POLITECNICO DI TORINO

Department of Management and Production Engineering

Master's Degree in Engineering and Management

Optimizing the MiR100 Performance in Internal Logistics through Digital Twin-Based Dynamic Speed Adjustment

Supervisor: Professor Giulia Bruno
Co-Supervisor: Dr. Khurshid Aliev
Candidate: Sevda Soheilifar

A.Y 2024/2025

Acknowledgements

I would like to sincerely thank my thesis supervisor, Professor Giulia Bruno, and my co-supervisor, Dr. Khurshid Aliev, from the Department of Management and Production Engineering (DIGEP) at Politecnico di Torino. I am truly grateful for your support and guidance throughout this journey. Your encouragement gave me the confidence to keep pushing forward.

I am deeply grateful to my family for their unconditional love and encouragement in every stage of my life, especially during these intense academic years. Your constant support, in every message, every call, and every moment, you reminded me to keep going meant the world to me. You taught me how to be strong, consistent, and to keep believing in myself, even when things were difficult.

I am grateful to my friends, both those I have known for years and those I have met along the way during this journey. Some of you may be far away, but always close to my heart. Your presence made this experience not only easier, but also more unforgettable.

Abstract

Internal logistics is a part of the logistics that deals with the flow of materials or information within the organization. In the manufacturing companies, this is focused on the delivery of materials or semi-finished products within the manufacturing companies. Thanks to technological advancement various industries' operation, particularly internal logistics has significantly changed.
This evolution is driven by Industry 4.0, under which manufacturing industries have embraced automation and smart manufacturing to digitalize these internal flows by using autonomous mobile robots (AMR) or automated guided vehicles (AGV). While automating logistics process eliminates many manual and repetitive operations, it also introduces new challenges such as use the vehicles as effectively as possible and minimizing their energy consumptions as they are dependent on energy sources such as batteries. Therefore, to achieve operational and sustainability objectives in manufacturing framework at the same time, an industry 5.0 approach, which focuses on protecting the environment and achieving resilient industrial operations, is introduced.
Speed of AMRs and AGVs is one of the critical factors in energy consumption, higher speeds deplete the battery much faster. By integration of Digital Twin technology with AMRs, manufacturers can have an opportunity to monitor real-time data and collateral adjustment based on a continuous performance evaluation, facilitating quick responses to changing production conditions to achieve maximum transport efficiency and improving manufacturing performance.

The focus of study is on optimizing energy consumption through speed adjustment while simultaneously improving operational efficiency. This thesis aims to contribute to leveraging Digital Twin technology with Autonomous mobile robot, specifically MiR100 within manufacturing frameworks. The research framework is implemented in Mind4Lab at Politecnico di Torino, to collect real time data of the MiR100 robot utilizing Node-Red, FlexSim simulation software and the Modbus communication protocol. The first part of the research is aimed at evaluating energy consumption and battery behavior across different speeds, specifically high-speed and low-speed scenarios, while the second part of the study contributes to leveraging Digital Twin framework that allows dynamic speed regulation in response to real-time data demonstrating how this can enhance energy and operational efficiency.

List of Contents

List of Figure

List of Table

Acronym List

AMR	Autonomous Mobile Robot
AGV	Automated Guided Vehicle
DM	Digital Model
DS	Digital Shadow
DT	Digital Twin
MiR	Mobile Industrial Robot
PLC	Programmable Logic Controller
PLM	Product Lifecycle Management
DMU	Digital Mock-Up
IoT	Internet of Thing
CAD	Computer-Aided Design
TCP/IP	Transmission Control Protocol / Internet Protocol
KPI	Key Performance Indicator
3D	Three-Dimensional
ML	Machine learning
FIFO	First in first out
LIFO	Last in first out

Glossary

Autonomous Mobile Robot: A robot capable of navigating its environment without fixed paths or human intervention, using sensors and onboard intelligence.

Automated Guided Vehicle: A mobile robot that follows predefined paths, typically using wires, magnets, or tracks on the floor.

Cycle Time: The total time required to complete a process or transport task from start to finish.

