot choose one? Even more, with more articulated systems (so, not 2D, and with more than 1 joint, so every real-life robot) the problem may be even more hard, and the solutions may for example be infinite (wonder about the same problem but in 3D, in that case does exist infinite direction that the joint could use to lower the tool).



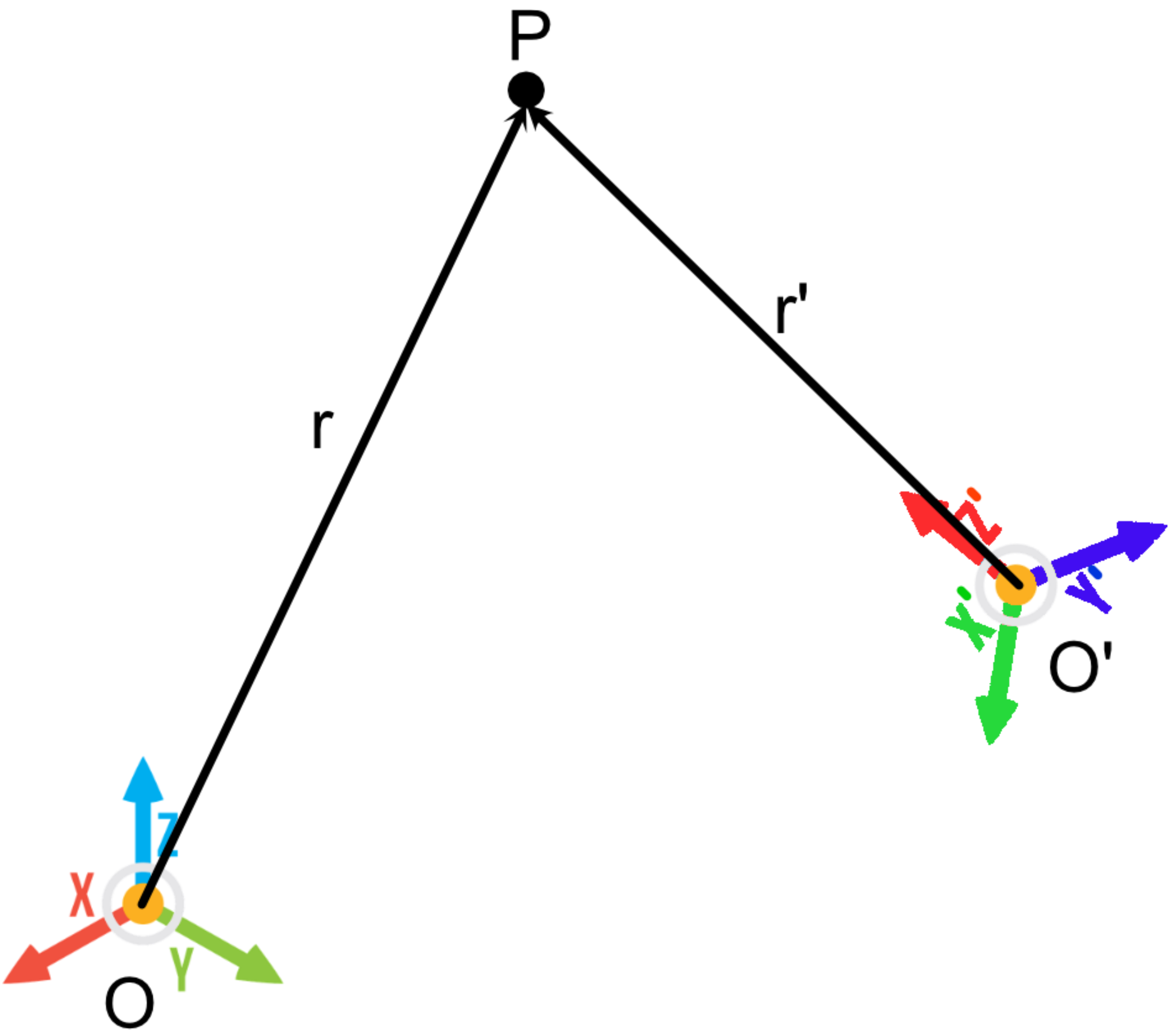
Robot software usually try to manage singularities, but sometimes they need some human workaround to deal with the problem.

(Owen-Hill, 2022; Siciliano et al. 2008)

Coordinate system transformation

In previous sections we noticed how it is possible to define a location in space, and how can a solid body be described in space. It is important to understand another concept, that relates to coordinate systems, and point of view.

A coordinate system may be arbitrarily defined, so considering Cartesian System you can define an origin point, and the direction of the axis. From another point of view, what does the previous sentence mean? Imagine you have a point in space. We can identify such a point by defining a Cartesian reference system, which we could call O , and indicate that this point with respect to O is at the coordinates $\vec{r} = (x, y, z)$. However, we can also define the position of the same point with respect to a different Cartesian reference system, let us call it O' for example, with respect to which the point is positioned at$\overrightarrow{r'} = (x', y', z')$. Obviously, the way we have described the point has changed, but the point itself has not changed.



This is the mathematical description of what we experience in everyday life, for example when we have a person in front of us. Our reference system is different from that of the other person, and if we ask this second person to move to the right, he will move relative to 'his/her' right, which is not necessarily 'our' right.

This is an important concept in the programming of a robotic arm. Both because it is necessary to be aware that our reference system does not coincide with that of the robot, but also because during programming itself it is possible to define points according to different reference points, and some of these may be more convenient than others depending on the situation.

For example, it is generally possible to have a reference system, typically called world, characteristically positioned at a notable point in the room or workbench where the robot is placed. Again, we can have a system typically called base, which usually coincides with the centre of its base on the workbench, or we can think as if our reference frame is positioned on the flange, or on the robot's tool.

There are mathematical relations that can allow us to pass from the description according to a given reference system to another, but usually these evaluations are automatically carried out by the robot's control system. For our purposes, it is only necessary to realise that we have the possibility of defining a point according to different reference systems, and that it is possible to switch from one to the other. In addition, it is always important to be aware of which reference system we are using at any given time, so that we are always sure of what the axis directions are and where the points are positioned in space.

(Craig, J., 2005; Siciliano et al., 2008)

Variable storage

In the previous section it has been described how the same point can be identified using cartesian coordinates, or in joint space. Usually it is possible to define which description one would like to use to record a position using a robotic arm.

If you record a position in joint space, even if it is harder to figure out and wonder the final TCP destination, this will be an absolute position, since you are telling the robot the exact position each joint should move to. This is pretty useful for out-of-the-way positions, or to define position you would like to be absolute (so indipendent from any changes in space, or in tool).

Otherwise, if you record it using the cartesian system, it is easier to wonder the position of the TCP. However, as you seen before that different references point may exist, theese are not absolute positions. If you change the reference point (depending on which system and command you are using) it may be that you are re-defining the stored position, respect to the new reference point.

Tools/end effectors

As previously noted in the introduction, industrial robots usually can mount different tools or end effectors, so different devices attached to the end of the arm that interacts with the environment to perform a specific task.

Does exist different kinds of tools, depending on the specific job that the robot needs to do. Every different job usually requires a specific end effector. The most common tool is some sort of gripper, so a device capable of grasping objects using some finger or other techniques to pick up objects. For example, does exist grippers that uses vacuum systems and suction caps, or they can use magnetic picking systems.

If the task is not picking up objects, but more like performing an action a process tool is required. Does exist, for example, tools that can weld, rivet, cut, paint etc. Imagine an electric power tool, like a drill or a welder: does exist its robotic version, that may be mounted as a tool.

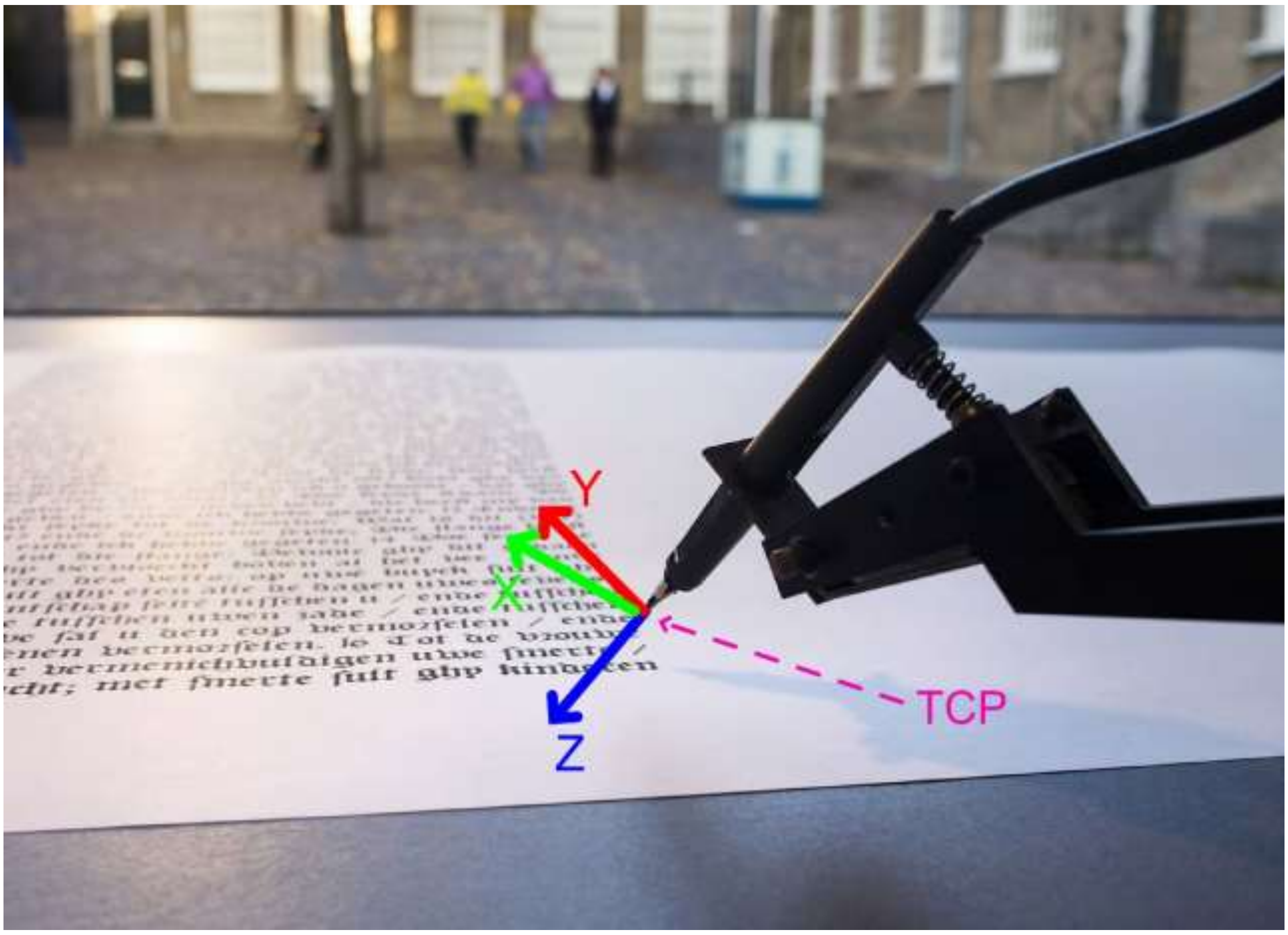
Further, some robots may change the tool during the operations, using a tool changers solution, that let release the first tool, and mount a second one ("Automatic Tool Changer", 2024; "Robot end effector", 2024).

ROBOTIC ARM
END-EFFECTORS

Robots may also use remote tools, with the workpiece mounted on the flange, and the tool outside the arm.

Since several tools may be used, it is significant to understand how to manage different utensil, and to do so it is mandatory to sink in the concept of Tool Centre Point (TCP). The TCP is the specific point on the end effector of a robotic arm where the tool's interaction with the environment is focused. It is the reference point used for programming and controlling the robot's movements and is crucial for tasks requiring precision and accuracy. For example, using a welding torch the TCP is the tip of the welding torch where the welding occurs, while for grippers the TCP might be defined at the centre between the gripping fingers when they are closed, and using a drilling tool the TCP is the tip of the drill bit. (Goodwin, 2022).

It is clear that using mathematic method to calculate the transformation of coordinates and kinematics it is feasible to locate the TCP in space, and points in space related to the TCP reference frame. Usually, robot can do it by themselves. In the TCP frame, the orientation of the XYZ axes is typically defined based on the end effector's standard working orientation, but may vary according to the convention used in robotic programming. Usually, the Z axis point toward the main movement of the robot tool (for instance, in a welding torch or drilling tool, the Z-axis often aligns with the tool's longitudinal axis or the direction in which the tool is expected to move forward), and XY axys follow the right hand rule. In the next image (Courtesy KUKA AG, edit by authors) you may see the KUKA robot writing a book, as seen in previous section, where it is highlighted what the TCP is, and a possibile orientation for TCP frame.



Robot Simulation System [Missing]

In the realm of industrial robot programming, simulation software serves as a vital tool for engineers and technicians alike. These programs create virtual environments where robots and their operations can be tested and optimized before implementation in real-world scenarios. Imagine a digital playground where you can design, program, and validate robotic tasks without the need for physical robots or risking production downtime.

These simulations accurately replicate the movements and behaviors of robots based on their physical characteristics and programmed instructions. They allow users to visualize how robots interact with their surroundings, such as workpieces and production lines. This capability is particularly useful for fine-tuning robot trajectories, ensuring smooth and efficient movements that minimize cycle times and avoid collisions.

One of the significant advantages of simulation software is its ability to support offline programming. This means engineers can develop and debug robot programs without access to the actual robot hardware. By integrating with Computer-Aided Design (CAD) software, simulations can directly import 3D models, facilitating precise modeling of work cells and production environments.

Simulation software also plays a crucial role in virtual commissioning, where engineers validate and optimize robot programs before deploying them in real production environments. This process helps identify and rectify potential issues early on, reducing risks and ensuring that robots perform tasks safely and accurately once deployed.

Furthermore, these tools are essential for educational purposes, providing students and trainees with hands-on experience in robot programming and operation. By using simulation software, learners can practice programming techniques, experiment with different scenarios, and gain a deeper understanding of robotics without the constraints of physical hardware.

In essence, simulation software for industrial robot programming empowers users to innovate, optimize, and enhance the performance of robotic systems across various industries. It bridges the gap between design and execution, enabling efficient deployment and operation of robotic solutions in modern manufacturing and automation processes.

Several different robot simulator does exist, for example RoboDK (Brand indipendent), RoboShop (COMAU), RoboSim (COMAU), KUKA Sim (KUKA), DobotLab (Dobot) RobotStudio Suite (ABB) etcetera.

They can help programmer in managing robots, in a safe environment, before putting hands on real robots. Inside this manual you will find some examples done with real robots, as well as examples performed with simulators.

Safety and Security Precautions

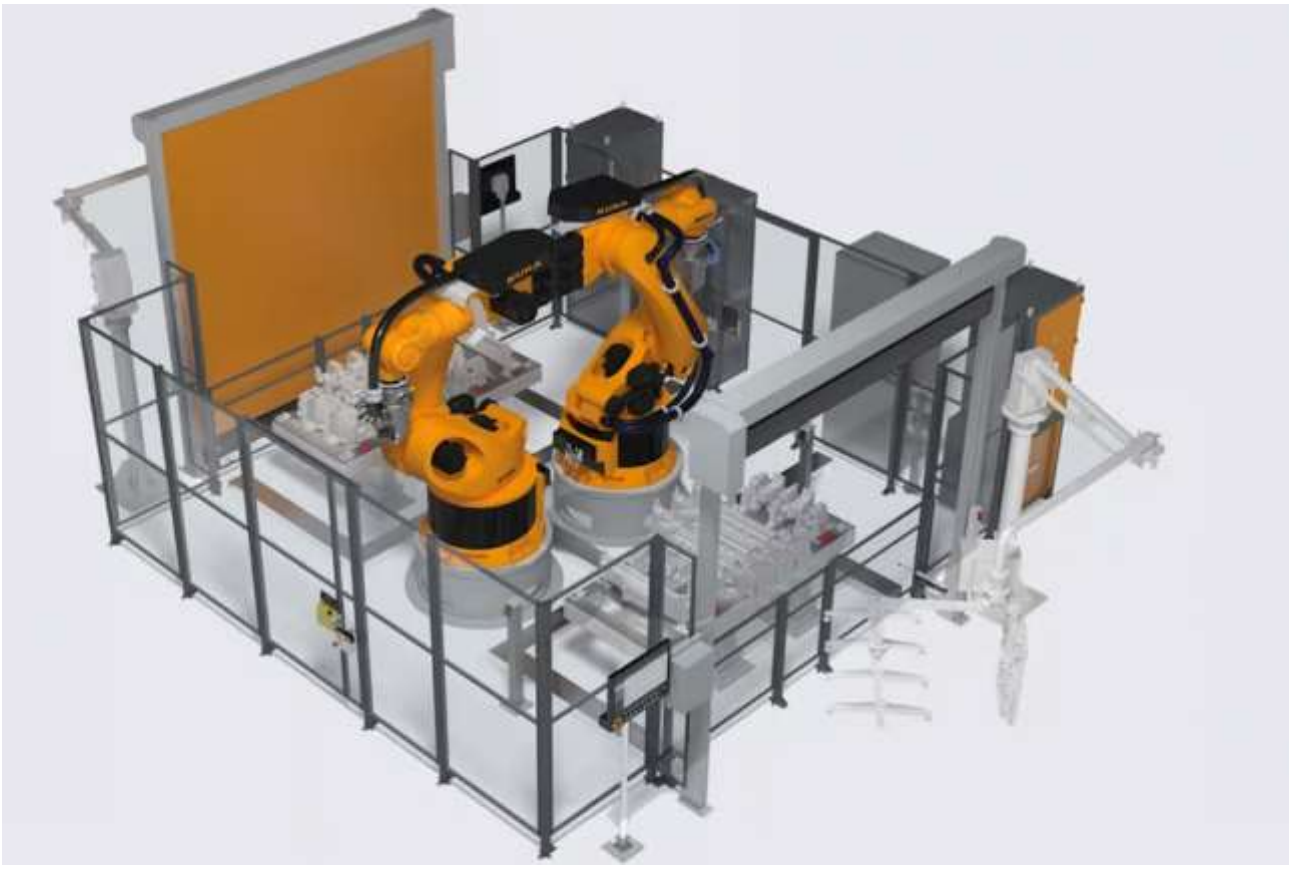
When operating robotic arms in industrial settings, adherence to safety standards and procedures is paramount to ensure the well-being of personnel and the efficient functioning of equipment. Industrial robots, often operating within dedicated safety cells, pose specific hazards related to their speed, weight, and force capabilities. Understanding and mitigating these risks is crucial for maintaining a safe working environment.

Safety standards such as ISO 10218 for industrial robots and ISO 13849 for safety-related parts of control systems provide guidelines for the safe design, operation, and integration of robotic systems. These standards emphasize the importance of risk assessment, protective measures, and safety monitoring to mitigate hazards effectively.

For our purpose it is important that robotic arms can move at high speeds and handle heavy payloads, presenting significant hazards if not properly controlled. The speed of robotic movements can lead to collisions with objects or personnel, while the weight and force exerted by robot arms can cause crushing injuries or damage to equipment.

Another significant risk associated with operating robotic arms in industrial settings is the potential for pinching or getting parts caught during movements: objects, including body parts or tools, can inadvertently be caught between moving parts of the robot or between the robot and its surroundings. This risk underscores the importance of maintaining safe distances, implementing effective guarding, and using sensors to detect unexpected obstacles or personnel within the robot's operating envelope.

To avoid this issues, industrial robots are often deployed within safety cells or guarded areas equipped with physical barriers or light curtains. These safety measures prevent unauthorized access and reduce the risk of collisions between humans and robots, however educational robots not always have this feature. Even if they are usually less hazardous, you should always not underestimate the risk of damages to users, surroundings, or the robot itself. In the image below (courtesy KUKA AG) you may see a roboti cell, so an environment where multiple robots and other industrial tools work together to perform a task. Note the fences and gates that prevent unautorized access.



Once that the most common hazard using a robot are known, it is mandatory to understand how human behaviour could expose us to risks.

Intelligent planning involves anticipating potential hazards and implementing preventive measures during robot setup and operation. This includes identifying pinch points, establishing safe operating distances, and ensuring clear communication of safety procedures to all personnel involved. It is also important to program a robot critically thinking about what it will going to do, and taking into account that it could be hazardous.

Before operational deployment, thorough testing and intelligent programming are essential. Every time you create a new program it is important to test it before run it automatically. Testing procedures should include running the robot at low speeds to verify trajectories, validate safety measures, and ensure correct program execution. This step-by-step approach minimizes risks associated with programming errors or unforeseen operational issues.

Using low speeds during testing allows operators to observe and intervene if necessary, without exposing personnel or equipment to high-speed hazards. This precautionary measure enhances safety during initial setup and debugging phases.

Industrial robots usually have emergency stop switches. They are critical safety devices installed within easy reach of operators. These switches immediately halt robot operations when pressed, providing a rapid response in emergencies to prevent accidents or injuries. Do not hesitate in pressing it in case of any emergency.

Industrial robot with a Teach Pendant usually has also dead man (enable) switches. They are switches usually located on the back of the teach pendant, that has three positions: depressed/released, fully pressed, halfway pressed. The robot is enabled to work only if the user presses the switches in the halfway position: this is requested for safety reasons. Pressing halfway requires that the program start is intentionally requested, and in case of any emergency or issue the operator usually releases the buttons, causing the robot stop. In some cases, an operator may grasp the teach pendant in case of danger, doing so the switches will be fully pressed and also in this case the robot will stop.

Operation modes

Industrial robots usually have different operating modes, used for different reasons. They are usually switched via a software button on the teach pendant, or a key switch on the pendant or case. A few different mode does exist, and may be called differently according to a different brand, but at least two should be noted: programming mode (somewhere called manual or T1), automatic mode (AUT).

- In programming mode, the robot usually moves at a reduced speed, it may be jogged using the teach pendant. This is the mode you usually use to manually move the robot, program it and do some tests. To execute a program, you have to press the dead man switches, and press (and usually keep pressed until the end) the program start button. Be aware that even if the robot moves slowly, usually this mode disable some (or all) safety restrictions, so you should carefully watch what the robot does, and immediately stop it (releasing the dead man, or start button) in case of unexpected behaviour. In manual mode usually, even if present, the robotic cell may be opened, and the operators may work (carefully) inside the robot cell.

- **IF ONLY AND JUST ONLY A PROGRAM HAS BEEN ALREADY TESTED** a robot may be switched to automatic mode, where usually it can move at full speed, it is not required to hold the dead man switches, and once the start button is pressed the program begins. In this case in case of emergency you SHOULD press the emergency stop button. If you want to stop the robot in a non-emergency situation you probably have access to a program stop button on the teach pendant. In automatic mode robots usually has all security systems on, and in a cell is present the robot may work only if the cell is closed.

Other working mode does exist (for example full speed test mode, or remote control mode) but we will not discuss this in detail.

Fieldbus

A really important aspect in industrial robotics is the fieldbus. Due to its complexity, and extreme different approaches that may be used, we will not cover this topic inside the manual, and we will focus solely on the robot itself. However, since it is a very important and common aspect, we provide you only a brief introduction, just to discover a bit of the concept.

In the realm of industrial robotics and automation, efficient communication between various devices and systems is crucial. This is where Fieldbus technology comes into play. Fieldbus is a network system for real-time distributed control, providing a standardized way for devices such as sensors, actuators, and controllers to communicate with each other. It is a family of industrial computer network protocols used for real-time distributed control. Unlike traditional point-to-point wiring, Fieldbus networks allow multiple devices to be interconnected through a single cable, facilitating efficient and reliable data exchange. This digital communication network connects various field devices such as sensors, actuators, and controllers, enabling them to communicate and operate in unison.

The Fieldbus system comprises several essential components. The master device, often referred to as the controller, oversees the network by initiating communication and coordinating data exchange among connected devices. The slave devices include sensors, actuators, and other field devices that perform specific tasks based on commands received from the master device. Communication media, transmit data signals between devices. Network interfaces are the hardware components that connect devices to the Fieldbus network, ensuring proper data transmission and reception. The protocol is the set of rules governing data exchange on the Fieldbus network, ensuring that all devices can communicate effectively. Several Fieldbus protocols are widely used in industrial automation, each catering to different requirements and standards.

Fieldbus technology offers several advantages in industrial automation. It significantly reduces wiring complexity and costs by using a single communication cable to connect multiple devices, compared to traditional point-to-point connections. The advanced diagnostic features supported by Fieldbus networks enable real-time monitoring and troubleshooting of connected devices, improving maintenance and reducing downtime. The scalability of Fieldbus systems allows for the easy addition of devices without major modifications, facilitating expansion and upgrades of industrial automation systems. Digital communication in Fieldbus ensures accurate and reliable data transmission, reducing the risk of signal degradation and interference associated with analog systems. The flexibility and compatibility of Fieldbus protocols with a wide range of devices and manufacturers promote interoperability and easier integration.

Fieldbus technology plays a vital role in industrial robotics by providing a robust, efficient, and scalable communication network for real-time distributed control. Its ability to reduce wiring complexity, enhance diagnostics, and improve data integrity makes it an indispensable component of modern industrial automation systems. As industrial robotics continues to evolve, the adoption and advancement of Fieldbus technology will be crucial in driving greater efficiency, reliability, and innovation in automation processes.

CHAPTER 3

This third chapter is the core of this manual, devoted to introducing and present some of the applications of robotic arms. Since a fully comprehensive manual, suitable for any different application is not achievable, we just bring a few typical applications of robotic arms. In the selection process we considered that the purpose of this document is to foster knowledge in the field of robotics, but also to support the learning in VET and technical schools, so we are proposing a selection of activities that can be directly done in class.

Several kinds of robotic arms – both by brand and by technical capabilities – exist, and also in this case it is not possible to cover all the existing cases. We are focusing on articulated arms, as proposed in the introduction, because of ease of use and diffusion. Furthermore, every different arm uses different approaches to do the same task, so we are going to propose a set of activities that we are going to introduce as general as possible, and then we will discuss deeper providing some case studies, selecting a specific arm. For different activities we will propose a few different arms, selected between the most common in industry and education world (KUKA, Mitsubishi, COMAU, Dobot). Keep in mind that they are only examples and case studies, so almost every proposed activity may be adapted to a different kind of robot or analysed using different kind of simulators.

Reference note: in this chapter we refer to every proposed activity and program to our expertise, and to official robotic arms manual, proposed in the bibliography section.

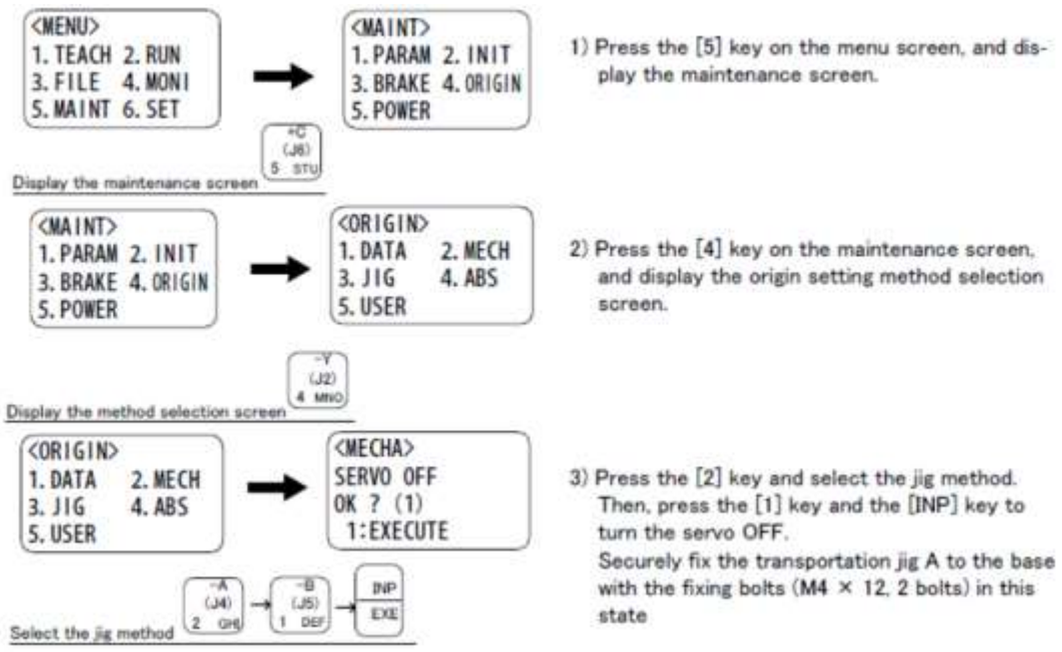
Adjusting the starting position

Before starting the robotic arm, we need to be aware that in most cases the robot does not know its current position. Therefore, we must adjust the starting position of the robot. Each robotic arm brand proposes several methods for adjusting the starting position, and all of them can be found in the appropriate manuals of the robot. Here we present several adjusting methods, most suitable for Mitsubishi and Comau brand.

- Mitsubishi example -

Mechanical stopper method (operating the teach pendant T/B, and manually moving the arm to the starting position):

- Menu / 5-Maint / 4-Origin / 2-Mech / 1 EXE



1) Press the [5] key on the menu screen, and display the maintenance screen.

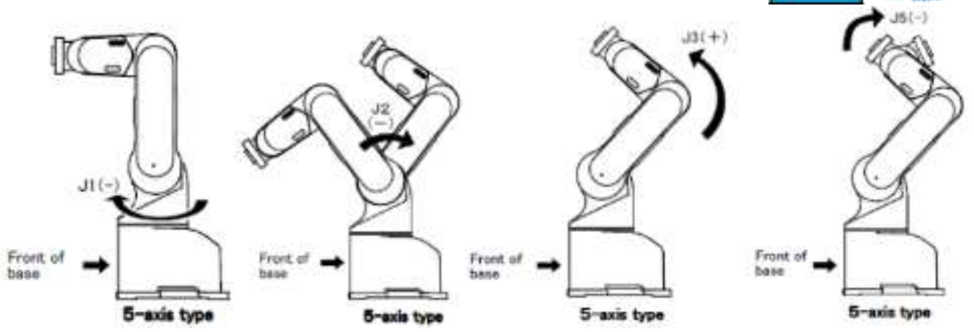
2) Press the [4] key on the maintenance screen, and display the origin setting method selection screen.

3) Press the [2] key and select the jig method. Then, press the [1] key and the [INP] key to turn the servo OFF. Securely fix the transportation jig A to the base with the fixing bolts (M4 × 12, 2 bolts) in this state

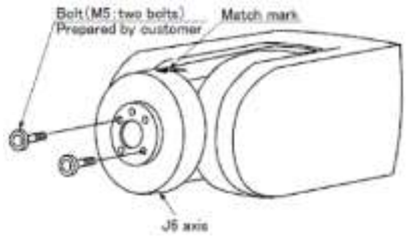
- Axis: 11101100

IMPORTANT – the following procedure turns off the servo breaks. This operation requires two students – first one operates the T/B, and the second moves the robotic arm manually to initial position.

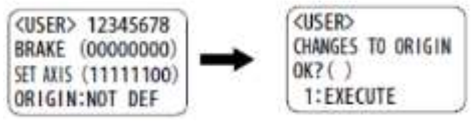
- Press the DEADMAN, STEP, and X+ on the teach pendant – this will turn off the servo breaks!
- The second student manually moves the robotic arm in the following directions: J2-, J1-, J3+, J5-



- J6 is rolled manually until the both mark points match



At this point, robotic arm is placed into its` initial position, and one can proceed with the T/B setting.



```
<USER> 12345678          <USER>
BRAKE  (00000000)        CHANGES TO ORIGIN
SET AXIS (11111100)  →   OK?( )
ORIGIN:NOT DEF            1:EXECUTE
```

- In the BRAKE row, please insert all zeros;
- In the AXIS row, please insert 1 and every axis you wish to save the coordinates for – in our case it will be 111011 (axis 1, 2, 3, 5, 6)
- Confirm your choice by entering - EXE / 1 / EXE

Types of movements of the robotic arm

Robotic arms usually move from a point to another one (point to point – PtP) or following a specific path. We will focus here on PtP movements. Let's consider two different points in a 3D space. Does exist an infinite number of trajectories that connects them. Obviously, robots should use one of these. Robots uses a mathematical approach called interpolation to determine which path it should follow to joint two points; however different kind of interpolation exist. Understanding these concepts helps in programming robots to perform tasks efficiently and accurately, depending on what you want the robot to do.

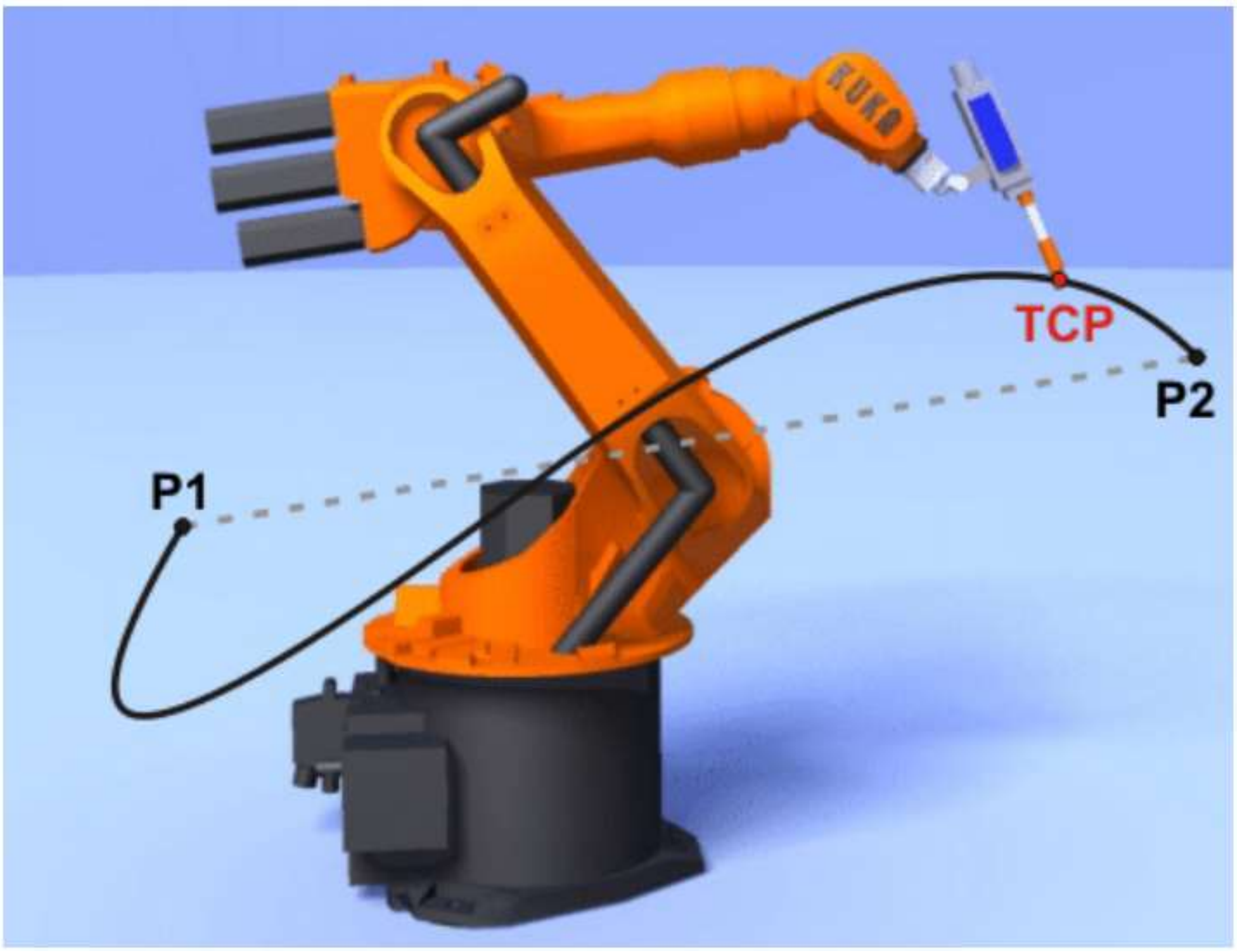
Joint Interpolation

The first interpolation mode to introduce is the joint mode. It is the standard in almost all of the industrial robots.

This movement type controls each joint of the robotic arm individually. The robot moves each joint to the desired position as quickly as possible, and as easier for the robot is, without considering the path the end-effector (the tool or gripper at the end of the arm) takes.

Example: Imagine your arm is a robot, and you want to touch your nose. You move your shoulder, elbow, and wrist all at once to get your finger to your nose. Each joint moves directly to its target position.

This is the easy way to move a robot, and with and articulated robot consist of a sum of circular motions: each joint in an articulated robot can rotate around its axis. When you command the robot to move its joints from their current positions to new target positions, each joint rotates independently to its new angle. Since each joint's movement is rotational, the end-effector's path is a result of combining these rotations.The robot does its calculations as easy as possible, and the movement is usually the faster.



Joint interpolation motion

However, the robot is not aware of what is present in the environment, and what the user would do. Since each joint contributes its own arc of movement, the end-effector's path is not a straightforward line or smooth curve but a combination of these arcs, making it difficult to predict the final trajectory. The exact trajectory depends on the sequence and speed of joint movements, which are often not intuitive to the user. Since the end-effector's path is hard to predict, there is a risk of collision with objects in the robot's environment. The robot does not inherently account for obstacles, workpieces, or other equipment when using joint interpolation. For example, a joint may swing an arm wide around an unexpected arc, potentially hitting something in its path.

The intermediate positions of the end-effector can vary greatly depending on the starting and ending joint angles. These unexpected intermediate positions can cause the robot to move through unintended areas, posing risks to both the robot and its surroundings.

Joint interpolation is, nevertheless, crucial. So how is it possible to use it? First, user should be aware of the potential risks and reduce the use of joint interpolation to the strictly necessary. Further, the user should critically think about where and when to use this. Then, every time a joint interpolation is used it should be tested carefully, with more care than in other cases. Once a joint interpolation has been tested, it could be used with no more issues, since every next execution in joint mode from the two previous point will follow the same trajectory, already tested.

Examples of command words to do a joint interpolation using a KUKA robot:

PTP home 0 TOOL_DATA[1] <Null> 100%

Here we see that the joint interpolation is achieved by a command line, starting with a PTP (point – to – point) command. The “home 0” is a target position. The tool data is being stored in the TOOL_DATA [1]. The base position coordinates are stored in the <Null> base data. The velocity of the movement is 100% - a maximum.

Examples of command words to do a joint interpolation using a Mitsubishi robot:

```
MOV P1          ' Moves to P1.
MOV P1+P2       ' Moves to the position obtained by adding the P1
and P2 coordinate elements.
MOV P1*P2       ' Moves to the position relatively converted from
P1 to P2.
MOV P1, -50     ' Moves from P1 to a position retracted 50mm in the
hand direction.
```

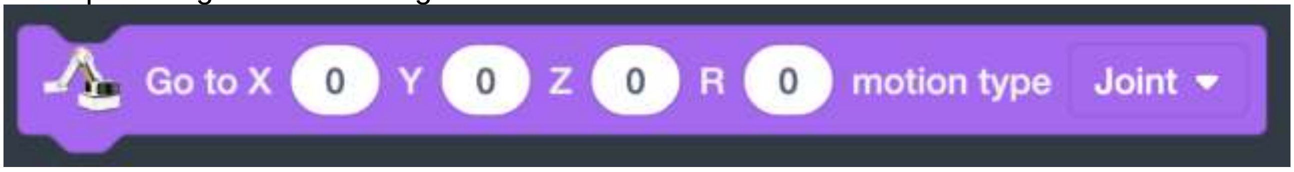
Example of command to do a joint interpolation using a COMAU eDo robot:

```
MOVE TO P1
```

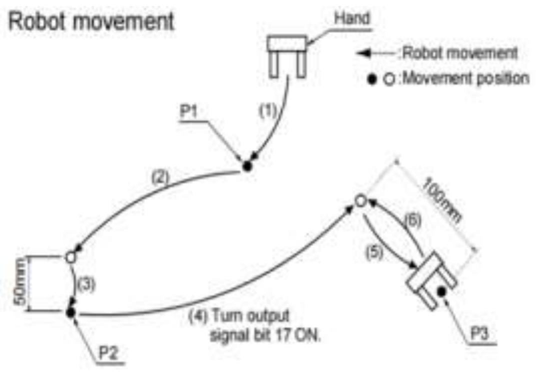
That is the default option, equivalent to

```
MOVE JOINT TO P1
```

Example using the Dobot Magician Lite



Task – write a custom program to do a following series of movements on various robotic arm brands, using joint interpolation:

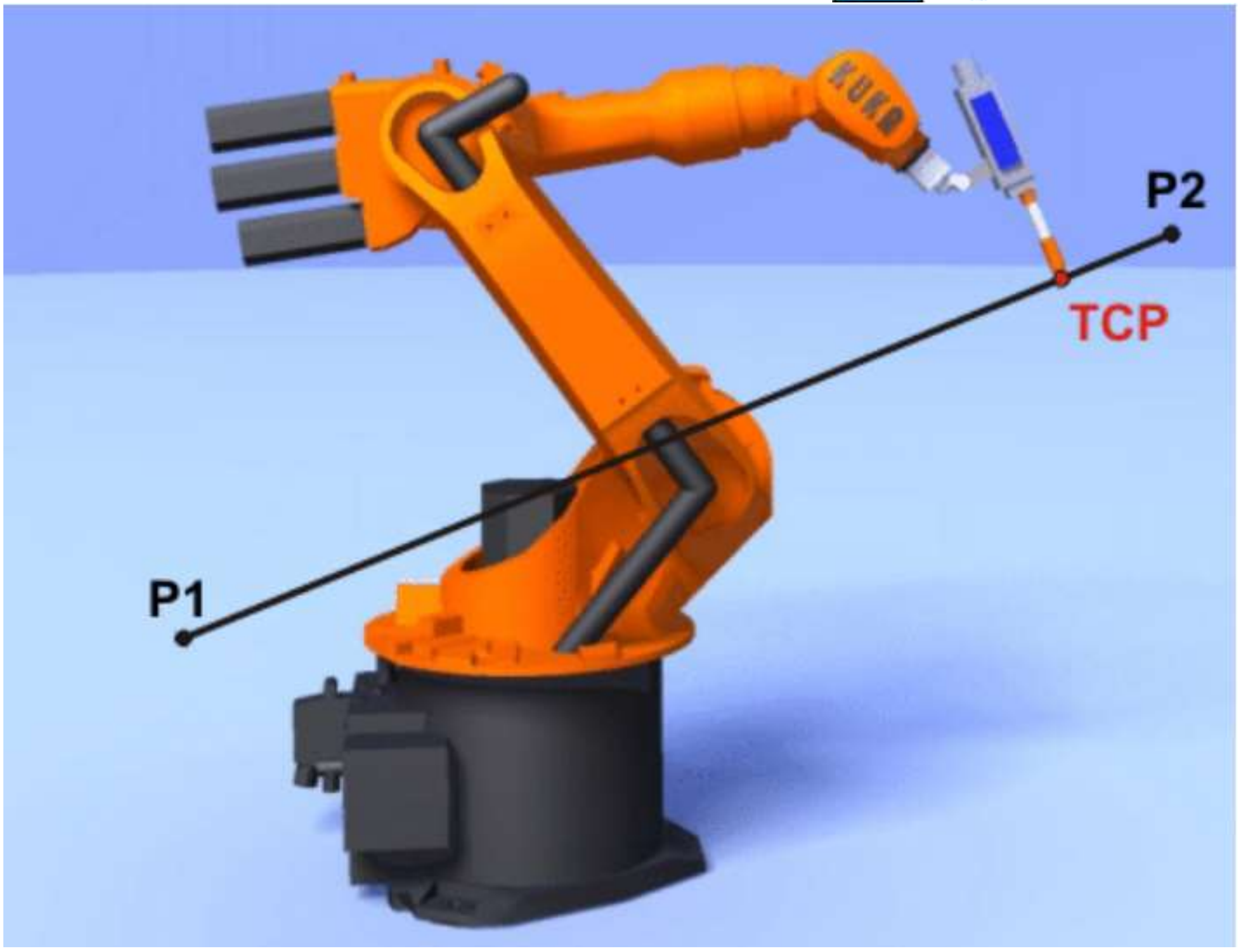


Linear Interpolation

A different, yet fundamental movement, is the linear move. This movement type ensures the end-effector moves in a straight line between two points. This is obtained adding the linearity constraint, it is harder to calculate for the robot, and not always doable (for example wonder to move your hand from the tip of your nose to the back of your head following a linear path: it cannot be done), but it is extremely predictable.

In linear interpolation the robot calculates a straight path and moves all joints in a coordinated way to keep the end-effector on this path, for **example** If you hold a marker and draw a straight line on paper from one point to another, your hand's movement is like linear interpolation.

This movement is really useful, especially if you want to be totally sure about how you will approach a target, or if you need to reach it from a specific orientation, for example linear moves is crucial in approaching items you should collect with a gripper: you may move over the piece (with different kind of movements, accordingly to your needs), but then go down to the piece using a straight line.