Digital Model: It is a static digital representation of an existing or planned physical object without automated data exchange

Digital Shadow: It enables real-time data flow from the physical system to the digital model, but does not affect the physical system automatically.

Digital Twin: A dynamic digital representation of a physical system that is continuously updated with real-time data and can interact with the physical system.

Physical Entity/System: It refers to the tangible components of the system, including the autonomous mobile robot (MiR100) and manipulator.

Interarrival Time: The time between the arrivals of two consecutive items in a system.

1. Introduction

Industries are the backbone of a nation's economy; by improving their performance and efficiency they could yield stronger economic growth. The concept of **Industry 4.0 (I4.0)** has emerged in recent years as a major transformation in the industrial sector. Manufacturing digitalization and robotization of production process is the main purpose of the fourth industrial revolution 4.0, also termed Industry 4.0 (I4.0) (Rizqi et al., 2024a); (Pizoń et al., 2024a). This paradigm focuses on increasing productivity and operational efficiency, improving the quality of products and processes, and reducing production costs, optimality and transparency of industry processes (Fera et al., 2020); (Rizqi et al., 2024b). Industry 4.0 is also used in the field of logistics to be more specific internal logistics with the introduction of robotics and automation (Zoubek & Simon, 2021). For instance, Autonomous Mobile Robots (AMRs) and AGVs are a key component of the internal transport system which improves the internal transport processes through eliminating many repetitive manual operations (Dobrzańska & Dobrzański, 2025). Development based on industry 4.0 presents a challenge such as managing energy consumption for this sector as well as opportunities. To deal with sustainability and resilience challenges, Industry 5.0 approach is introduced. The optimization of AMR robots in terms of energy consumption and manufacturing parameters is a critical part of successful and sustainable manufacturing performance. Therefore, both approaches are considered in this study.

Minimizing energy consumption can be achieved in several ways, such as finding better routes, avoiding frequent accelerations and turns (Mei et al., n.d.).(Wu et al., n.d.) in their study titled "Green Automation: Empowering a Sustainable Future with Energy-Efficient Autonomous Mobile Robots in Manufacturing" indicated that battery Management Systems (BMS) technologies by deployment sophisticated algorithms, monitor battery health, manage charging and discharging cycles, predict battery life expectancy and change the operations to keep the optimal energy usage during the battery's lifespan. Another way is adaptive control strategies that dynamically adjust the operational parameters of AMRs such as speed, acceleration, and payload handling based on current energy levels and task requirements in real-time. As mentioned, reducing the speed of the AMRs could be helpful. However, after conducting research on MiR100 framework, there is not any study which analyzes the possible correlation between transporting speed and energy consumption. Therefore, the first part of the experiment is done to evaluate battery behavior within different speeds, specifically high speed and low speed. The second part of the study, the main part of the study, focused on identifying the optimal speed to obtain the lowest energy consumption, considering that the speed changes when the floor storage, where it is the destination of transported items by MiR100, reaching the maximum content on.

To implement this adaptive control system, Digital Twin framework is used, which could adjust the MiR100 speed, based on the real-time data. This adjustment ensures that the robot operates and will not stay idle and decrease energy consumption at the same time.

To evaluate the proposed framework, the experiment was implemented in the Mind4Lab at Politecnico di Torino.
The work is divided into following parts: (i) provide the theorical background in industrial revolution, Digital Twin concepts and automated mobile robot, (ii) explore relevant previous study, (iii) provide a brief description of Digital Shadow and Digital Twin framework and motivation (iv) describe the case study and scenarios to evaluate the framework. Then it compares the results, indicating how the proposed framework helps to achieve the operational and sustainability objectives.

2. Theorical Background

2.1 Overview of Industrial Revolution

The increasing integration of the Internet of Thing into the industrial value chain has built the foundation for Industry 4.0 (Hermann et al., 2016). However, industry growth is fundamentally related to social and economic development. Only a decade after industry 4.0, the world came across new global challenges, so the European Commission announced Industry 5.0 to address these challenges (Aheleroff et al., 2022). These industrial revolutions follows three previous industrial revolutions in human history.