Linear motion

Examples of command words to do a linear interpolation using a KUKA robot:

```
LIN P1 TOOL_DATA[1] <Null> 20mm/s
```

Here we see that the joint interpolation is achieved by a command line, starting with a LIN (linear motion) command. The "P1" is a target position. The tool data is being stored in the TOOL_DATA [1]. The base position coordinates are stored in the <Null> base data. The velocity of the movement is 20mm/s.

Examples of command words to do a linear interpolation using a Mitsubishi robot:

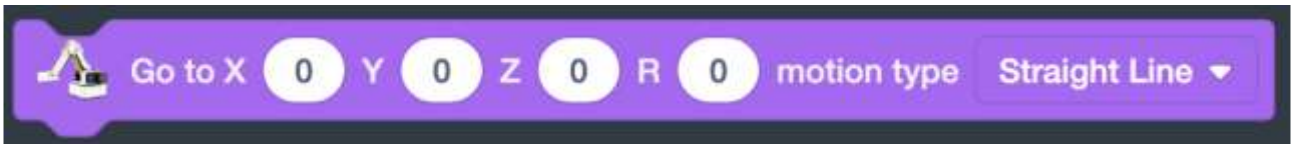
```
MVS P1            ' Moves to P1
MVS P1+P2         ' Moves to the position obtained by adding the P1
and P2 coordinate elements.
MVS P1*P2         ' Moves to the position relatively converted from
P1 to P2.
```

```
MVS P1, -50       ' Moves from P1 to a position retracted 50mm in
the hand direction.
MVS, -50          ' Moves from the current position to a position
retracted 50mm in the hand direction.
```

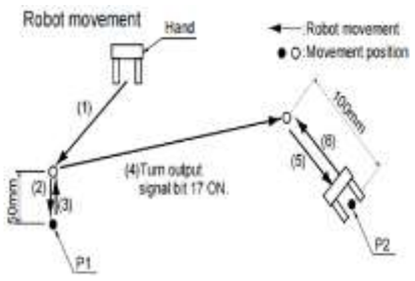
Example of command to do a linear interpolation using a COMAU eDo robot:

```
MOVE LINEAR TO P1
```

Example using the Dobot Magician Lite

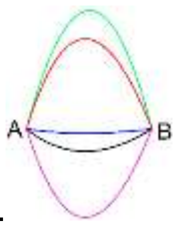


Task – write a custom program to do a following series of movements on various robotic arm brands, using linear interpolation:

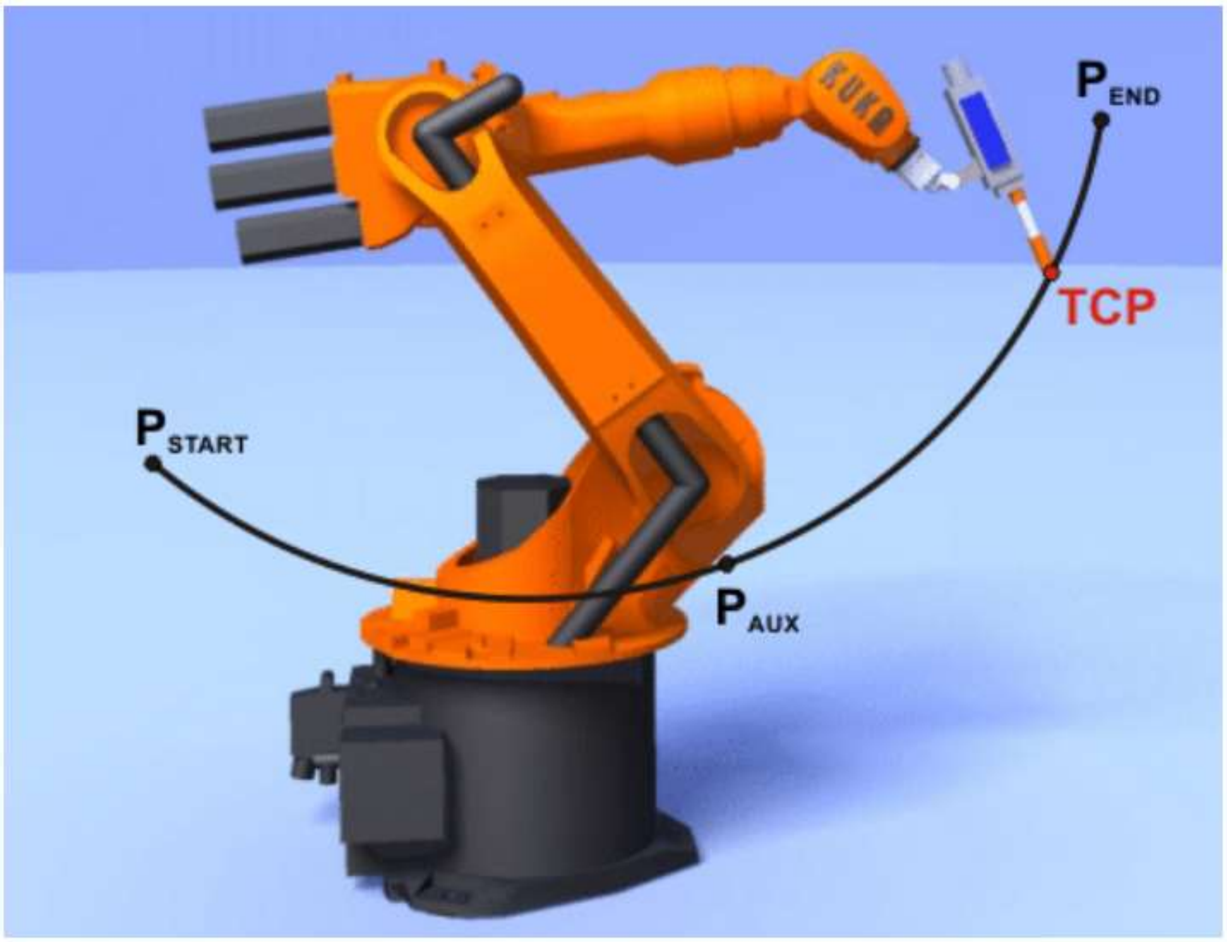


Arc Interpolation

A third interpolation that requires and additional constraint is the arc interpolation. This movement type makes the end-effector move along a curved path or an arc. The robot moves through a series of points that define the arc, coordinating the joints to follow this curved trajectory. However, if you want to move from a starting position to an end position it is not enough to define only this point, since sever arches may connect two points as you can see in the next image, where point A and B are connected by several arches.



There are different ways of determining a unique arch, for example providing starting point and radius, or providing start and end point plus the constraint that the arch necessarily passes through a third passage point (3-point arches definition).



Examples of command words to do a circular interpolation using a KUKA robot:

```
CIRC P1 P2 Vel=2 m/s CPDAT1 Tool[1]:TOOL_DATA[1] Base[0]
```

Here we see that the joint interpolation is achieved by a command line, starting with a CIRC (circular motion) command. The “P1” is a help position, while the “P2” is a target position. Movement data is stored in the CPDAT1. The tool data is being stored in the TOOL_DATA [1]. The base position coordinates are stored in the Base[0] base data. The velocity of the movement is 2m/s.

Examples of command words to do a circular interpolation using a Mitsubishi robot:

```
Command word - MVR
```

Explanation - Designates the start point, transit point and end point, and moves the robot with circular interpolation in order of the start point - transit point - end point.

```
Command word – MVR2
```

Explanation - Designates the start point, end point and reference point, and moves the robot with circular interpolation from the start point - end point without passing through the reference point.

```
Command word – MVR3
```

Explanation - Designates the start point, end point and center point, and moves the robot with circular interpolation from the start point to the end point.

```
Command word – MVC
```

Explanation - Designates the start point (end point), transit point 1 and transit point 2, and moves the robot with circular interpolation in order of the start point - transit point 1 - transit point 2 - end point.

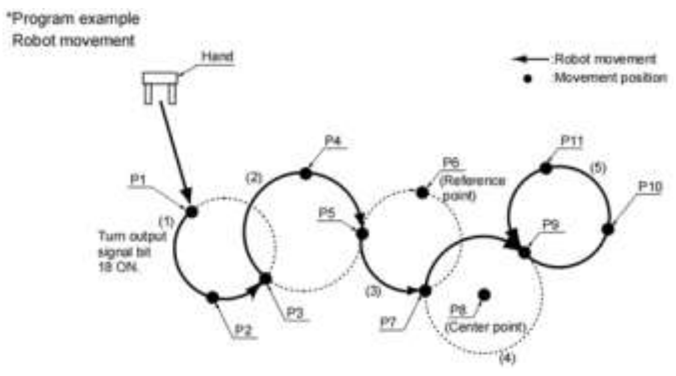
Example of arch interpolation using COMAU eDo

The eDo COMAU robot uses the 3-point definition of arches, so the command result as follows:

```
MOVE CIRCULAR TO P2 VIA P1
```

It means that it is requested to move the robot to the point called P2, through an arch, that necessarily passes via P1. Note that the starting point is not defined: this is because it is the point where the TPC is located at the moment of the execution of the command.

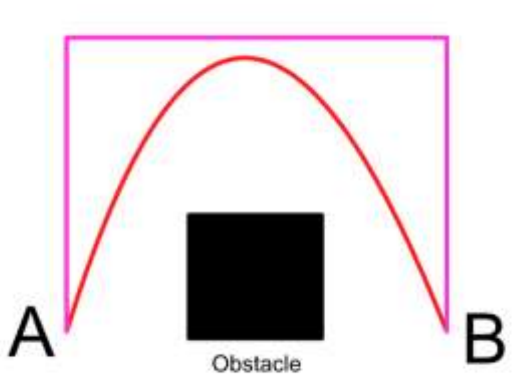
Task – the following image represents circular movement of the robotic arm and the proposed solution using Mitsubishi robot. Write a custom program that follows the same movements on your robotic arm brand:



10 MVR P1, P2, P3 WTH M_OUT(18) = 1 ' (1) Moves between P1 - P2 - P3 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point. (P1) output signal bit 18 turns ON simultaneously with the start of circular movement. 20 MVR P3, P4, P5 ' (2) Moves between P3 - P4 - P5 as an arc.30 MVR2 P5, P7, P6 ' (3) Moves as an arc over the circumference on which the start point (P5), reference point (P6) and end point (P7) in the direction that the reference point is not passed between the start point and end point.
40 MVR3 P7, P9, P8 ' (4) Moves as an arc from the start point to the end point along the circumference on which the center point (P8), start point (P7) and end point (P9) are designated.
50 MVC P9, P10, P11 ' (5) Moves between P9 - P10 - P11 - P9 as an arc. The robot current position before movement is separated from the start point, so first the robot will move with linear operation to the start point.(1 cycle operation)
60 END ' Ends the program.

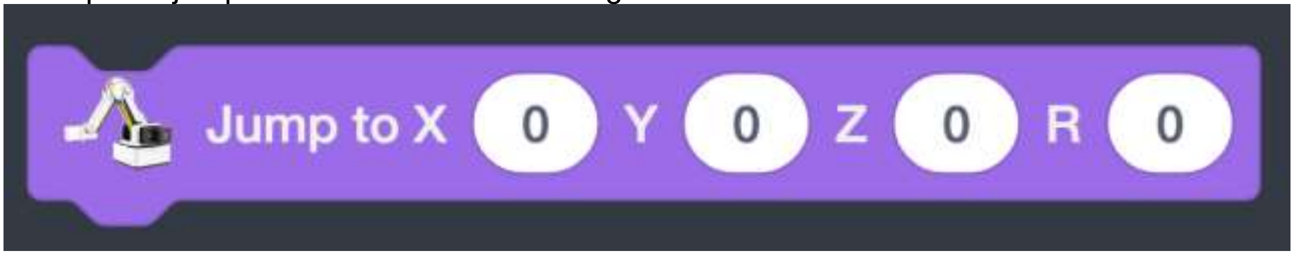
Jump Interpolation

The jump interpolation is not so common in robotic programming, but it is a very suitable command. This movement type allows the end-effector to move from one point to another doing a "jump" so moving on a parabolic path upwards, or at least rising as if to leap over the course.
The robot moves to an intermediate safe position above the workspace before moving to the target point, avoiding obstacles in between. Example**:** Imagine you need to pick up two objects on a table, and you lift your hand high between the two points to avoid knocking anything over, as shown below.



Not all arms let you use the jump command, sometimes you need to "build" it yourself (i.e. the purple path in the previous image may be realised joining 3 sequential liner moves).

Example of jump command with Dobot Magician Lite


Jump to X 0 Y 0 Z 0 R 0

Opening and closing the gripper

Since many tasks of the robotic arms are in fact pick – and – place, it is obvious one needs to get to know how to control the gripper, which is essentially a picking tool.
Examples of gripper control command words using a Mitsubishi robot:

Command word - HOPEN

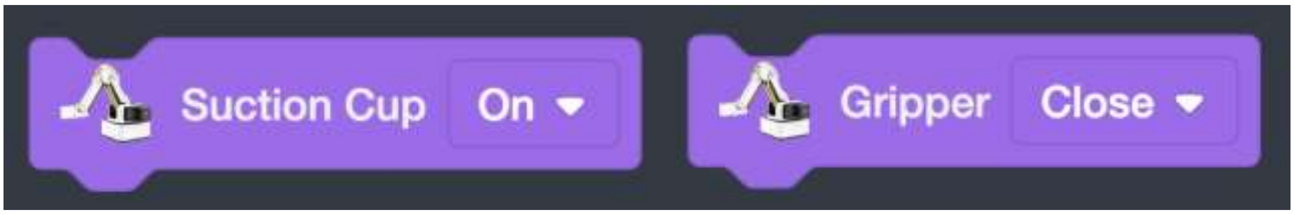
Explanation - Opens the designated hand.

Command word - HCLOSE

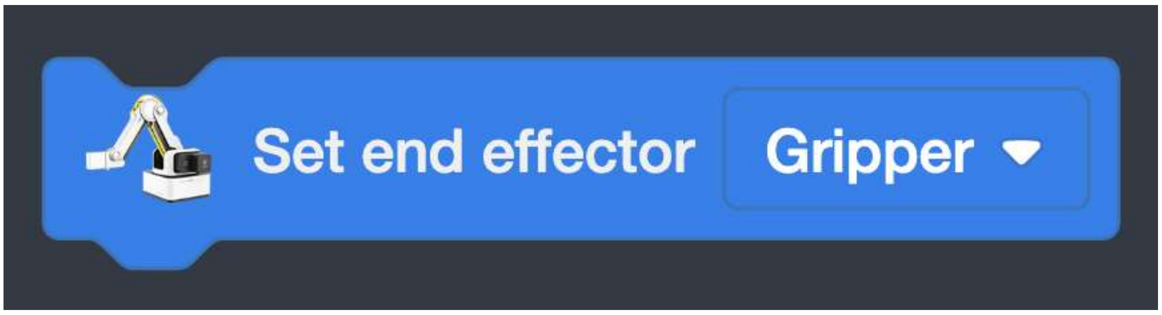
Explanation – Closes the designated hand.

Example of gripper control using DOBOT Magician Lite

Since the DOBOT Magician Lite is designed to mount as a tool both a gripper or a suction cup, programming it using blocks does exist two different commands to activate the desired tool, depending on which one would like to install and use. Trough the dropdown menu it is possible to select the desired action (so suction cup off, or gripper release).



Remember to select the correct tool BEFORE using it

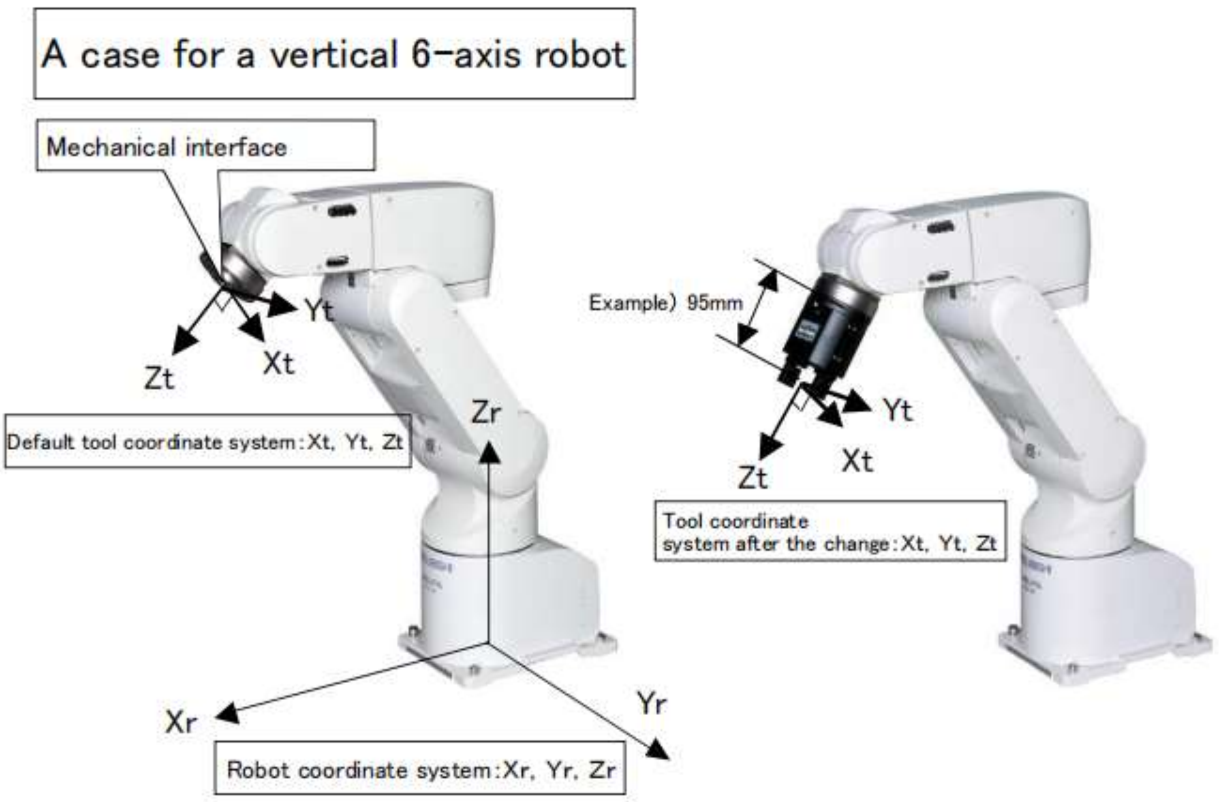


Example of gripper control using the COMAU eDo

The COMAU eDo robotic arm uses an electric motor-controlled gripper. The gripper is connected to an auxiliary axis (aux axis 7). When one would like to control the gripper has to control the aux axis. The easiest way is to use a regular movement command (like MOVE, MOVE LINEAR or MOVE CIRCULAR), but storing the position using an extended position variable, that let to control aux axis.

TCP (Tool Center Point) redefinition

The robot tool, or Tool Center Point (TCP), is the point used to move the robot to a Cartesian position (such as a Cartesian target given XYZWPR values). The TCP is defined as a transformation from the robot flange. Defining the TCP properly is important in any robot application, either if it involves Offline Programming or not. Robot TCP, also known as Tool Center Point, refers to the specific location on a robot where the end effector or tool attaches. It represents the center point of the tool and serves as a reference for the robot's movements and calculations. The TCP defines the position and orientation of the tool in relation to the robot's coordinate system, allowing for precise control and manipulation.


A case for a vertical 6-axis robot
Mechanical interface
Yt
Zt
Xt
Default tool coordinate system:Xt, Yt, Zt
Zr
Xr
Yr
Robot coordinate system:Xr, Yr, Zr
Example) 95mm
Yt
Zt
Xt
Tool coordinate system after the change:Xt, Yt, Zt

Example of TCP transformation command word using a Mitsubishi robot:

```
TOOL (0, 0, 95, 0, 0, 0)     Sets the robot control point to the
position 95 mm from the flange plane in the extension direction.
```

Example of TCP transformation command word using COMAU eDo or racer 3 Using COMAU eDo or racer 3 you can insert the offset data, as described before, trough the "Data" menu. You have the chance of defining multiple different tools or frames, identified by an ID number, that has a specific offset. Opening the menu, you will find an editable table similar to this (assuming it defines the tool)

ID number	x	y	z	φ	θ	ψ
1	0	0	0	0	0	0

2	0	0	95	0	0	0
3	100	150	0	0	50	0

The parameters are the offset respect to the reference point along x,y,z axis and rotations around the Euler Angles.

For example, in the previous table, the tool number 2 is defined with the TCP aligned to the flange, but 95 mm far from the flange center point along z direction.

You can then recall it through programming using the command

```
ToolFrame(<tool ID>, <frame ID>, <arm ID>)
```

Where the tool ID is the ID number of the tool you would like to use, as the frame ID is the number that defines the frame you would like to recall. Since one can program multiple arms simultaneously, it does exist also an arm ID that let select the arm. When using only one arm the arm ID is 1.

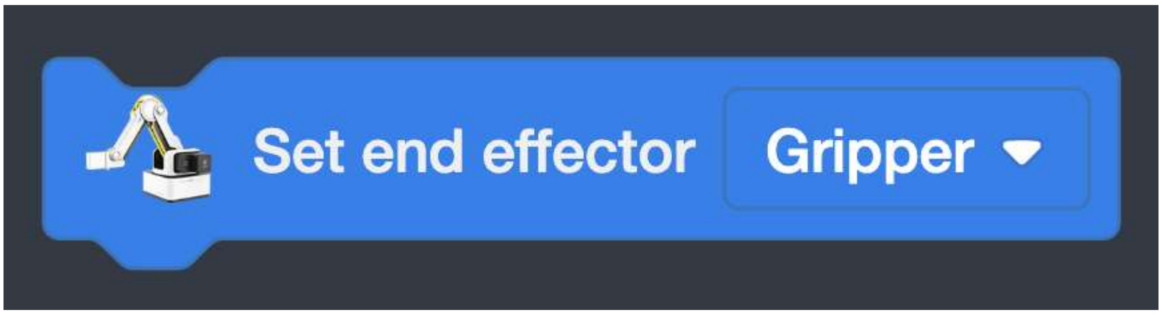
Assuming to use the tool number 2, referred to the frame number 1, using only one arm, you will call

```
ToolFrame(2,1,1)
```

Remember that until you execute this instruction, the ToolFrame command has no effect, so if you record any position, you are recording it respect to the previously definied tool and reference frame.

Proceding this way you should fill the data table manually. The COMAU arm has specific functions to automatically calculate the tool offset, but they are not treated inside this document. Please refer to official manual to discover this.

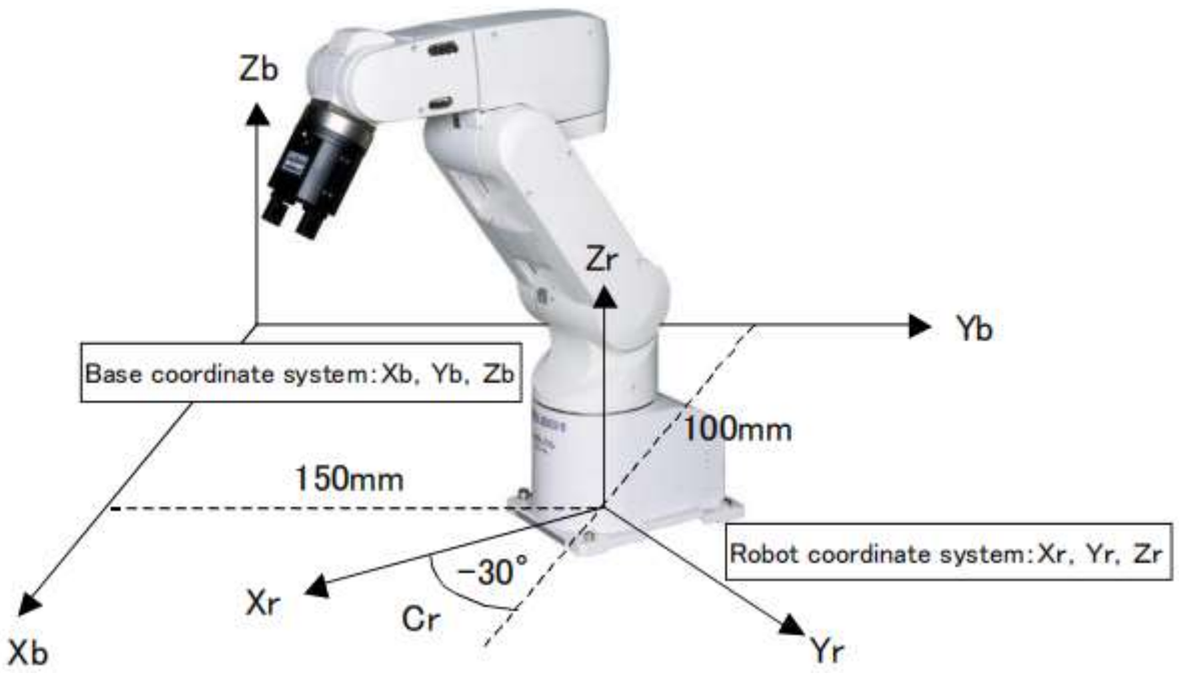
Example using DOBOT Magician Lite
Since the Magician Lite may mount only three official tools (gripper, suction cup, pen) the selection of the TCP can be simply done using a dropdown menu in the toll selector command seen before



Standard base coordinates

When shifting the robot origin to a position other than the center position of the J1 axis of the robot, the conversion is performed using the base coordinate system. When base data is changed, the coordinates of teaching positions will be values based on the base coordinate system.

Structure of base coordinate system data: X, Y, Z, A, B, and C X, Y and Z axis : The position of robot coordinate system from the base coordinate system origin
A axis : X-axis rotation in the base coordinate system
B axis : Y-axis rotation in the base coordinate system
C axis : Z-axis rotation in the base coordinate system


Zb
Zr
Yb
Base coordinate system:Xb, Yb, Zb
100mm
150mm
−30°
Robot coordinate system:Xr, Yr, Zr
Xr
Cr
Xb
Yr

Example of base coordinate system transformation command word using a Mitsubishi robot:

```
10 BASE (100,150,0,0,0,-30)
```

Example of base coordinate system transformationwith COMAU eDo or racer 3 The redefinition of the reference frame is done in the same manner as the definitition of a new tool, so refer to the previous section for details.

Speed regulation

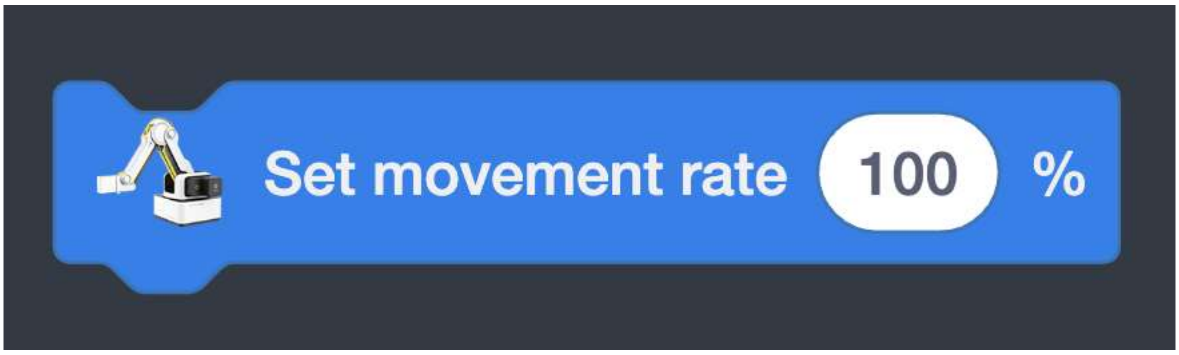
To control the movements of the robotic arm and operate it safely but efficiently, one must be able to use the appropriate command words to control the speed of the arm.

Examples of speed control command words using a Mitsubishi robot:

```
Command word - OVRD
Explanation - Designates the movement speed applied on the entire
program as a percentage (%) in respect
to the maximum speed.

Command word - JOVRD
Explanation - Designates the joint interpolation speed as a
percentage (%) in respect to the maximum speed.

Command word - SPD
Explanation - Designate the linear and circular interpolation
speed with the hand end speed (mm/s).
```

Example using DOBOT Magicial Lite

To edit the robot speed, you can only use a manual dial inside the software, or trough the programming recall the command:



Set movement rate 100 %

Example using COMAU eDo or racer 3

Several variables are available to modify speed, acceleration, and deceleration. The $GEN_OVR variable is managed through the dedicated buttons on the teach pendant. It can be immediately changed by the user without entering the program but cannot be modified through programming.

The following variables can be controlled by programs and allow for the modification of various parameters, as shown below:

```
$ARM_OVR --> General speed and acceleration – It has effects on
ALL programs launched later
$ARM_SPD_OVR --> General speed, not acceleration – It $ARM_ACC_OVR
or $ARM_DEC_OVR --> Only acceleration/deceleration
```

Therese variabiles effects on the executed program, and all the other executing (and future launched) programs.

You can do the same operations, but effecting only the specific program where the command is used, using the following

```
$PROG_SPD_OVR, $PROG_ACC_OVR, $PROG_DEC_OVR
```

Reading top down the previous commands you note that each one influences the previous (i.e. $ARM_OVR control the speed and the acceleration/deceleration of every program, so that effects the lower-level command $ARM_SPD_OVR that control only the speed, but not the acceleration).

The speed is expressed in percentual, so using a value from 0 to 100.

Usage example – Edit the acceleration only inside the specific selected program:

```
$PROG_ACC_OVR := 30
```

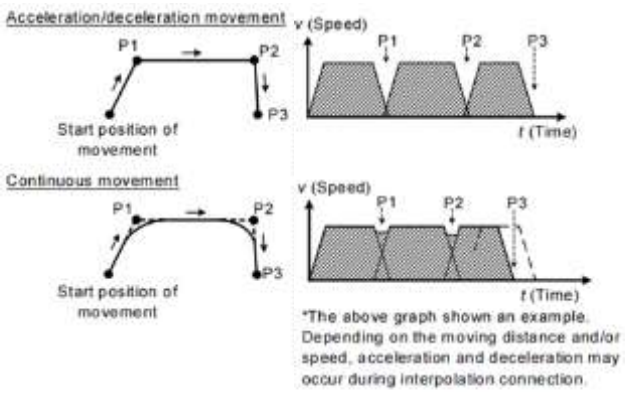
Does exist more specific commands, to edit specific different speeds (i.e. only in linear moves) but it goes out of the scope of this example. Please refer to official motion programming guide for further details.

Following a trajectory

Continuous Movement

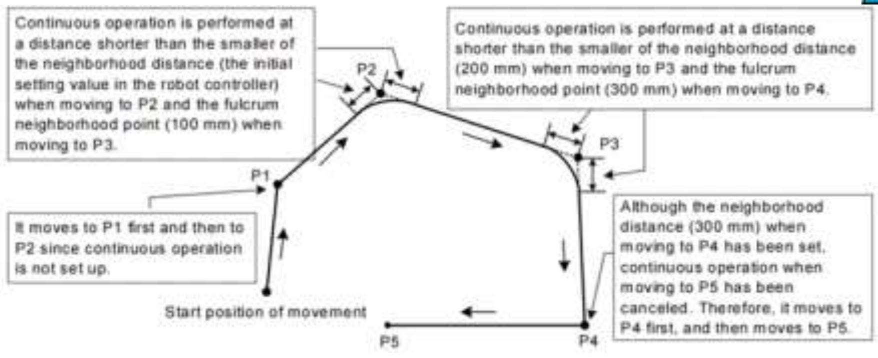
In all the previous cases if you add a list of movements, you are requesting the robot to stop in every ending point. However, does also exist the option to ask robot to links multiple movements together smoothly, so the robot doesn't stop between them. The robot plans its path to transition smoothly from one movement to the next, maintaining continuous motion. Continuous motion is managed very differently on different machines. For instance, the KUKA LBR iisy robot uses the SPLine approach, that keeps a constant speed during the travel, while the COMAU eDo uses a different approach and does not exactly pass through all the designated point.

Continuous movements are primarily used when very smooth path are required, when you want to reduce acceleration/deceleration or when you try to reduce the cycle time of an application.


Acceleration/deceleration movement
P1
P2
P3
Start position of movement
v (Speed)
t (Time)
Continuous movement
*The above graph shown an example. Depending on the moving distance and/or speed, acceleration and deceleration may occur during interpolation connection.

As shown, in the acceleration and deceleration operating mode, the speed is reduced in front of the target position. After moving to the target position, the speed for moving to the next target position starts to be accelerated. On the other hand, in the continuous operating mode, the speed is reduced in front of the target position, but it does not stop completely. The speed for moving to the next target position starts to be accelerated at that point. Therefore, it does not pass through each target position, but it passes through the neighborhood position.

Examples of command words to do a continuous movement using a Mitsubishi robot:

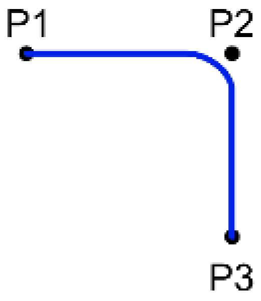

Continuous operation is performed at a distance shorter than the smaller of the neighborhood distance (the initial setting value in the robot controller) when moving to P2 and the fulcrum neighborhood point (100 mm) when moving to P3.
Continuous operation is performed at a distance shorter than the smaller of the neighborhood distance (200 mm) when moving to P3 and the fulcrum neighborhood point (300 mm) when moving to P4.
P2
P3
P1
It moves to P1 first and then to P2 since continuous operation is not set up.
Although the neighborhood distance (300 mm) when moving to P4 has been set, continuous operation when moving to P5 has been canceled. Therefore, it moves to P4 first, and then moves to P5.
Start position of movement
P5
P4

```
10 CNT 0 ' Invalidate CNT (Continuous movement).
20 MVS P1 ' Operate with acceleration/deceleration
30 CNT 1 ' Validate CNT (Continuous movement).
(Operate with continuous movement after this line.)
40 MVS P2 ' The connection with the next interpolation is
continuous movement.
50 CNT 1,100,200 ' Continuous operation specification at 100 mm on
the starting side and at 200 mm on the end side.
60 MVS P3 ' Continuous operation at a specified distance before
and after an interpolation.
70 CNT 1,300 ' Continuous operation specification at 300 mm on the
starting side and at 300 mm on the end side.
80 MOV P4 ' Continuous operation specification at 300 mm on the
starting side.
90 CNT 0 ' Invalidate CNT (Continuous movement).
100 MOV P5 ' Operate with acceleration/deceleration
```

Example of continuous motion using the COMAU eDo or racer 3 arm

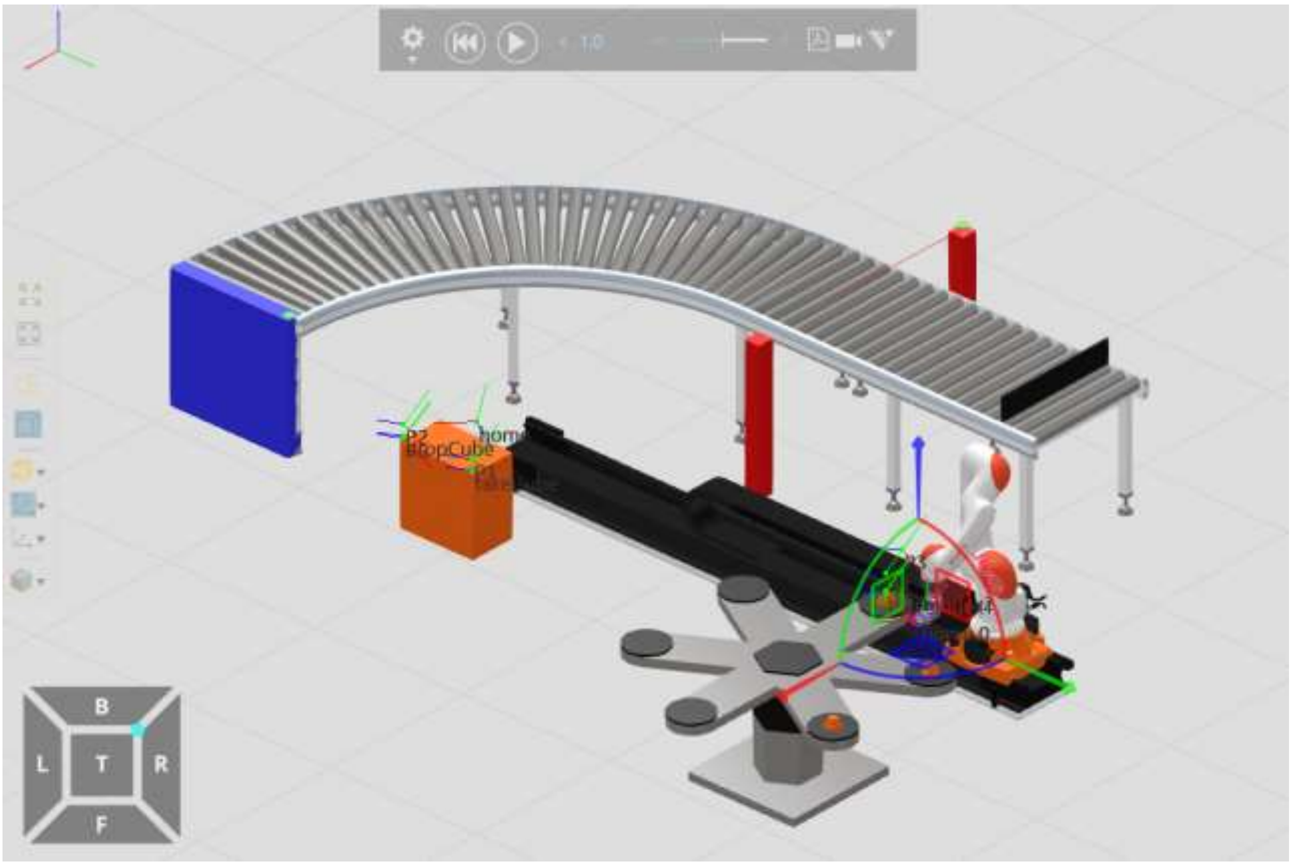
The eDo arm uses the MOVEFLY keyword to request a continuous motion, instead of the instruction MOVE. If MOVEFLY is followed by another movement, the arm will not stop at the first destination, but will move from the starting point of the first movement to the end point of the second movement without stopping at the point in common with the two movements. This also means that the point requested in the MOVEFLY action may not exactly be reached, but the TCP may only pass somewhere near it. MOVEFLY may be combined with a clause called ADVANCE, that will take in account up to the next 7 movements to upgrade performances.

```
MOVE TO P1
MOVEFLY TO P2 ADVANCE
MOVE TO P3
```



Inputs and outputs

When working with a robotic arm, it must be considered that almost never the robot stands for itself in the factory – it is always placed in the industrial surrounding, and it is constantly communicating with external devices, such as PLC’s, sensors, switches, motors, etc...

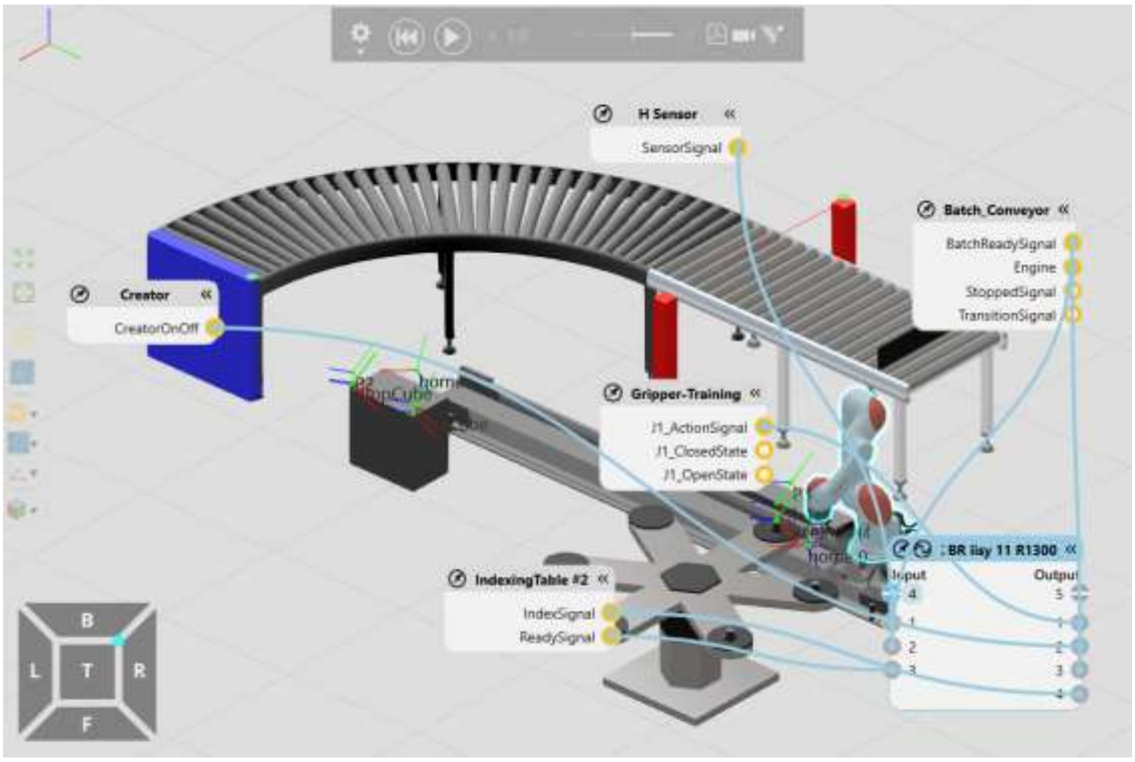


Example of a robotic arm in working situation

In the situation above, we can see that the robotic arm is placed within a production unit, equipped with a conveyer with an optical sensor for detection of objects. Robotic arm is equipped with a gripper, which places objects on the indexing table.

Before getting the arm to communicate with the environment, we must physically connect the arm with external components. In case of big production lines, communication between the robot and the components takes place via PLC. In smaller production units, inputs and outputs on the controller are enough for any task.

(1) Input signals
Signals can be retrieved from an external device, such as a programmable logic controller, or a sensor.
(2) Output signals
Signals can be output to an external device, such as a programmable logic controller, motor, a LED or sound indicator.



Example of connections between the arm and external components

Statement examples for automating inputs and outputs for KUKA robot:

Set OUT[2] == False

Set OUT[4] == True

Wait IN[1] == True

Statement examples for automating inputs and outputs for Mitsubishi robot:

```
WAIT M_IN(1)=1          Waits for the input signal bit 1 to turn ON.
M1=M_INB(20             Substitutes the input signal bit 20 to 27, as an 8-bit state, in numeric
variable M1.
M_OUT(1)=1              Turns the output signal bit 1 ON.
M_OUTB (8)=0            Turns the 8 bits, from output signal bit 8 to 15, OFF.
M_OUTW (20)=0           Turns the 16 bits, from output signal bit 20 to 35, OFF.
M_OUT(1)=1 DLY 0.5      Turns the output signal bit 1 ON for 0.5 seconds. (Pulse output)
```

Statement example for waiting a specific signal using the COMAU eDo or racer 3

WAIT FOR $DIN[1] = 1 // Wait until the digital input 1 is activated

$DOUT[21]:=OFF // Deactivate the digital output number 21

Sorting out objects

Sorting out black objects – Mitsubishi robot example

Task conditions: after the production process is completed, it is necessary to sort the workpieces, in such a way that all black objects are stored in one, and all other objects in another receiving warehouse. A robotic arm with a suitable control controller and a suitable sensor system is used for sorting objects.