The first industrial revolution was the introduction of mechanical production facilities starting in the second half of the 18th century and being intensified throughout the entire 19th century (Hermann et al., 2016). This revolution was characterized by single or multiple machines for a particular and repetitive task (Zafar et al., 2024). From the 1870s on, electrification and the division of labor (i.e. Taylorism) led to the second industrial revolution (Hermann et al., 2016). Industry 2.0 is characterized using the electrical power and mass production (Zafar et al., 2024). The third industrial revolution, also called “the digital revolution”, set in around the 1970s, when advanced electronics and information technology developed further the automation of production processes (Hermann et al., 2016).

The term "Industry 4.0" was first introduced in Germany. It is associated with digital transformation of industrial processes piloted by German industry (Golovianko et al., 2022). This transformation is identified by the IoT, cloud computing, AI, machine learning and cognitive computing and complex control systems which enable dynamic and flexible production processes (Zafar et al., 2024); (Golovianko et al., 2022); (Barata & Kayser, 2024). Numerous manufacturing technologies play an important role in realizing the promises of Industry 4.0. Within this technological framework, Digital Twin has become the leading and important of manufacturing digitalization technology, as it connects the physical and cyber worlds by creating high-fidelity virtual models that replicate and simulate the behavior of their physical counterparts (Ouahabi et al., 2024).

Industry 5.0 is introduced by the European commission after about 10 years of industry 4.0 introduction (Golovianko et al., 2022). Sometimes it is called the “human-centered revolution” as it seeks to blend advanced technologies, with human-centric design principle that focus on quality of life, sustainability, and social well-being (Zafar et al., 2024). Both Industry 4.0 and Industry 5.0 rely on the extensive use of new technologies, while the former is often recognized as a technology-driven industry, the latter is a value-driven one (Golovianko et al., 2022). Figure 1 illustrates the industrial revolution (Zafar et al., 2024).