General view of the robotic workstation:

In this workcell housings are sorted into stacks by a Mitsubishi RV-2AJ industrial robot (object name RV-E2). Housings can differ concerning material and colour and are sorted as follows:

Stack	**Color**	**Material**
1	black	plastics
2	Red, silver	Plastics, metal

We will determine the colour of the object using an optical diffuse proximity switch placed on the robotic gripper. Keep in mind that diffuse optical switches will detect all objects, with one difference – the black objects will be detected on significantly smaller distance. The reason for this is the fact that the black object absorbs most of the light transmitted from the sensor. Due to this reason, detection position (P6 in the given example) is to be chosen in such way that robotic gripper optical switch detects all but black objects:

```
IF M_IN(9) = 1 THEN GOTO 240 ELSE GOTO 380    /   if the condition
is true, object is non-black, else it is black
```

Know-how

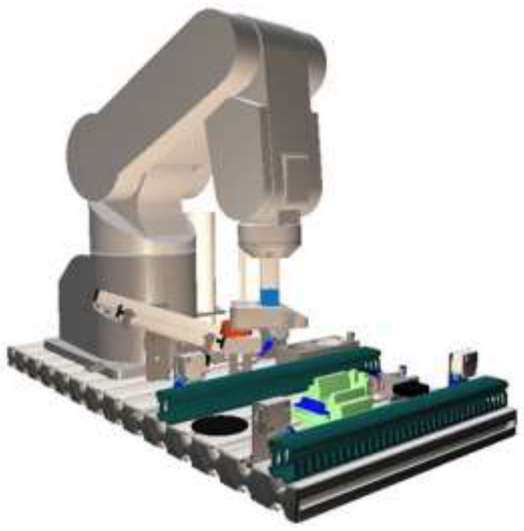
The workcell shows you how to solve the following tasks:

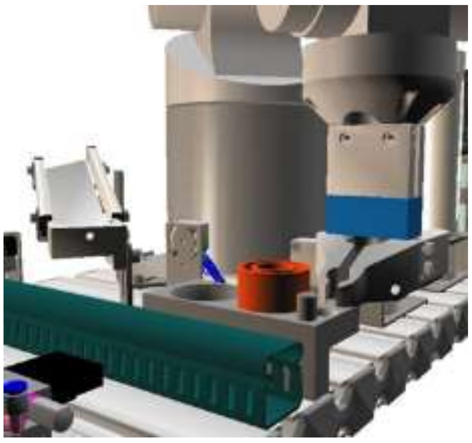
Exactly teaching positions
Using sensors to identify parts
Branches and robot programs

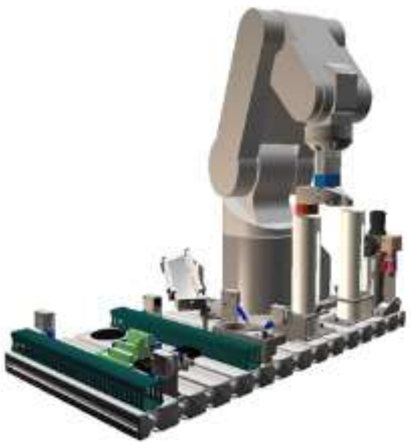
The appropriate **teaching points** that should be used in the program are listed in the following table:

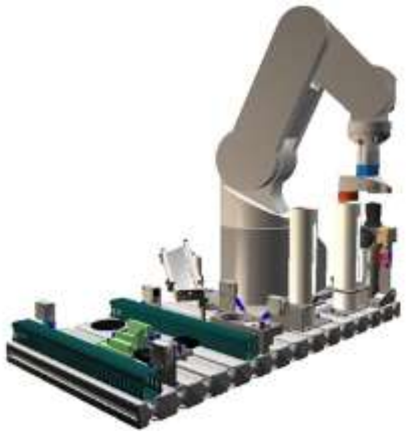
P1	Position above slide storage
P2	To grasp an object within slide storage

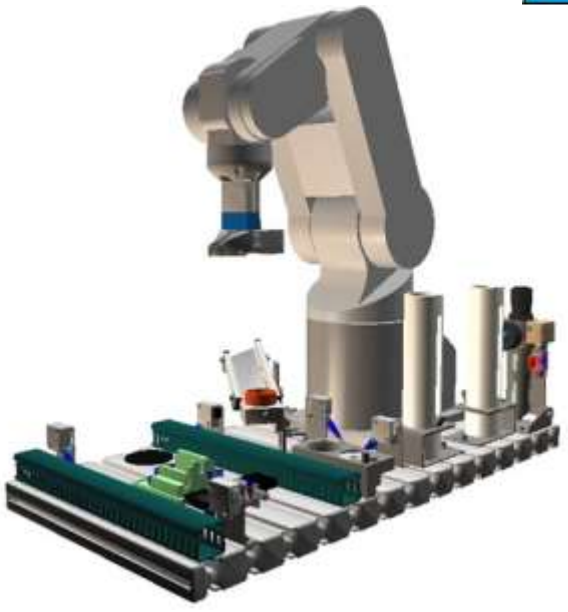
P3	Position above stack
P4	To leave an object in the stack for detection
P5	Position above colour detection
P6	To detect a colour of the object
P7	Position above black objects magazine
P8	Position above non-black objects magazine
P9	Neutral position


Position P2: Pick-up workpiece


Position P6: Identification (colour of workpiece)


Position P7: Place position black workpiece


Position P8: Place position red/metallic workpiece



Position P9: Neutral point

Programming

Main program

Program steps	Comment
Initialization	Override is reduced to 40%, the program waits for the object to be placed into the slide storage, robotic arm is placed into neutral point
Pick – and - place	Neutral point – positioning – grasping - detection
Read optical sensor, Branch	Does the optical sensor see the object? • If no, the object is black • If yes, the object is non - black
Sort	Black object is being placed into appropriate magazine, while non – black objects are placed in the second magazine

Step by step program solution

```
10 OVRD                  / reduce speed to 40% of maximum
20 WAIT M_IN(8) = 1      / wait for confirmation from the optical
sensor from the receiving magazine that the workpiece is present
30 MOV P9                / deflection to the neutral starting point
40 HOPEN 1               / hand opening
50 OVRD 80               / reduce speed to 80% of maximum
60 MOV P1                / positioning above the object
70 OVRD 15               / reduce speed to 15% of maximum
80 MVS P2                / straight descent into the position for
grasping the object
90 HCLOSE 1              / closing the hand and grasping the object
```

```
100 MOV P1              / closing the hand and grasping the object
110 OVRD 80
120 MOV P3              / Deflection to a color detection neutral
point
130 OVRD 15
140 MOV P4              / lowering the object to the place where
the color detection will be performed
150 HOPEN 1
160 OVRD 80
170 MVS P3              / straight line return to the neutral
point for detection
180 HCLOSE 1
190 MVS P5              / deflection to the detection position
200 OVRD 30
210 MVS P6              / straight descent into detection position
220 DLY 2               / mandatory retention in the detection
position
230 IF M_IN(9) = 1 THEN GOTO 240 ELSE GOTO 380        / condition
- does the optical sensor see the object?
240 OVRD 80             / the object has been detected
250 MVS P3
260 HOPEN 1
270 OVRD 15
280 MVS P4
290 HCLOSE 1
300 MVS P3
310 OVRD 80
320 MOV P9
330 MVS P8              /position above the magazine with non-
black items
340 ACCEL 30,30
350 HOPEN 1
360 GOTO 10
370 ENDIF
```

Palletization and depalletization

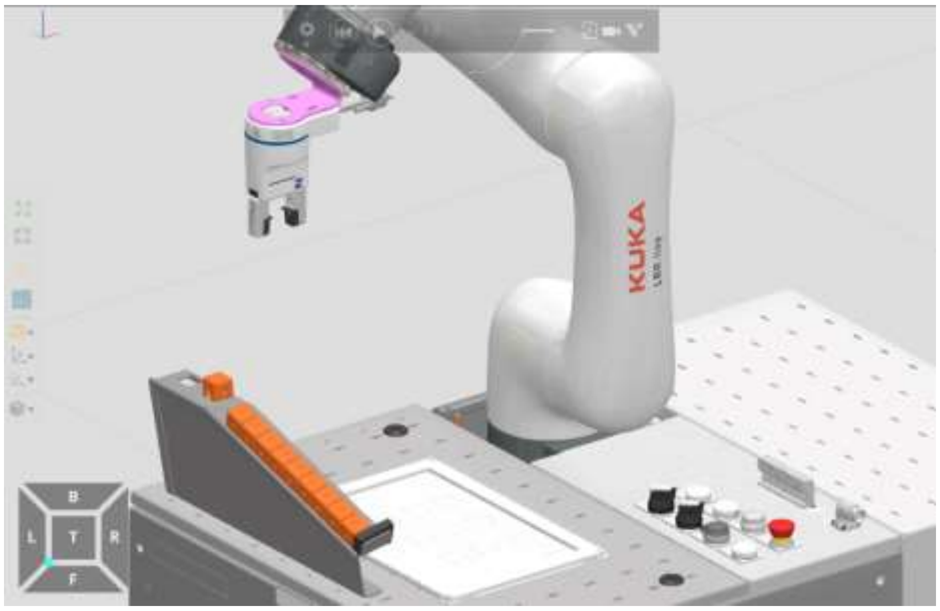
Robotic palletizing is the practice of using an industrial robot to place and stack goods onto a pallet for transportation. Many robots can be used for palletizing and the most common approach is to use dedicated palletizing robots for quick transportation of heavy, bulky items. All the modern robotic arm brands support the palletization process, but the technologies for achieving this task can vary significantly.

Why is the palletization programming so important?

The simplest way to answer this question would be, in fact, to explain the process of placing objects on the pallet with, let's say 100 places. Without palletization, we would have to manually insert the coordinates of all the positions on the pallet; we would also need to manually write the codes that would serve for placing objects on all those 100 positions, which would make the code ridiculously long, and the possibility for mistake would be huge. So, we use the palletization to shorten the program code, and preparation time, and make the process more efficient and accurate.

On the following pages we will present examples of palletization on the KUKA arm using the *while* program loop, and Mitsubishi arm, using the built – in PLT function.

Paletization of objects using KUKA LBR iisy robotic arm – Simulated in KUKA Sim

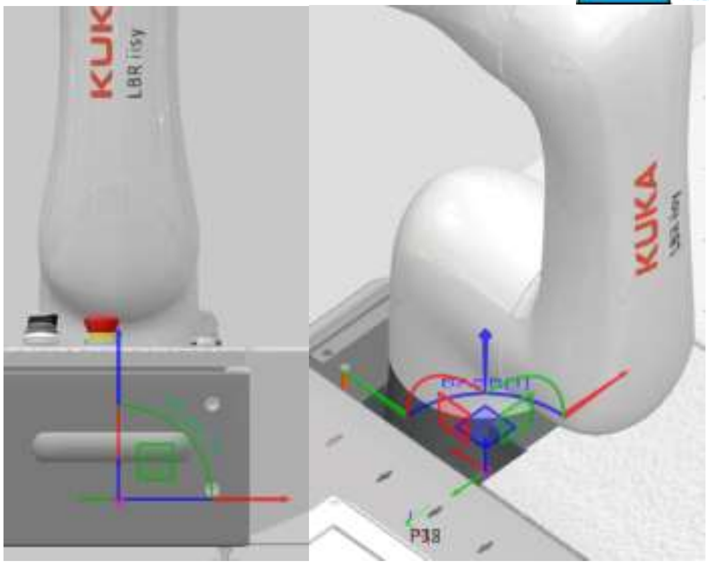


Problem description - 9 cubes, palet size 3x3, horizontal distance between objects is 60mm, vertical distance is 40mm in the parent coordinate system.

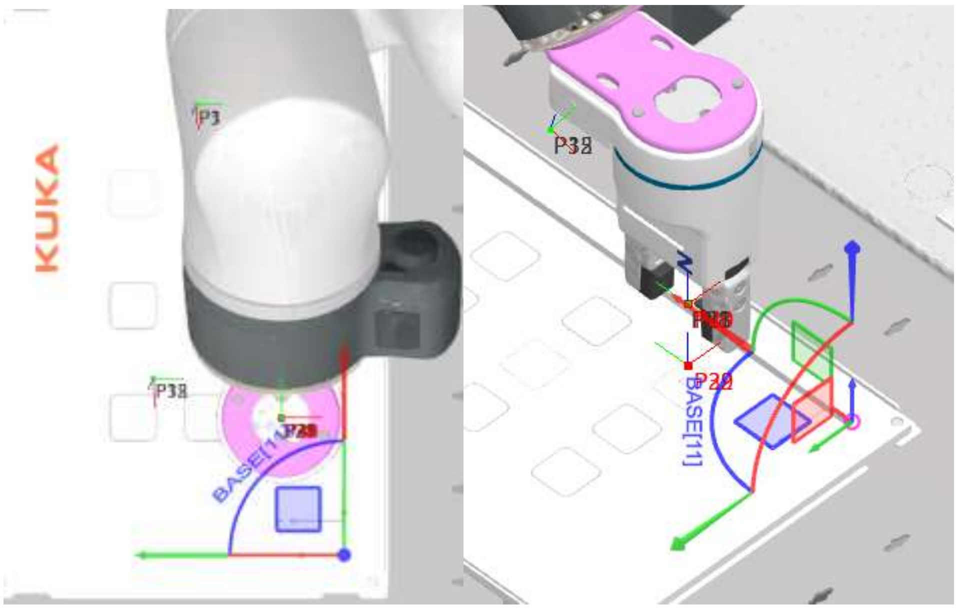
Problem solution – each position on the pallet is obtained using the base coordinate system shifting in each program iteration, using the *while* program loop. Base shifting is equal to distance between places on the pallet.

Base coordinate systems initial settings:

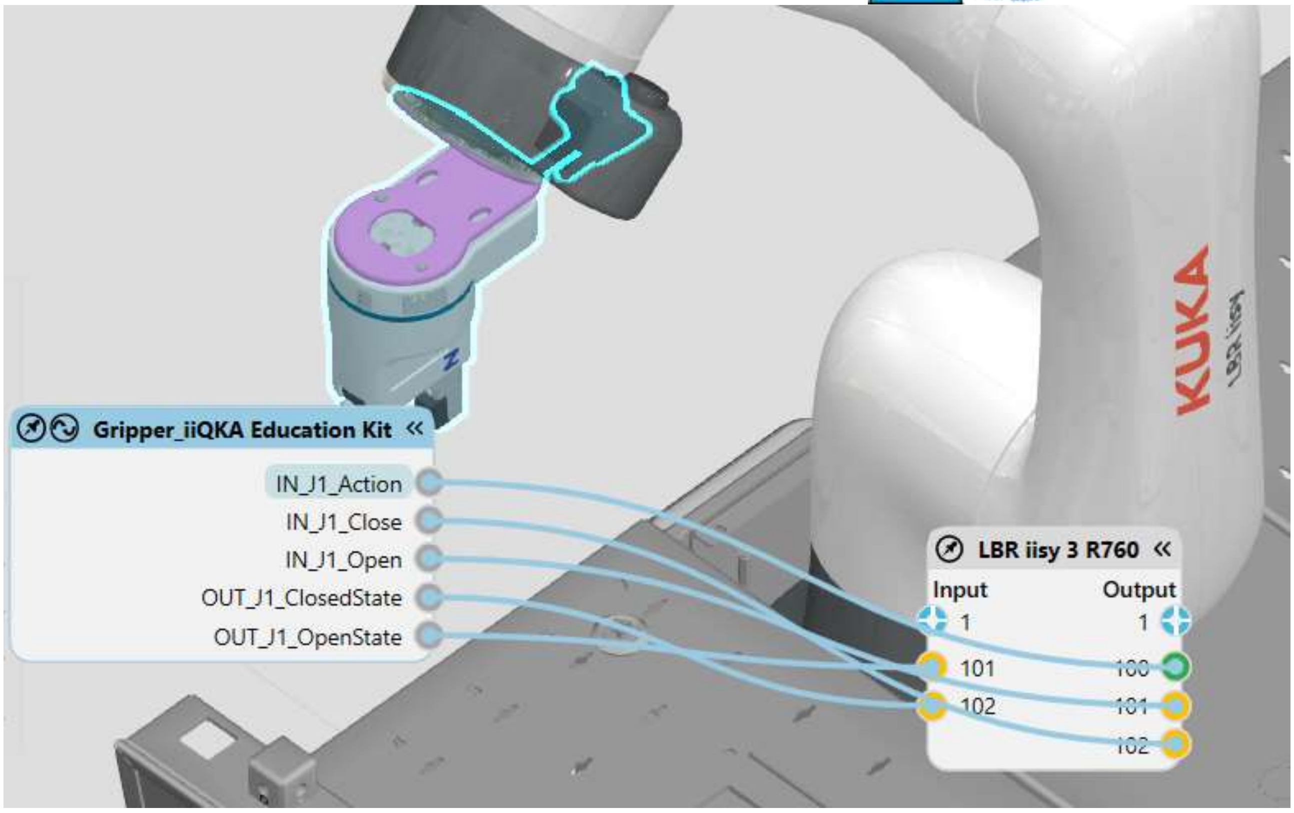
BASE_DATA[1] (world coordinates)



BASE_DATA[11] (parent coordinates)



I/O signal map:

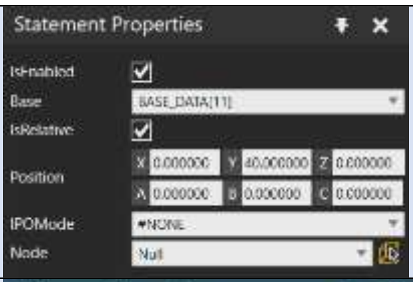


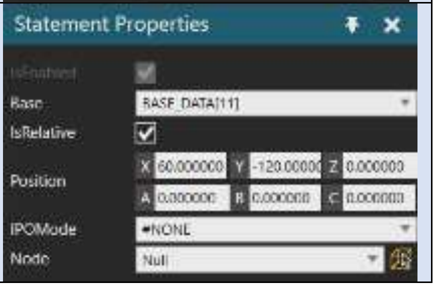
Main program body:

```
PTP P7 TOOL_DATA[1] BASE_DATA[1] 100%
Call iiQKA_Put_Pattern
Delay 2s
Call iiQKA_Rem_Pattern
```

iiQKA_Put_Pattern

```
Assign Int_1 = 1
Assign Int_2 = 1
While Int_2<=3:
     While Int_1<=3:
          Call iiQKA_Get_Cube
          PTP P21 TOOL_DATA[11] BASE_DATA[11] 100%
          #  ----- Pre position -----
          LIN P22 TOOL_DATA[11] BASE_DATA[11] 2000mm/s
          #  ----- Release position -----
          Set OUT[12] == False
          Set OUT[100] == False
          Delay 1s
          #  ----- Release part and open gripper -----
          LIN P23 TOOL_DATA[11] BASE_DATA[11] 2000mm/s
          #  ----- Linear motion back to pre position -----
```

`SetBase BASE_DATA[11]`
`Assign Int_1 = Int_1+1`
`#  ----- Base-Shift, variable upcounting and Call iiQKA_GetCube -----`
	 Statement Properties
	IsEnabled ✓
	Base BASE_DATA[11]
	IsRelative ✓
	Position X 0.000000 Y 40.000000 Z 0.000000 A 0.000000 B 0.000000 C 0.000000
	IPOMode ◆NONE
	Node Null

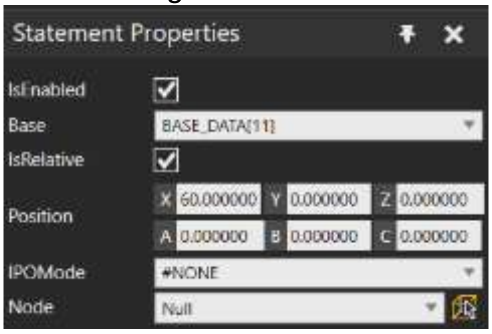
`SetBase BASE_DATA[11]`
`Assign Int_2 = Int_2+1`
`Assign Int_1 = 1`
`#  ----- Base-Shift, variable upcounting and reset variable -----`
	 Statement Properties
	IsEnabled ✓
	Base BASE_DATA[11]
	IsRelative ✓
	Position X 60.000000 Y -120.00000 Z 0.000000 A 0.000000 B 0.000000 C 0.000000
	IPOMode ◆NONE
	Node Null

Comment – in each iteration of the while loop of the Int_1, the base coordinate system BASE DATA11 shifts 40.00mm in the Y direction. For this reason, the three objects are being placed in the Y+ direction, starting from the bottom position, forming the first column. Anytime the variable Int_2 increases in the second while loop, the BASE DATA11 should be shifted 60mm in the X direction and -120mm in the Y direction to be placed exactly on the bottom position in the next column. The process is being repeated 3 times, because there are 3 columns on the pallet.

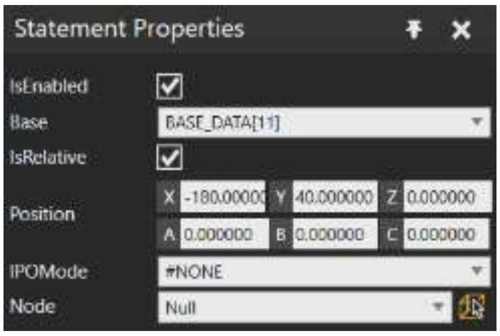
```
     iiQKA_Get_Cube
               PTP P15 TOOL_DATA[11] BASE_DATA[1] 100%
               #  ----- Start point get cube -----
               PTP P16 TOOL_DATA[11] BASE_DATA[1] 100%
               #  ----- Pre position -----
               LIN P17 TOOL_DATA[11] BASE_DATA[1] 2000mm/s
               #  ----- Pick position -----
               Set OUT[12] == True
               Set OUT[100] == True
               Delay 1s
               #  ----- Pick part and close gripper -----
               LIN P19 TOOL_DATA[11] BASE_DATA[1] 2000mm/s
               #  ----- Linear motion back to pre position -----
               PTP P32 TOOL_DATA[11] BASE_DATA[1] 100%
               #  ----- PTP motion back to start point get cube -
               ----

iiQKA_Rem_Pattern

     PTP P48 TOOL_DATA[1] BASE_DATA[1] 100%
     #  ----- Start point put pattern -----
```

```
    SetBase BASE_DATA[11]
    Assign Int_3 = 1
    Assign Int_4 = 1
    While Int_4<=3:
        While Int_3<=3:
            PTP P38 TOOL_DATA[11] BASE_DATA[11] 100%
            #  ----- Pre position -----
            LIN P39 TOOL_DATA[11] BASE_DATA[11] 2000mm/s
            #  ----- Pick position -----
            Set OUT[12] == True
            Set OUT[100] == True
            Delay 1s
            #  ----- Pick part and open gripper -----
            LIN P40 TOOL_DATA[11] BASE_DATA[11] 2000mm/s
            #  ----- Linear motion back to pre position -----
            SetBase BASE_DATA[11]
            Assign Int_3 = Int_3+1
            Call iiQKA_Lay_Cube
            #  ----- Base-Shift, variable upcounting and Call
            iiQKA_GetCube -----
        SetBase BASE_DATA[11]
        Assign Int_4 = Int_4+1
        Assign Int_3 = 1
        #  ----- Base-Shift, variable upcounting and reset
        variable -----
```

BASE DATA11 shiftings:


Statement Properties

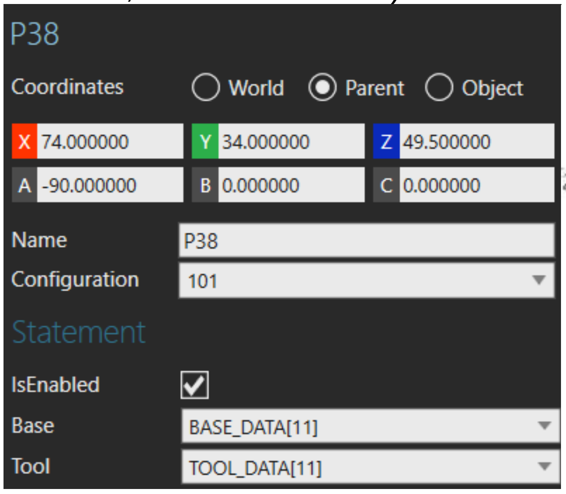
IsEnabled	✓
Base	BASE_DATA[11]
IsRelative	✓
Position	X 60.000000 Y 0.000000 Z 0.000000
	A 0.000000 B 0.000000 C 0.000000
IPOMode	#NONE
Node	Null


Statement Properties

IsEnabled	✓
Base	BASE_DATA[11]
IsRelative	✓
Position	X -180.00000 Y 40.000000 Z 0.000000
	A 0.000000 B 0.000000 C 0.000000
IPOMode	#NONE
Node	Null

Comment – as it is shown on the images above, the first base shifting is 60mm in the X direction, which means that the objects are being depalletized starting from the bottom row on the pallet (unlike the palletization, which was carried out by placing objects in columns). The second base shifting is –180mm in the X direction and 40mm in the Y direction, which is in fact the distance to the first position in the next row on the pallet.

Comment – please note that the parent coordinates of the positions on the pallet are referred to BASE DATA11 coordinate system, mentioned in the code - PTP P38 TOOL_DATA[11] BASE_DATA[11] 100%. Parent coordinates are in fact distances from the positions on the pallet to the center of the coordinate system BASE DATA11, as shown in the images below (X distance – 74mm, Y distance 34mm):



P38

Coordinates ( ) World [✓ Parent] ( ) Object

X 74.000000 Y 34.000000 Z 49.500000

A -90.000000 B 0.000000 C 0.000000

Name P38

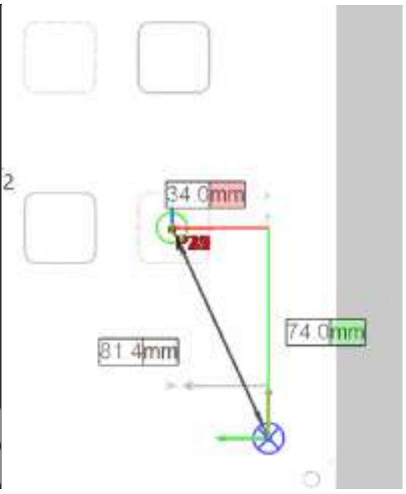
Configuration 101

Statement

IsEnabled ✓

Base BASE_DATA[11]

Tool TOOL_DATA[11]



```
     iiQKA_Lay_Cube
          PTP P1 TOOL_DATA[11] BASE_DATA[1] 100%
          #  ----- Start point lay cube -----
          PTP P24 TOOL_DATA[11] BASE_DATA[1] 100%
          #  ----- Pre position -----
          LIN P33 TOOL_DATA[11] BASE_DATA[1] 2000mm/s
          #  ----- Release position -----
          Set OUT[12] == False
          Set OUT[100] == False
          Delay 1s
          #  ----- Release part and open gripper -----
          PTP P34 TOOL_DATA[11] BASE_DATA[1] 100%
          #  ----- Linear motion back to pre position -----
          LIN P35 TOOL_DATA[11] BASE_DATA[1] 2000mm/s
          PTP P3 TOOL_DATA[11] BASE_DATA[1] 100%
          #  ----- PTP motion back to start point get cube -
          ----
```

Palletisation of objects using Mitsubishi RV-2AJ robotic arm

As previously mentioned, the Mitsubishi robotic arm can perform a pallet operation, using Melfa Basic IV as the main programming language, that contains the PLT instruction especially built-in for this purpose. The PLT instruction allows to sort out 3 different types of pallets – two rectangular pallets (zigzag and same direction), and radial pallet.

Command word - PLT

Explanation - Obtains the designated position on the pallet with operations.

[Format]
PLT[]<Pallet No.> , <Grid No.>

[Terminology]
<Pallet No.> Select a pallet No. between 1 and 8 that has already been defined with a DEF PLT command. Specify this argument using a constant or a variable.
<Grid No.> The position number to calculate in the palette. Specify this argument using a constant or a variable.

Statement example Explanation

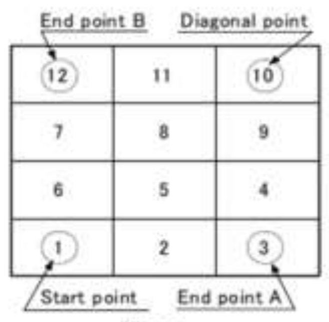
DEF PLT 1, P1, P2, P3, P4, 4, 3, 1 .... Defines to operate pallet No. 1 with a start point = P1, end point A = P2, end point B = P3 and diagonal point = P4, a total of 12 work positions (quantity A = 4, quantity B = 3), and a pallet pattern = 1(Zigzag).

DEF PLT 2, P1, P2, P3, , 8, 5, 2................................ Defines to operate pallet No. 2 with a start point = P1, end point A = P2, and end point B = P3, a total of 40 work positions (quantity A = 8, quantity B = 5), and a pallet pattern = 2 (Same direction).

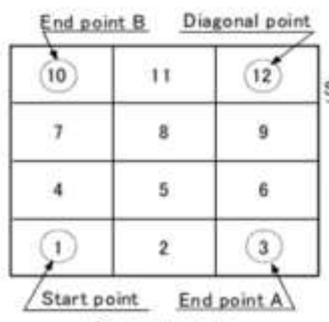
DEF PLT 3, P1, P2, P3, , 5, 1, 3............................. Define that pallet No. 3 is an arc pallet having give five work positions on an arc designated with start point = P1, transit point = P2, end point = P3 (total three points).

(PLT1, 5) .... Operate the 5th position on pallet No. 1.

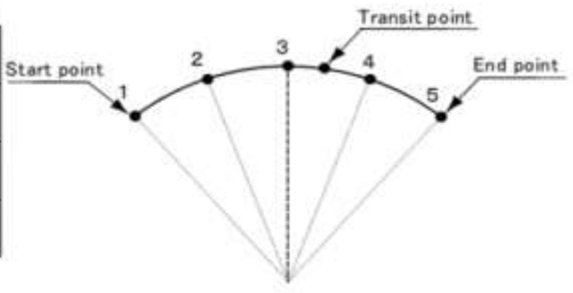
(PLT1, M1) ....Operate position in pallet No. 1 indicated with the numeric variable M1.



Zigzag
Pallet pattern = 1 (zigzag)



Same direction
Pallet pattern = 2 (same direction)



Arc pallet
Pallet pattern = 3 (arc pallet)

```
[Reference Program]
100 DEF PLT 1,P1,P2,P3,P4,4,3,1' The definition of the four-point
pallet. (P1,P2,P3,P4)
110 '
```

```
120 M1=1 ' Initialize the counter M1.
130 *LOOP
140 MOV PICK, 50 ' Moves 50 mm above the work unload position.
150 OVRD 50
160 MVS PICK
170 HCLOSE 1 ' Close the hand.
180 DLY 0.5 ' Wait for the hand to close securely (0.5 sec.)
190 OVRD 100
200 MVS,50 ' Moves 50 mm above the current position.
210 PLACE = PLT 1, M1 ' Calculates the M1th position
220 MOV PLACE, 50 ' Moves 50 mm above the pallet top mount
position.
230 OVRD 50
240 MVS PLACE
250 HOPEN 1 ' Open the hand.
260 DLY 0.5
270 OVRD 100
280 MVS,50 ' Moves 50 mm above the current position.
290 M1=M1+1 ' Add the counter.
300 IF M1 <=12 THEN *LOOP ' If the counter is within the limits,
repeats from *LOOP.
310 MOV PICK,50
320 END
```

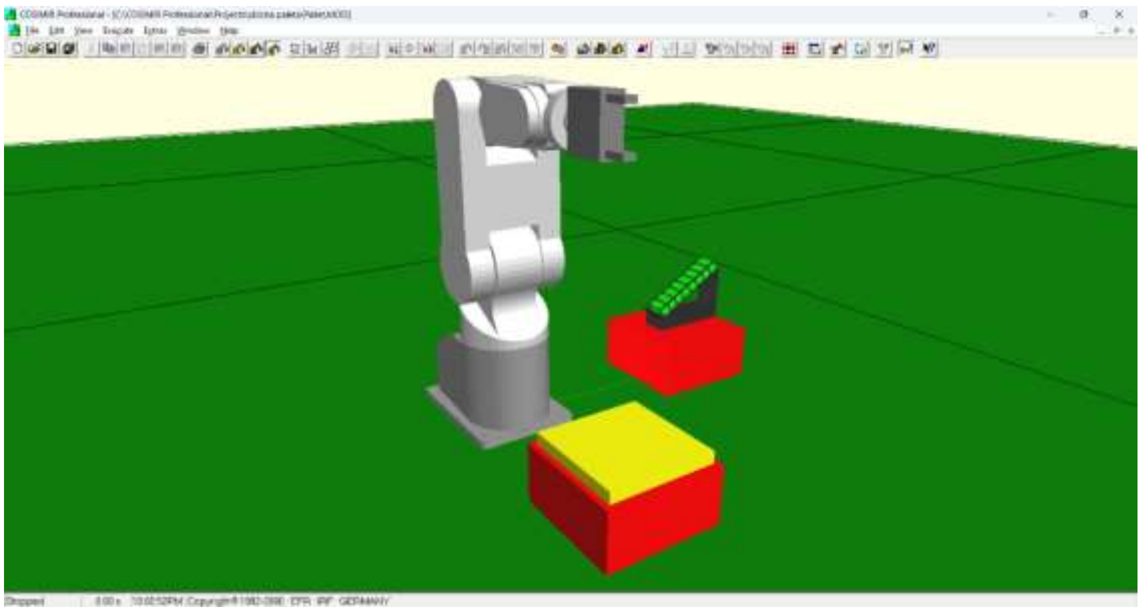
Crucial steps, when writing a palletization program in Melfa Basic IV:

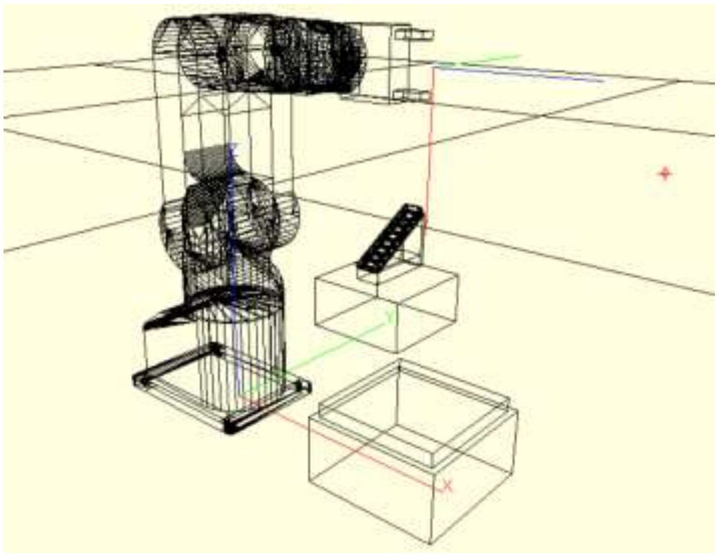
- Define the pallet – DEF PLT instruction;
- Initialize the counter variable – type integer, and the value of the variable will correspond to the positions on the pallet later;
- Insert the *LOOP label. The counter will be incremented in each iteration of the program. The initial value of the counter variable should be outside of the loop;
- Obtain the positions on the pallet using the PLT1, M1 line – the program calculates the positions on the pallet using the 4 corner pallet points initially pre-defined;
- Incriminate the counter – if the start value of the counter is, for instance, 1, and the counter is being incriminated for value of 1 in each program iteration, the objects will be placed on all the positions on the pallet; if, however, the initial value of the counter is 2, and the counter is incriminated for 2, we will have the objects only on 2nd, 4th, 6th, etc etc place on the pallet;
- Examine if the counter value is within the limits – number of places on the pallet. In this case the IF THEN loop has been used, but generally there are no restrictions in this context, and we can also use the FOR NEXT, WHILE, or any other cycle appropriate.

Example of palettization process using the Cosimir Professional software:

- RV-2AJ robotic arm
- Rectangular pallet type 2 (same direction)

- 2 columns, 4 rows, 8 objects in total


Environmental setup


World and the tool centre point (TCP) coordinate system

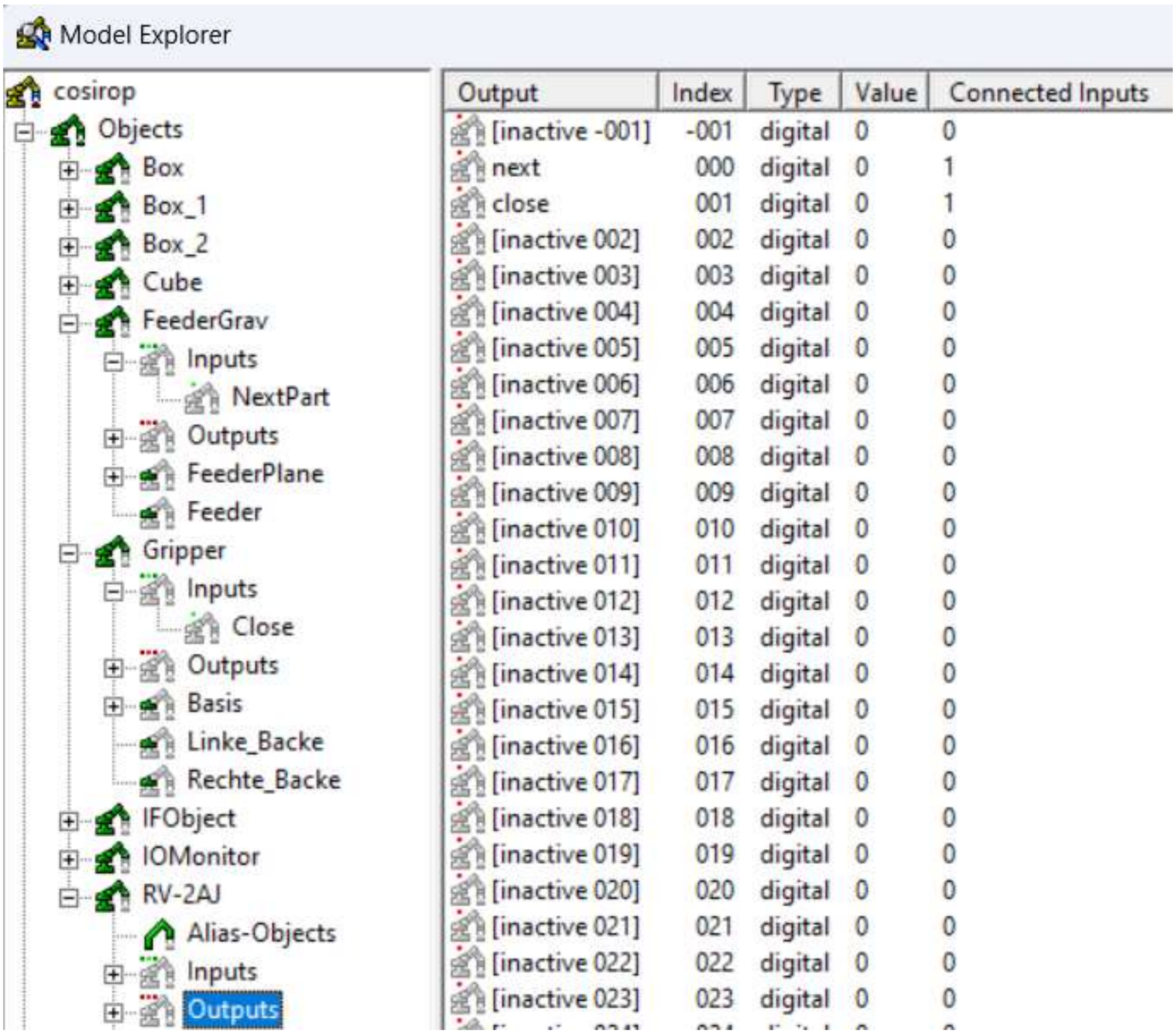
Program points coordinates:

No.	Position	Orientation	Comment
P1	0.73, 361.4, 126.6	-166, 149, R, A	*Get cube*
P10	255, 165.1,283.5	-31, 160, R, A	*Neutral point*
P2	300, 50, 110	-90, 180, R, A	*Pallet – lower left*
P3	300, -50, 110	-90, 180, R, A	*Pallet – lower right*
P4	400, 50, 110	-90, 180, R, A	*Pallet – upper left*

P5	400, -50, 110	-90, 180, R, A	*Pallet – upper right*

By analysing the pallet corner points coordinates, we can calculate that the horizontal and the vertical distance between pallet corner points is 100mm. The pallet is, in fact a square, with 100mm's in length. In this pallet we will place the 8 objects, and sort them out in 2 columns and 4 rows. It means that the horizontal distance will be 100mm, while the vertical distance will be exactly 33.33mm (please note that the 4 objects in the column will be placed on the following points – 0mm, 33.33mm, 66.66mm, 100mm).

I/O signal map:



Output	Index	Type	Value	Connected Inputs
[inactive -001]	-001	digital	0	0
next	000	digital	0	1
close	001	digital	0	1
[inactive 002]	002	digital	0	0
[inactive 003]	003	digital	0	0
[inactive 004]	004	digital	0	0
[inactive 005]	005	digital	0	0
[inactive 006]	006	digital	0	0
[inactive 007]	007	digital	0	0
[inactive 008]	008	digital	0	0
[inactive 009]	009	digital	0	0
[inactive 010]	010	digital	0	0
[inactive 011]	011	digital	0	0
[inactive 012]	012	digital	0	0
[inactive 013]	013	digital	0	0
[inactive 014]	014	digital	0	0
[inactive 015]	015	digital	0	0
[inactive 016]	016	digital	0	0
[inactive 017]	017	digital	0	0
[inactive 018]	018	digital	0	0
[inactive 019]	019	digital	0	0
[inactive 020]	020	digital	0	0
[inactive 021]	021	digital	0	0
[inactive 022]	022	digital	0	0
[inactive 023]	023	digital	0	0

We see that the output 0 of the RV-2AJ arm is connected to the "NextPart" Input of the Gravity Feeder object. The output 1 of the RV-2AJ is connected to the "close" Input of the Gripper.

Proposed program solution:

```
10 ovrd 30                    //reduce global speed of the arm on
30 percent
20 spd 30                //reduce linear speed to 30mm/s
30 def plt 1,p2,p3,p4,p5,2,4,2     //define the rectangular pallet
type 2, 2 columns 4 rows
40 def pos p20           //define variable position on the pallet
50 m_out(1)=0
60 m1=1                  //initialize the counter
70 *loop                 //beginning of the loop
80 mov p10
90 mov p1, -50           //get cube
100 mvs p1
110 m_out(1)=1
120 dly 1
130 mvs p1, -50
140 m_out(0)=1
150 mov p10
160 m_out(0)=0
170 p20 = plt 1,m1            //variable pallet position is being
calculated
180 mov p20, -30              //place the cube to the appropriate
pallet position
190 mvs p20
200 m_out(1)=0
210 dly 1
220 mvs p20, -30
230 m1=m1+1              //incriminate the counter – change the
object position
240 if m1<=8 then *loop else goto 250   //if the number of objects
being placed is less then                               8, go
back to the loop
250 mov p10
260 end
```

CHAPTER 4

In this last chapter we will propose some more advanced examples of industrial robots, and we will cover a few concepts about something not strictly related to robot progamming, but that can be used while creating a robotic field.

Due to its great balance of capabilities and its availability "out-of-the-box", and also considering our chance in robot aviaability for testing purposes we are proposing in this section only two examples, using the robot DOBOT Magician Lite, but keep in mind that – with correct corresponding hardware, and with necessary software adaptations – the same applications may be reproduced using other devices.

Use of a conveyor belt

In industrial robotics, conveyor belts are used to transport materials or products from one point to another within a manufacturing or assembly process. They are essential for automating the movement of items, enhancing efficiency, and reducing manual labour.

They are ideal in environments where large quantities of products are being manufactured, assembled, or packaged.
Conveyor belt may be used when items need to be sorted or handled automatically and especially when a continuous and steady flow of materials is required.

Industrial robots' arms play a crucial role in cells where conveyor belts are used. They can pick items from the conveyor belt and place them in specific locations for further processing or packaging or perform precise assembly tasks on products moving along the conveyor or load raw materials onto the conveyor or unload finished products from it.

On the other and arms may operate Inspection and Quality Control: with sensors and cameras, they can inspect products for defects and remove faulty items from the line.

As seen in the previous chapters their multiple degrees of freedom allow them to handle a variety of tasks with precision and adaptability.

Overall, the integration of conveyor belts with industrial anthropomorphic robots leads to highly efficient, precise, and scalable automation solutions in manufacturing and assembly lines.