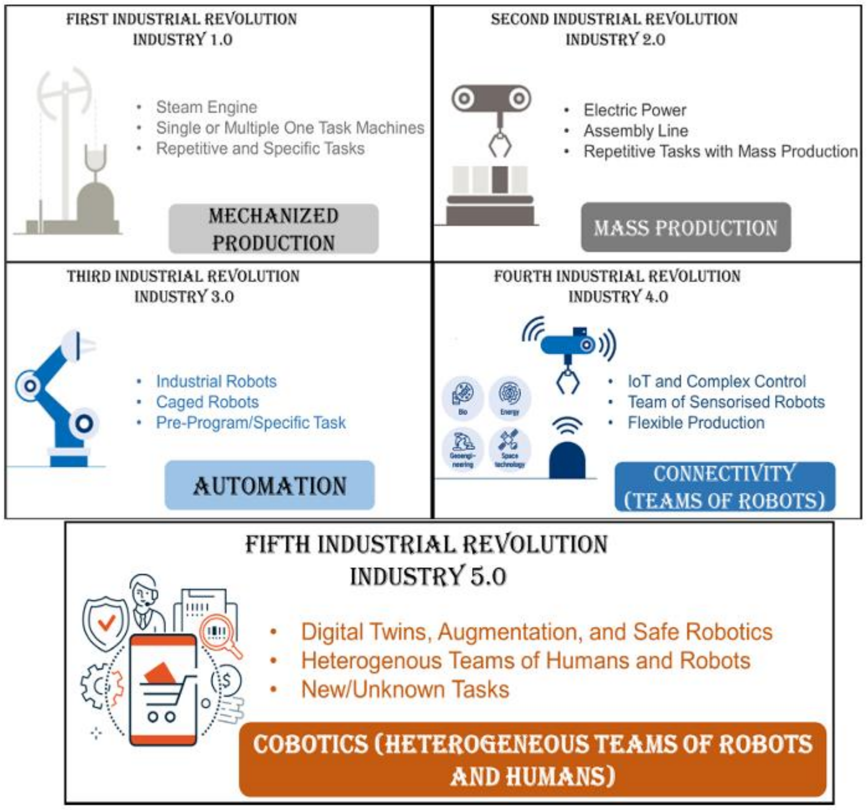

FIRST INDUSTRIAL REVOLUTION
INDUSTRY 1.0
Steam Engine
Single or Multiple One Task Machines
Repetitive and Specific Tasks
MECHANIZED PRODUCTION
SECOND INDUSTRIAL REVOLUTION
INDUSTRY 2.0
Electric Power
Assembly Line
Repetitive Tasks with Mass Production
MASS PRODUCTION
THIRD INDUSTRIAL REVOLUTION
INDUSTRY 3.0
Industrial Robots
Caged Robots
Pre-Program/Specific Task
AUTOMATION
FOURTH INDUSTRIAL REVOLUTION
INDUSTRY 4.0
IoT and Complex Control
Team of Sensorised Robots
Flexible Production
CONNECTIVITY (TEAMS OF ROBOTS)
FIFTH INDUSTRIAL REVOLUTION
INDUSTRY 5.0
Digital Twins, Augmentation, and Safe Robotics
Heterogenous Teams of Humans and Robots
New/Unknown Tasks
COBOTICS (HETEROGENEOUS TEAMS OF ROBOTS AND HUMANS)

Figure 1-Industrial Revolution from Industry 1.0 to Industry 5.0

2.2 Digital Twin Concept

The term Digital Twin introduced by Dr. Michael Grieves in 2003 in his course “Product life cycle management” at the University of Michigan (Steinmetz et al., 2022). Then in 2012, was brought to the public for the first time in NASA’s integrated technology roadmap under the technology area 11: Modeling, Simulation, Information Technology & Processing (Rosen et al., 2015). Digital Twin (DT) can be defined in various ways due to differing interpretations of its concept. These variations arise from perspectives that view DT as a model, a simulation technology, or its

integration with the Internet of Things (IoT). This difference is based on the level of integration and the mode of data exchange (Ramasubramanian et al., 2022).
As illustrated in Figure 2, the concept of Digital Twin can be categorized into three subcategories along the product lifecycle: Digital Model (DM), Digital Shadow (DS), and Digital Twin (DT) [23, 3].

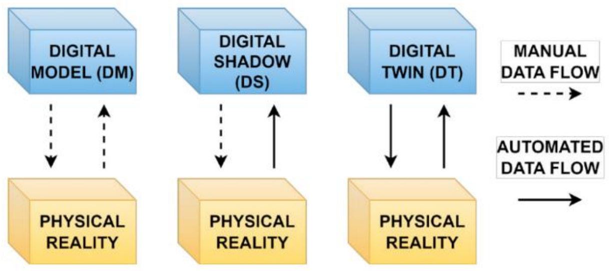


Figure 2-Digital Model, Digital Shadow and Digital Twin

The distinction between Digital Model, Digital Shadow, and Digital Twin is often ambiguous, with overlapping definitions and varying interpretations depending on context and application. To gain a clear understanding of the Digital Twin concept, it is crucial to differentiate between three related terms: **Digital Model**, **Digital Shadow**, and **Digital Twin**. The following sections outline the key features of this technology and provide precise definitions for these concepts, highlighting their unique roles and interconnections within the realm of Industry 4.0.

Digital Model

A Digital Model is a static digital representation of an existing or planned physical object without automated data exchange. While it may include detailed descriptions and rely on manually integrated data but lacks real-time interaction between the physical and digital entities. For example, simulation models of factories or mathematical models of new products are considered as digital models. In a digital model, changes in the physical object do not directly affect the digital model, and vice versa (Kritzinger et al., 2018). The modeling and simulation process refers to the process of addressing and analyzing a real-world challenge in a practical case study (Yin & McKay, 2018). These simulation models are commonly used in the design phase to predict how a system might perform under certain conditions like operational loads, degradation mechanisms, etc (VanDerHorn & Mahadevan, 2021). So this distinction is further emphasized by the fact that the Digital Model cannot send or receive data from the physical object, as it primarily comprises a 3D virtual representation (Antonelli et al., 2024).