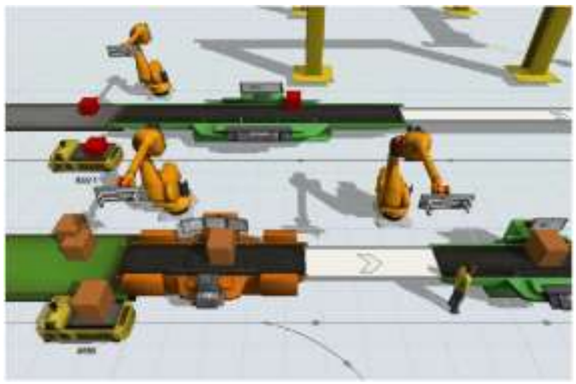


Image (C) Ján Duplák et al, CC BY-NC 4.0 License

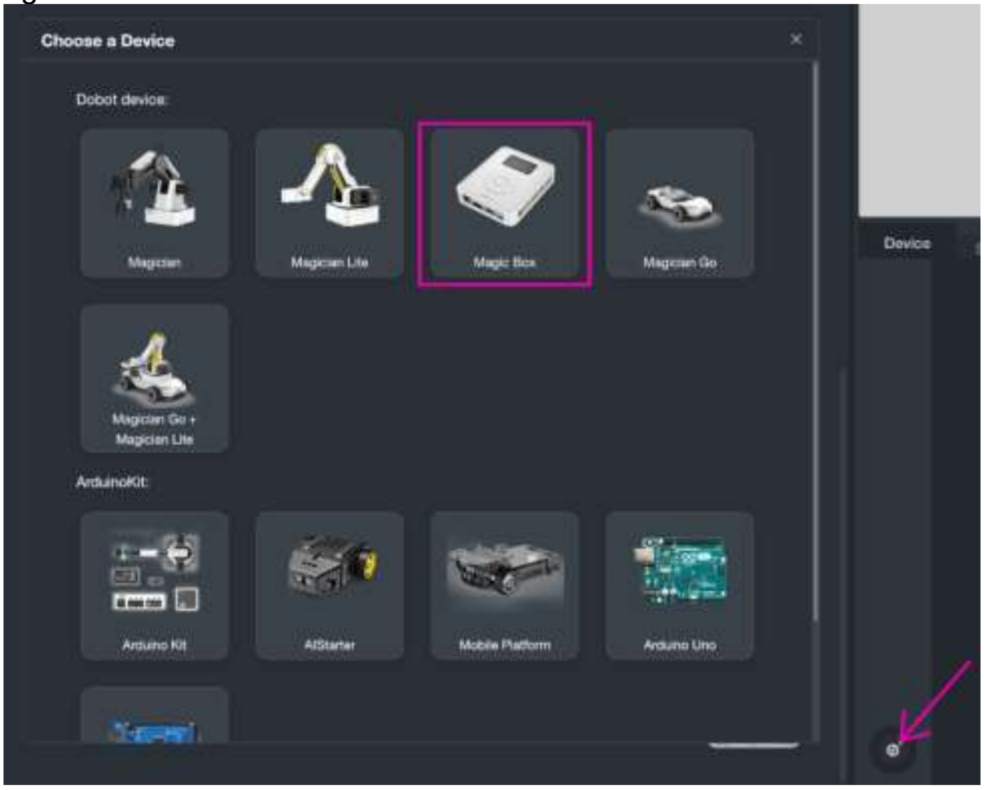
To use a conveyor belt a few steps must be considered, but some of the most relevant are:

- Define the layout, including the start and end points, turns, and intersections.
- Selection of Conveyor Type: Choose based on the weight, size, and nature of the items being transported (e.g., belt, roller, chain).
- Find the way of interaction with robots (ie code to manage the conveyor's speed and direction with external programmable controllers)
- Synchronization: Ensure the conveyor is synchronized with robotic operations (ie timing correctly, or using sensors and control systems to coordinate movements)

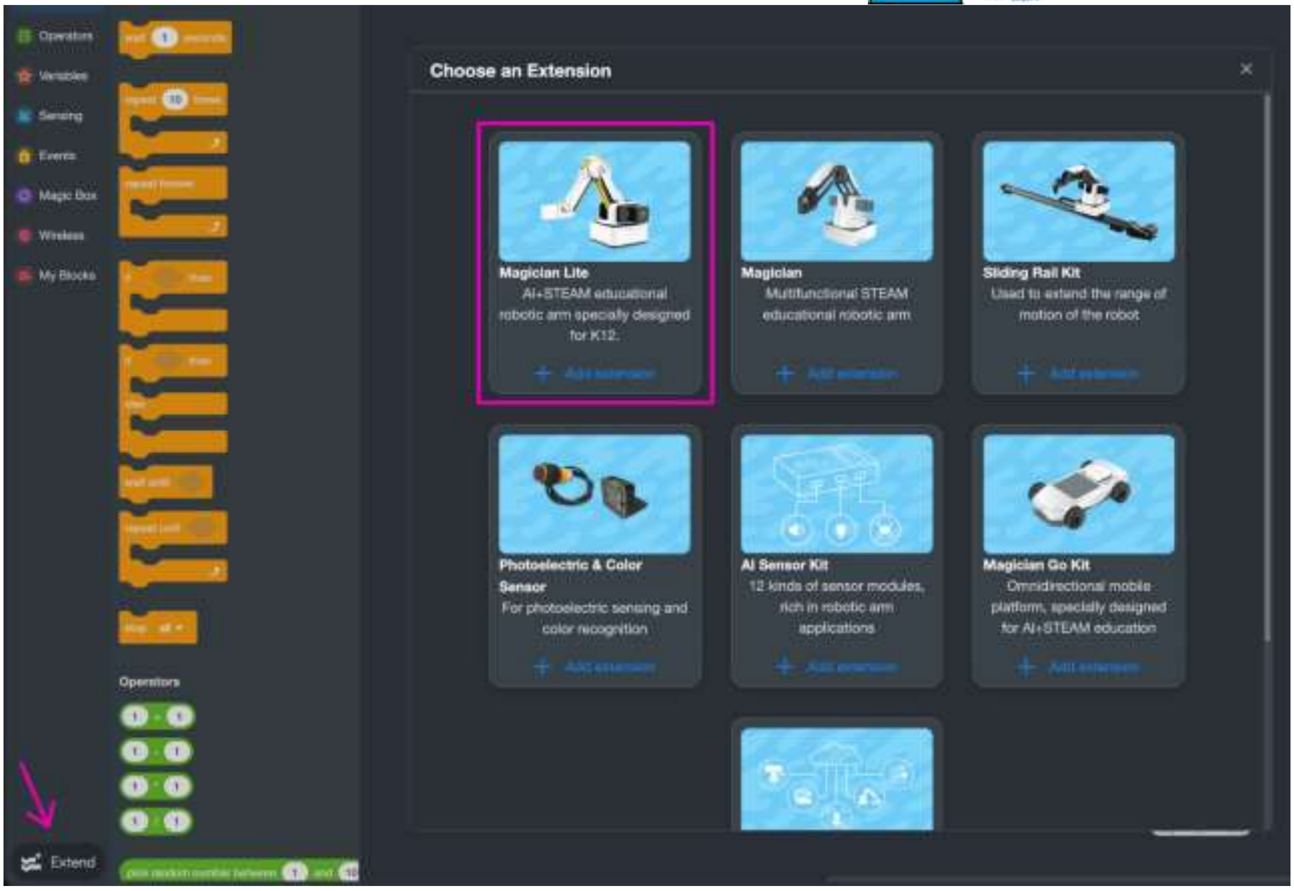
DOBOT Magician Lite robot can be easily connected to a conveyor belt, and a specific solution for DOBOT robotics does exist. The robot may be connected to an external controller, called Magic Box. It acts as a sort of PLC, so an external programmable device, that may communicate with different devices. In that case the Magic Box will be both connected to the arm, and the conveyor belt, and thus will let code both of them.

The DOBOT conveyor belt is a specific device, especially designed to work with both the DOBOT Magician Lite and the Magic Box, and the motion is applied trough the rotation of a stepper motor. However, different stepper motor belt may be connected to the Magician Lite robot, and similar solutions may be used with other brands arms.

To control both the arm and the belt we stated that it is necessary to code the external controller called Magic Box, that can be added as a programmable device inside the programming software:



In this way only the Magic Box may be programmed, but if you want to control also the arm it is necessary to add the Arm extension



Choose an Extension

Magician Lite
AI+STEAM educational robotic arm specially designed for K12.
+ Add extension

Magician
Multifunctional STEAM educational robotic arm
+ Add extension

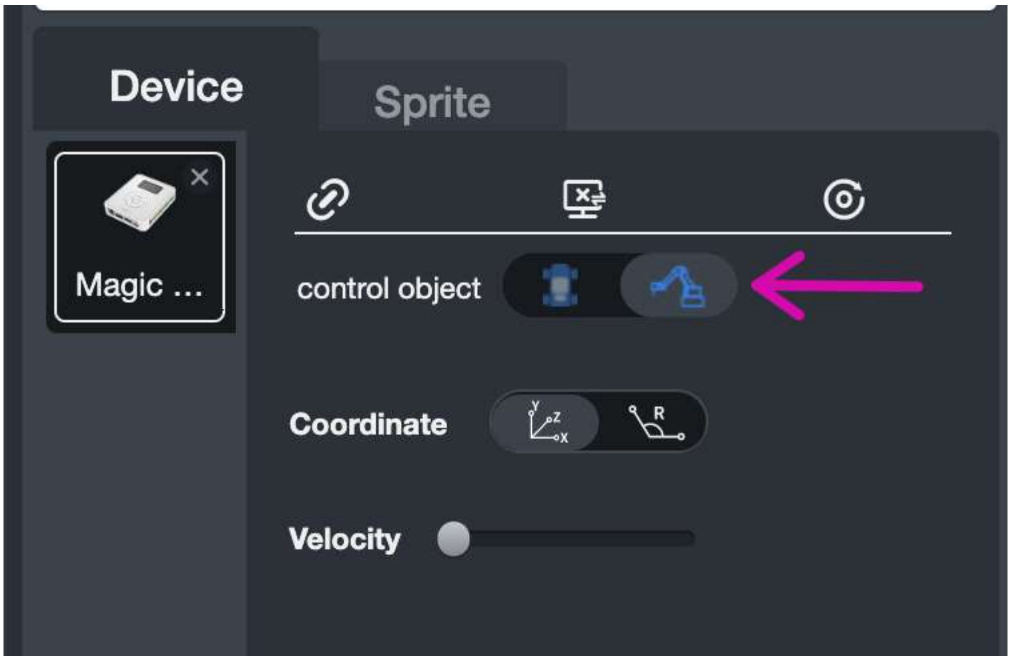
Sliding Rail Kit
Used to extend the range of motion of the robot
+ Add extension

Photoelectric & Color Sensor
For photoelectric sensing and color recognition
+ Add extension

AI Sensor Kit
12 kinds of sensor modules, rich in robotic arm applications
+ Add extension

Magician Go Kit
Omnidirectional mobile platform, specially designed for AI+STEAM education
+ Add extension

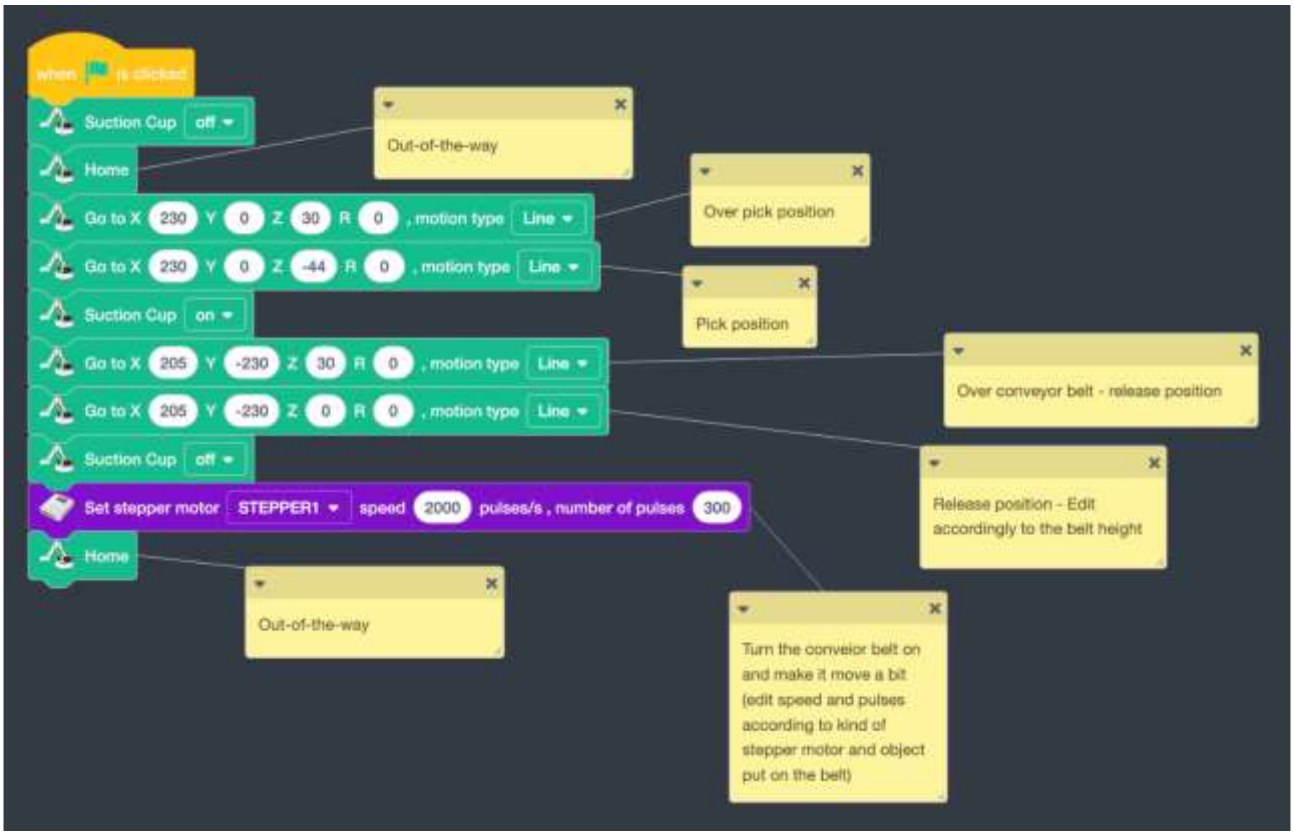
and then select – from the Magic Box setup sections – that you would like also to control the Arm:



Below you may find a general example program that operate a simple process:

```
Pseudocode:
RELEASE GRIPPER
MOVE TO Out-of-the-way position
MOVE TO Over Pick Position
MOVE TO Pick Position
GRIP GRIPPER
MOVE TO Over Conveyor Belt
MOVE TO Conveyor Belt Release Position
RELEASE GRIPPER
TURN ON Conveyor Belt
MOVE TO Out-of-the-way position
```

And this example may be implemented with the real robot as follow.



```
when 🏳 is clicked
Suction Cup off
Home                                                         Out-of-the-way
Go to X 230 Y 0 Z 30 R 0 , motion type Line                  Over pick position
Go to X 230 Y 0 Z -44 R 0 , motion type Line                 Pick position
Suction Cup on
Go to X 205 Y -230 Z 30 R 0 , motion type Line               Over conveyor belt - release position
Go to X 205 Y -230 Z 0 R 0 , motion type Line                Release position - Edit accordingly to the belt height
Suction Cup off
Set stepper motor STEPPER1 speed 2000 pulses/s , number of pulses 300
                                                             Turn the conveior belt on and make it move a bit (edit speed and pulses according to kind of stepper motor and object put on the belt)
Home                                                         Out-of-the-way
```

Different arms or conveyor belt may be used differently, and has different technical needs, but the general approach may be shared and used as we presented. At the same time, different techniques are shared with a conveyor belt: if you reason about what a conveyor belt does, it applies a translation to pieces layed onto the belt. However, sometimes it is useful to move the robot itself instead of the pieces (ie if the same robot should operate in different locations, or if a piece must be moved from one location to another and the conveyor belt may be not appropriate). In that cases the translation can be applied to the robot using some linear actuator, or other devices, but the general approach is pretty similar to the conveyor belt usage.

Computer vision

Computer vision is a transformative technology in industrial robotics, enabling machines to interpret and make decisions based on visual data. By simulating human vision, computer vision systems allow robots to perform tasks with higher precision, efficiency, and autonomy.

Computer vision in industrial robotics begins with image acquisition, where cameras capture visual data from the environment. This data is then processed using algorithms that enhance image quality, detect edges, and identify patterns. Advanced techniques, often incorporating machine learning, allow the system to recognize and classify objects within these images. Finally, the robot uses this processed information to make decisions and perform actions, such as picking and placing objects, navigating spaces, or inspecting products.

The integration of computer vision transforms a robot from a simple automated machine into a sophisticated system capable of complex interactions with its environment. By interpreting visual inputs, the robot can adapt to varying conditions, recognize different objects, and execute tasks that require a high degree of precision and adaptability.

One of the primary applications of computer vision in industrial robotics is quality control and inspection. Vision systems can detect defects in products on the assembly line, ensuring that each item meets the required standards. This automated inspection process not only improves product quality but also reduces the incidence of defective goods reaching consumers.

In pick and place operations, vision-guided robots excel at locating, identifying, and manipulating objects with remarkable precision. This capability is crucial in packaging, sorting, and assembly tasks, where accurate positioning is essential for efficiency and product integrity.

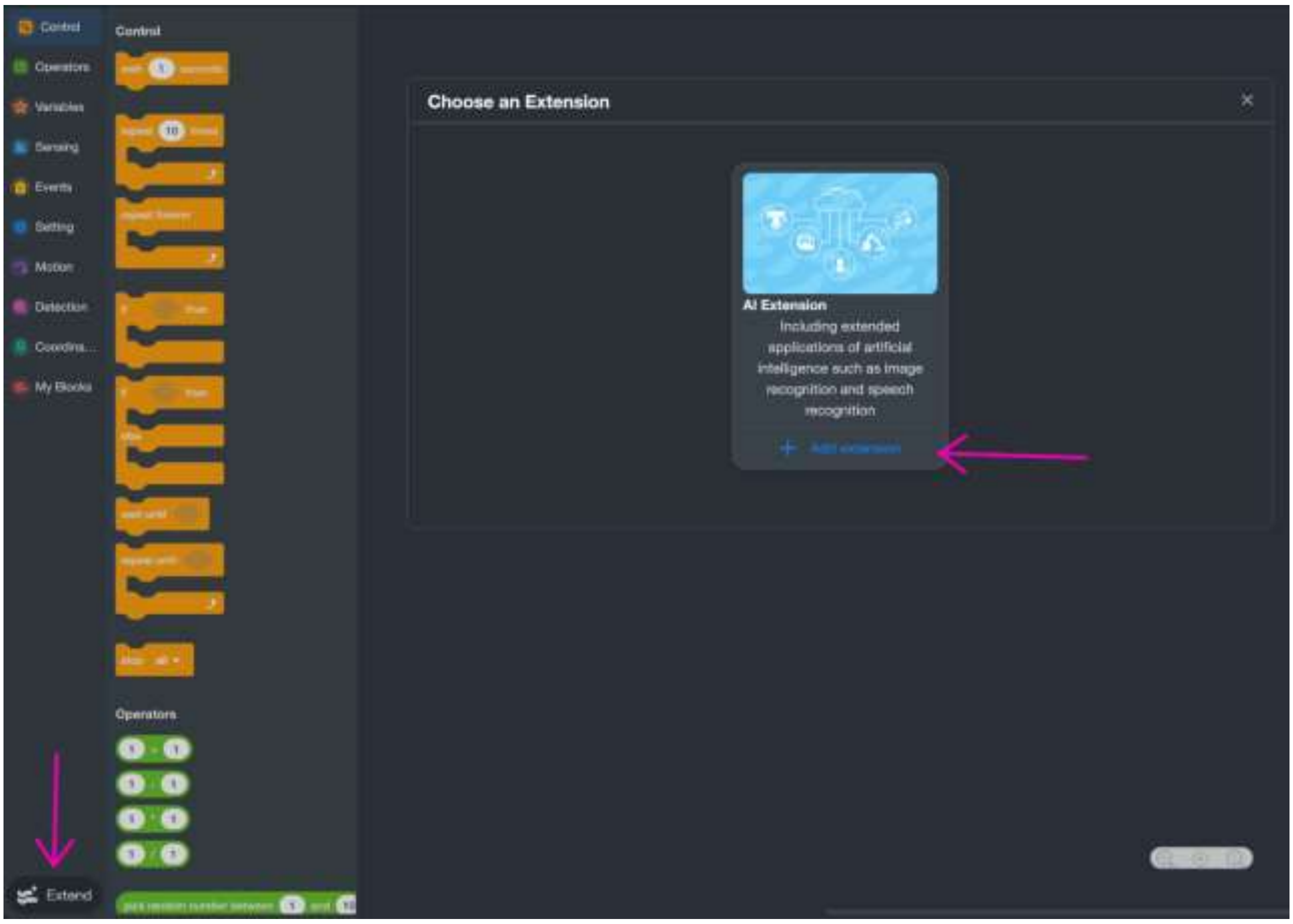
For guidance and navigation, computer vision enables robots to navigate complex environments by recognizing obstacles and planning optimal paths. This functionality is particularly beneficial in warehouses and manufacturing floors, where dynamic and unpredictable conditions prevail.

Another important application is in measurement and metrology. Vision systems can measure the dimensions and geometries of objects with high accuracy, which is crucial in manufacturing processes that demand exact specifications. This ensures that components fit together perfectly, reducing waste and rework.

Example of Computer Vision with DOBOT Magician Lite

The DOBOT Magician Lite robot is equipped with a computer vision system, so an exteral camera that can be used to detect objects. Since the camera can be calibrated, it let also detect workpiece coordinates.

To use the computer vision system it is required to mount the external webcam onto the arm, then inside the programming software to upload the artificial intelligence extension:



After it, it is necessary to define a new classification model. DOBOT software uses an AI based approach.

The software simplifies the process of creating image classification models by providing an intuitive, interface, that requires a few steps

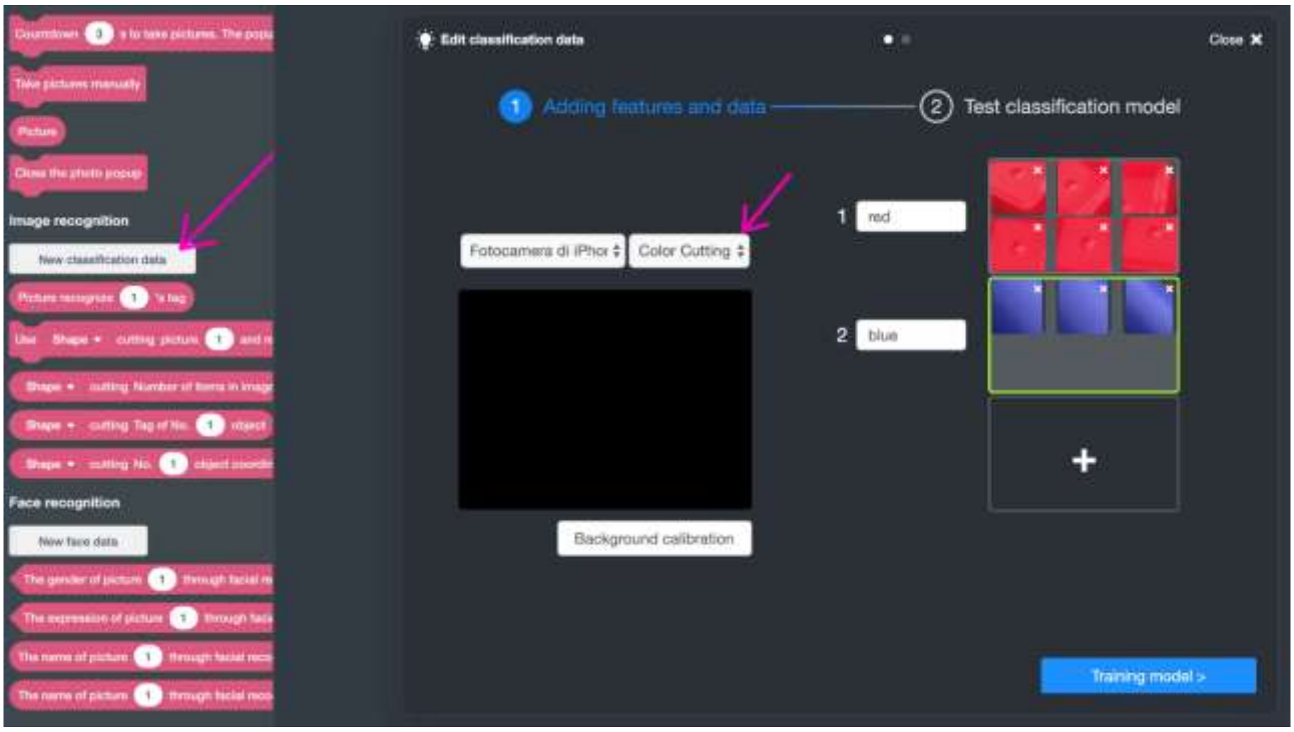
1. **Data Collection**: users begin by collecting and uploading images for each category they want to classify. These categories could be anything, such as different types of objects

2. **Training the Model**: Once the images are uploaded, the software uses machine learning algorithms to train a model. During training, the system analyzes the uploaded images to learn distinguishing features for each category. This involves adjusting the model's parameters to minimize classification errors
3. **Testing and Improving**: After training, users can test the model by providing new images to see how well it classifies them. If the model's performance is unsatisfactory, users can add more images or make adjustments to improve accuracy
4. **Use the Model**: once the model performs satisfactorily, it can be applied and integrated into various applications

To follow this steps, you should start a new classification data, and then select if you want to dedect:

- Only shapes
- Only colours
- Objects

In the following example the procedure is done to detect colours, so "Color Cutting" is selected.



Fotocamera di iPhor | Color Cutting

1 red

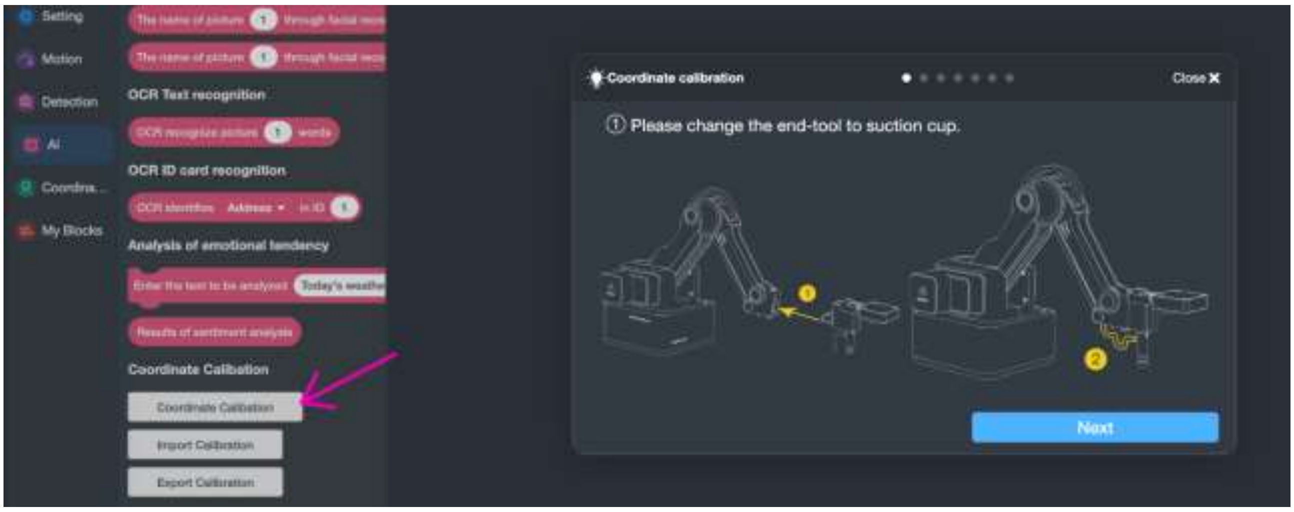
2 blue

Then the robotic arm – and its camera – should be positioned over the working plane. The system works better if the background is uniform in colour, and it has a bright contrast with the objects. The first step is to click the "Background calibration" button, with the camera facing the empthy working plane. It let the arm have a reference point.

Then different objects may be shown to the camera, over the plane, and manually clicked and sorted into categories. In this example red and blue categories has been (manually) created, and a few examples of red and blue objects has been shown and stored inside the system.

At this point it is required to train the model, and then the software will provide a test environment to validate if the model works properly.

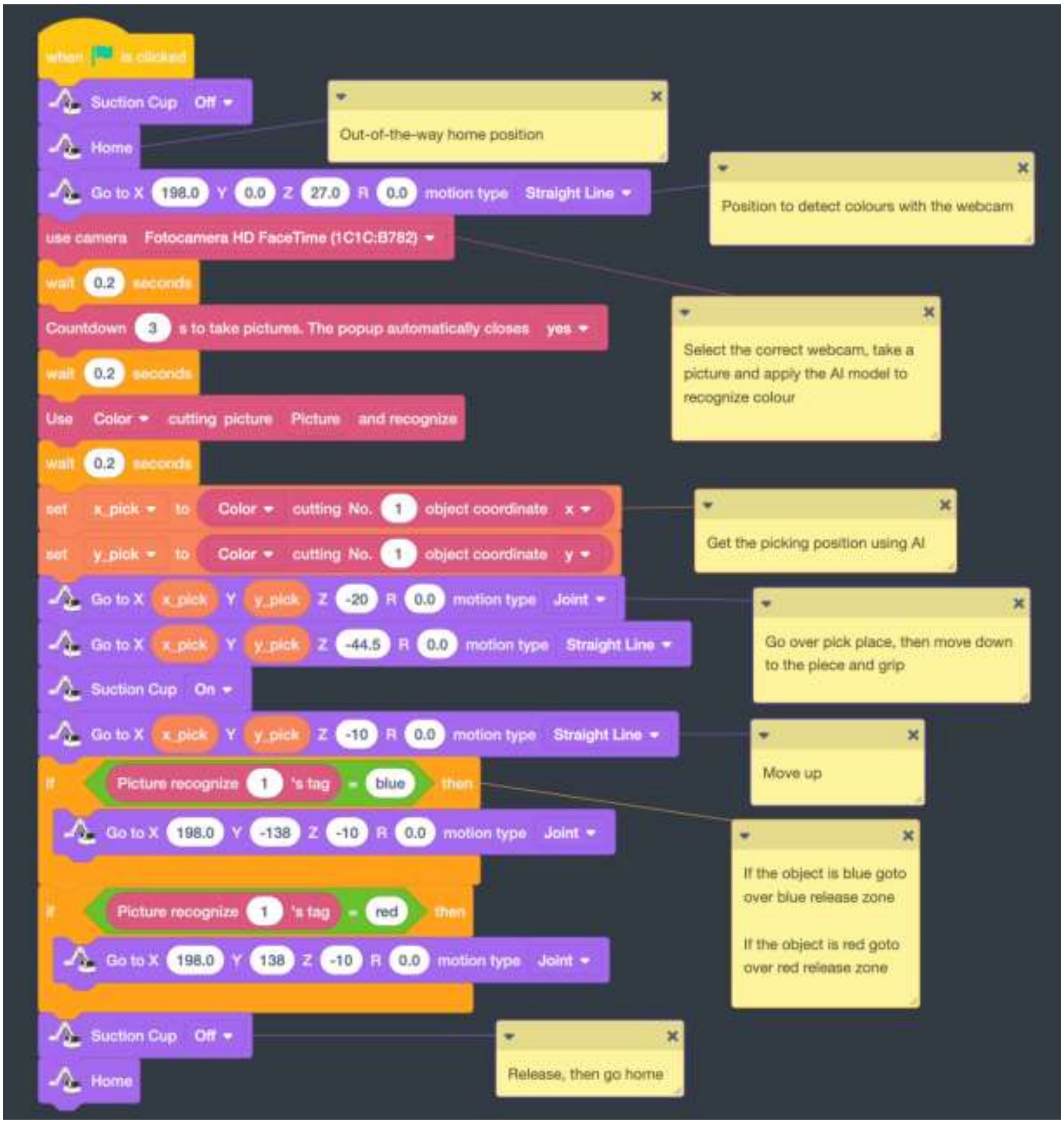
Once it has been done, the IA is capable of recognising objects, but since the webcam may be mounted in different positions, the arm is not capable of determine the position of the identified objects. If you want to do so, you should do a calibration procedure (operations shown in the popup window, and inside the robot manual), to let the robot calculate the offset between the centre of the webcam and the TCP.



Please change the end-tool to suction cup.

Once the calibration is done, the robot is also capable of determine where a specific piece is.

The following example uses this approach, then with computer vision recognise the colour of objects on the workplate, then identify their position. It uses this data to sort objects by colour.



```
when [flag] is clicked
Suction Cup Off
Home
Go to X 198.0 Y 0.0 Z 27.0 R 0.0 motion type Straight Line
use camera Fotocamera HD FaceTime (1C1C:B782)
wait 0.2 seconds
Countdown 3 s to take pictures. The popup automatically closes yes
wait 0.2 seconds
Use Color cutting picture Picture and recognize
wait 0.2 seconds
set x_pick to Color cutting No. 1 object coordinate x
set y_pick to Color cutting No. 1 object coordinate y
Go to X x_pick Y y_pick Z -20 R 0.0 motion type Joint
Go to X x_pick Y y_pick Z -44.5 R 0.0 motion type Straight Line
Suction Cup On
Go to X x_pick Y y_pick Z -10 R 0.0 motion type Straight Line
if Picture recognize 1 's tag = blue then
    Go to X 198.0 Y -138 Z -10 R 0.0 motion type Joint
if Picture recognize 1 's tag = red then
    Go to X 198.0 Y 138 Z -10 R 0.0 motion type Joint
Suction Cup Off
Home
```

Out-of-the-way home position

Position to detect colours with the webcam

Select the correct webcam, take a picture and apply the AI model to recognize colour

Get the picking position using AI

Go over pick place, then move down to the piece and grip

Move up

If the object is blue goto over blue release zone

If the object is red goto over red release zone

Release, then go home

The previous approach may be considered as the following pseudocode, that may be used with different computer vision systems (delay are here removed, since in different systems it may be not necessary).

```
RELEASE GRIPPER
MOVE TO Out-of-the-way position
MOVE TO Detect Position
CONFIGURE Camera
TAKE Picture
APPLY AI MODEL TO Picture
GET COORDINATE OF THE PIECE
MOVE TO Over Piece
MOVE TO Piece Position
GRIP GRIPPER
MOVE TO Over Piece
IF OBJECTS IS BLUE
MOVE TO Blue Release Zone
IF OBJECTS IS RED
MOVE TO RED Release Zone
RELEASE GRIPPER
MOVE TO Out-of-the-way position
```

CHAPTER 5

In automated production and industrial robotics, errors and issues are inevitable. Effective troubleshooting not only helps resolve immediate problems but also contributes to performance improvements and reduces downtime. This chapter covers the most common errors that occur with KUKA, FANUC, and Mitsubishi robotic arms, along with steps for their resolution.

---

1. Common Types of Errors

In automation and robotics, errors can lead to significant downtime and even damage to equipment or materials. Understanding the basic types of errors in robotic arm operations is crucial for effective troubleshooting and maintaining continuous operation. The following are the most common types of errors encountered in robotic arm operation.

1.1 Sensor and Actuator Errors

Sensors and actuators are essential components for the precision of a robot's movements. Sensors gather information about the environment, while actuators enable precise movements. The most common issues in this area include:

- **Inadequate sensor calibration**: Over time, sensors may be affected by vibrations, temperature, or regular use. When a sensor is not accurately calibrated, it may send incorrect data to the controller, resulting in inaccurate movements.
- **Incorrect encoder data**: Encoders measure the position, speed, and direction of the robotic arm. When they are damaged or improperly connected, the robotic arm may lose reference points, causing positioning errors.
- **Actuator issues**: Faulty actuators can cause imprecise movements, leading to increased wear on parts and higher maintenance costs.

1.2 Software Errors

Software errors often include problems in programming, configuration, and interconnections between robot components. For example:

- **Incorrect program logic**: Poorly arranged program blocks can cause collisions or unexpected movements, often due to programming errors or inadequate path planning.
- **Incompatibility between software modules**: Using outdated or incompatible software versions can cause operational issues, especially when external add-ons or complex algorithms are used.
- **Errors during software updates**: Software updates may introduce incompatibilities or new settings that require additional configurations. During updates, random interruptions or code errors may disrupt normal robot operations.

1.3 Communication Errors

Communication between the robot controller, sensors, and control systems (such as PLC controllers) is essential for coordinated operation. Communication errors often involve:

- **Interruptions in Ethernet or Wi-Fi connections**: Cables may become damaged or connectors poorly secured, disrupting data transmission.
- **Communication protocol issues**: Industrial robots use various communication protocols (such as Ethernet/IP, DeviceNet, or ProfiNet). Problems arise when these protocols are improperly configured or there is a mismatch between devices.
- **Interference and signal latency**: When a robot operates in an environment with numerous electronic devices, signal interference may reduce communication reliability.

1.4 Mechanical Errors

Mechanical errors can result from prolonged use, poor maintenance, or unexpected impacts and vibrations. They are most commonly found in:

- **Worn parts**: Robotic arms consist of many moving parts that wear out over time, especially in intensive production processes.
- **Cracking or bending of joints and shafts**: This damage can occur due to improper loading or unexpected movements. Bent or damaged components cause issues with precision and strength.
- **Misaligned gears or chains**: Gears and chains that transmit movement may suffer excessive wear, leading to skipping or irregular operation of the robotic arm.

These errors are characteristic of robotic arm operation in industrial settings, where a high level of accuracy and reliability is required. Addressing these problems in a timely manner allows companies to significantly reduce maintenance costs and improve production efficiency.

---

2. Troubleshooting KUKA Robotic Arms

KUKA robotic arms use KRC controllers known for flexibility and advanced error detection options. The most common errors with KUKA robots include:

1. **Sensor error in movement precision**:
    - Check sensor connections.
    - Calibrate sensors using KUKA calibration software.
    - Review errors in the KUKA HMI interface and display error details.
2. **Software issues in KUKA WorkVisual or KSS system**:
    - Update KSS (KUKA System Software) to the latest version.
    - Check all movement-related parameters in WorkVisual software.

- Restart the system and check if the error persists.

3. **Communication interruption with the controller (Ethernet/Fieldbus)**:
    - Check network cables and connectors.
    - Test communication using diagnostic tools available on the KRC controller.
4. **Motor or servo drive overheating**:
    - Adjust cooling speed or check for vent blockages.
    - Reduce the robot's movement speed in program instructions to decrease load.

---

3. Troubleshooting FANUC Robotic Arms

FANUC robotic arms use R-30iA, R-30iB, and the latest R-30iB Plus controllers. The most common issues include:

1. **Software errors (SRVO-037, SRVO-023)**:
    - SRVO-037: Check the encoder position and verify all motor connectors.
    - SRVO-023: Reset the error and check the motor power supply.
2. **Collision or scratching issues with objects**:
    - Use the FANUC iRVision system to verify position accuracy.
    - Check all points in the program for potential collisions and adjust paths.
3. **Controller error reset**:
    - Use the FANUC Teach Pendant to reset errors.
    - Log in as an administrator for access to advanced diagnostic options.
4. **Communication errors (Ethernet/IP or DeviceNet)**:
    - Check the status of the Ethernet/IP adapter.
    - Use diagnostic tools on the FANUC controller for communication analysis.

---

4. Troubleshooting Mitsubishi Robotic Arms

Mitsubishi robotic arms use the CR800 series controllers and MelFA-Works software for programming and diagnostics. Common errors include:

1. **Initialization or calibration error**:
    - Restart the calibration process using MelFA-Works software.
    - Check motor status and positional encoders.
2. **Incorrect arm movement or positioning**:
    - Ensure all points are precisely set in the program.
    - Use the Position Adjustment option in MelFA-Works software.
3. **Communication errors with the controller**:
    - Check all communication cables and connectors on the controller.
    - Use diagnostic tools in MelFA-Works to test network status.
4. **Overheating or power loss**:
    - Check ventilation and clean vent openings.
    - Reduce speed and check the torque on the motors.

---

5. Preventive Measures for Reducing Errors

- **Regular software updates**: Ensure all robots are updated to the latest software versions.
- **Regular sensor calibration and inspection**: Maintaining sensor accuracy reduces positional errors.

- **Employee training**: Well-trained operators and engineers significantly reduce operational errors.
- **Proper equipment maintenance**: Routine maintenance, replacement of worn parts, and ventilation system cleaning increase efficiency and extend robot life.

Timely detection and resolution of errors are essential for reducing downtime and maintaining high operational efficiency in industrial robots. By understanding and applying troubleshooting procedures, users of KUKA, FANUC, and Mitsubishi robotic arms can improve system performance, extend equipment life, and reduce production costs.

CONCLUSIONS

Worldwide, industry and manufacturing continue to install robots at a significant growth rate. The IFR, International Federation of Robotics' latest World Robotics report recorded 4,281,585 units operating in factories worldwide in 2023. "This is a 10% increase from the year before. For the third year in a row, annual robot installations exceed half a million units" reports the International Federation of Robotics (IFR). By region, 70% of all newly deployed robots in 2023 were installed in Asia, 17% in Europe, and 10% in the Americas".

In all nations, even advanced ones, the demand for robots in production has increased and "Automation allows manufacturers to locate production in developed economies without sacrificing cost efficiency, according to the Frankfurt, Germany-based organization".

The IFR said it expects the global economic downturn to bottom out this year, with global robot installations levelling off at 541,000 units. It expects growth to accelerate in 2025 and continue in 2026 and 2027. There are no signs that the overall long-term growth trend will end in the near future, asserted the IFR. (IFR, World Robotics Report, 2024).

The authors of this Handbook INDUSTRIAL ROBOTICS - operating and programming, which the Project offers to teachers and students, certainly did not aim to exhaust the topics of an industrial robotics textbook, but rather to interest many young people in pursuing the study of industrial robotics, and robotics in general.

As manipulative robotics, robotic arms, have a long history of research and application, we see that the scope of their use has expanded to other fields, and they are now used not only in manufacturing, but also in logistics and transport, shipping and naval technologies, medicine, space robotics, space travel and assistance and entertainment.

Developing a good industrial robotics competence opens up unforeseen avenues for young people that were unthinkable just a few years ago.

Bibliography

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. Themes in Science & Technology Education, Vol 6, pp 63-71.

Angel-Fernandez, J., M., Vincze, M. (2018). Philipp Zech, Justus Piater (Eds.) Proceedings of the Austrian Robotics Workshop 2018, pp 37-42. Innsbruck university press, ISBN 978-3-903187-22-1, DOI 10.15203/3187-22-1

Automatic Tool Changer. (2024). In Wikipedia. Retrieved May 28th, 2024 from https://en.wikipedia.org/wiki/Automatic_tool_changer

Ben-Ari M., Mondada F. (2018). Robots and Their Applications. Elements of Robotics. Springer, Cham. https://doi.org/10.1007/978-3-319-62533-1_1

Comau (2023). Comau Robotics Product Instruction. PROGRAMMAZIONE DEL MOVIMENTO C5GPlus – R1C Rel. 4.3x - Software di Sistema [MOVEMENT PROGRAMMING C5GPlus - R1C Rel. 4.3x - System Software].

Comau (2023). Comau Robotics Product Instruction. USO DELL’UNITÀ DI CONTROLLO C5GPlus – R1C Rel. 4.3x - Software di Sistema [USE OF THE C5GPlus CONTROL UNIT - R1C Rel. 4.3x - System Software]

Craig, J. J. (2005). Introduction to Robotics: Mechanics and Control. Pearson Prentice Hall.

De Luca, A. (nd). Robotics 1 - Direct kinematics. Sapienza Università di Roma. Retrieved May 10, 2024 from https://www.diag.uniroma1.it/~deluca/rob1_en/09_DirectKinematics.pdf

DOBOT. (2022). Dobot Magician Lite User Guide.

EARLY - Education Advancements through Robotics Labs for Youth. (n.d.). Robotics database. Retrieved April 8, 2021 from https://edurobots.eu/robotics-database/

EARLYCODE Consortium. (2021). Developing Teaching Materials for Preschool Teaching Undergraduates on Computational Thinking and Introduction to Coding. pp 12-13. Retrived May 8, 2024 from http://www.earlycoders.org/Pages/2009/1217/IO---3.

e-Media project Consortium. (2019). Educational Robotics. Retrieved April 8, 2021 from https://all-digital.org/resources/educational-robotics-handbook/

Encyclopedia of Mathematics. (2014) Cartesian orthogonal coordinate system. Retrieved May 9, 2024 from:
http://encyclopediaofmath.org/index.php?title=Cartesian_orthogonal_coordinate_system&oldid=18006

Flynt, J. (2019). What are robotic arms and how do they work?. Retrieved May 9, 2024 from https://3dinsider.com/what-are-robotic-arms/

Goodwin, D. (2022). How to Define the Tool Center Point (TCP) on a Robot. Retrieved May 28, 2024 from https://control.com/technical-articles/how-to-define-the-tool-center-point-tcp-on-a-robot/

Guizzo, E. (2018, August 01). What Is a Robot? Top roboticists explain their definition of robot. IEEE Robots, your guide to the world of robots. https://robots.ieee.org/learn/what-is-a-robot/

HarperCollins. (n.d.). Robot. In HarperCollins COBUILD Advanced English Dictionary, online. Retrieved April 8, 2021 from https://www.collinsdictionary.com/dictionary/english/robot

International Federation of Robotics. (2022). Industrial Robots Definition.

International Organization for Standardization. (2021). ISO 8373:2021. Robotics – vocabulary. Edition 3.

Meir, et al. (2024). Kinematic Optimization of a Robotic Arm for Automation Tasks with Human Demonstration.

Merriam-Webster. (n.d.). Robot. In Merriam-Webster.com dictionary. Retrieved April 8, 2021 from https://www.merriam-webster.com/dictionary/robot

Owen-Hill, A. (2022). Robot Singularities: What Are They and How to Beat Them. RoboDK Blog. Retrieved May 15, 2024 from https://robodk.com/blog/robot-singularities/

Papert, S. (1980). Mindstorms: Computers, Children and Powerful Ideas. Basic Books.

Piaget, J. (1974). To Understand is to Invent. Basic Books.

Robot end effector. (2024). In Wikipedia. Retrieved May 28th, 2024 from https://en.wikipedia.org/wiki/Robot_end_effector

Robots Done Right. (2023). Robotic Controller. Retrieved May 9th, 2024 from https://robotsdoneright.com/Articles/robotic-controller.html

Sant’Anna School of Advanced Studies – Pisa. (n.d.). Educational robotics. Retrieved April 8, 2021 from https://www.santannapisa.it/en/institute/biorobotics/educational-robotics

Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G. (2008). Robotics: Modelling, Planning and Control (1st. ed.). Springer Publishing Company, Incorporated.

Universal Robots. (2022). Robotic Arm: Components, Types, Working, Applications & More. Retrieved May 9, 2024 from https://www.universal-robots.com/in/blog/robotic-arm/

Universal Robots. (2022). Types of Industrial Robots - Articulated, SCARA, Delta, Cartesian & More.

Universal Robots. (2022). Six-Axis Robots - A Comprehensive Overview.

Universal Robots. (2022). Collaborative Robots (Cobots) - What Are They & How Do They Work?

Walker, J., Resnick, R. and Halliday, D. (2014) Fundamentals of Physics / Extended. Hoboken, NJ: John Wiley & Sons, Inc.

Wegener et al. (2015). Robot Assisted Disassembly for the Recycling of Electric Vehicle Batteries. Procedia CIRP. 29. 10.1016/j.procir.2015.02.051.

Wikipedia. (2023). Robot control. Retrieved May 9, 2024 from https://en.wikipedia.org/wiki/Robot_control