Digital Shadow

According to the definition of a Digital Model, if an automated one-way data flow is established from the state of a physical object to its digital counterpart, this configuration is often referred to

as a Digital Shadow. In this case, changes in the physical object's state are reflected in the digital object, but the reverse interaction does not occur (Kritzinger et al., 2018).
In summary, Digital Shadow can interface with the real system and receive and process information that has already been obtained, as well as manage information that is received in real time (Antonelli et al., 2024).

Digital Twin

The last concept, Digital Twin (DT), as previously mentioned, was introduced by Dr. Michael Grieves contains three main parts: physical products in real space, virtual products in virtual space and the connections of data and information that connects the physical and virtual space (Liau et al., 2018). It is called 'Twin' because it is like mirroring or twinning of systems between what existed in real space to what existed in virtual space and vice versa (Rizqi et al., 2024b).
Digital Twin aims at creating high-fidelity virtual models for each physical entity to emulate their states and behaviors with abilities of evaluating, optimizing, and predicting (Semeraro et al., 2021). When data flows between a physical object and its digital representation are fully integrated in both directions, the system is referred to as a Digital Twin. In this configuration, the digital object can also serve as a controlling entity for the physical object. Additionally, other physical or digital entities may influence changes in the state of the digital object. A change in the physical object's state directly affects the digital object, and changes in the digital object similarly impact the physical counterpart (Kritzinger et al., 2018).
A key difference between Digital Twin and a traditional simulation lies in how they represent the system's state. While simulations are based on predefined assumptions to predict future behavior, Digital Twin continuously monitors and reflects the real-time state of an actual, operating system (Antonelli et al., 2024). Sharman et al. (2022) mentioned how Digital Twin can be different from existing technologies as described in table 1.

TECHNOLOGY	**HOW OTHER TECHNOLOGY DIFFERS FROM DT**
SIMULATION SYSTEMS	No real-time twinning
MACHINE LEARNING	No twinning
DIGITAL PROTOTYPE	No IoT components necessarily
OPTIMISATION	No simulation and real-time tests
AUTONOMOUS	No self-learning (learning from its past outcomes) necessarily
AGENT-BASED MODELLING	No real-time twinning

Table 1-Difference between Digital Twin and Other Technologies

In the last two decades, Digital Twins have gained a lot of attention for their ability to improve industrial processes by creating virtual models of physical systems, leading to increased interest in many industries. The basic need which motivates the use of Digital Twin is to monitor and test a system that changes over time. Digital twins are used to improve the efficiency of machines, conveyors, and other devices. By tracking and modeling changes that take place over time in the virtual representation as a function of input variables to the system, one can approximate the past, present, and future states in order to control the decision process(Stączek et al., 2021). One of the main advantages of Digital Twin is that it gives real-time data that can help in learning, reasoning, and understanding how objects and systems function. It enables users to analyze, model, and optimize a physical object’s performance across its lifespan (Javaid et al., 2023). Moreover, the conceptualization, comparison and collaboration capability of Digital Twin enables us to conceptualize with manufacturing process visually, compare the option and outcome, and then finally collaborate with other manufacturing section (Liau et al., 2018).

In summary, a common understanding of the term Digital Twin (DT) is that it is a high-fidelity, multi-physics digital model designed to replicate both the micro and macro features of a physical system. It mirrors the system's state and behavior while offering the capability to perform simulations in the virtual model and synchronize them with the physical counterpart in real-time. One of DT's defining features is its two-way, real-time communication with the physical system. Overall, the definition of a Digital Twin varies depending on the concepts, context, and specific system for which it is developed (Ramasubramanian et al., 2022).

2.2.1 The Emergence and Evolution of Digital Twin

With the advancement of digital technology, the representation of key factors in product manufacturing has evolved significantly. Initially relying on simple coding and identification techniques, it has now progressed to sophisticated Digital Twin technology featuring virtual reality interaction. This evolution can be categorized into four distinct stages, as illustrated in the Figure 3 (Li et al., 2022).

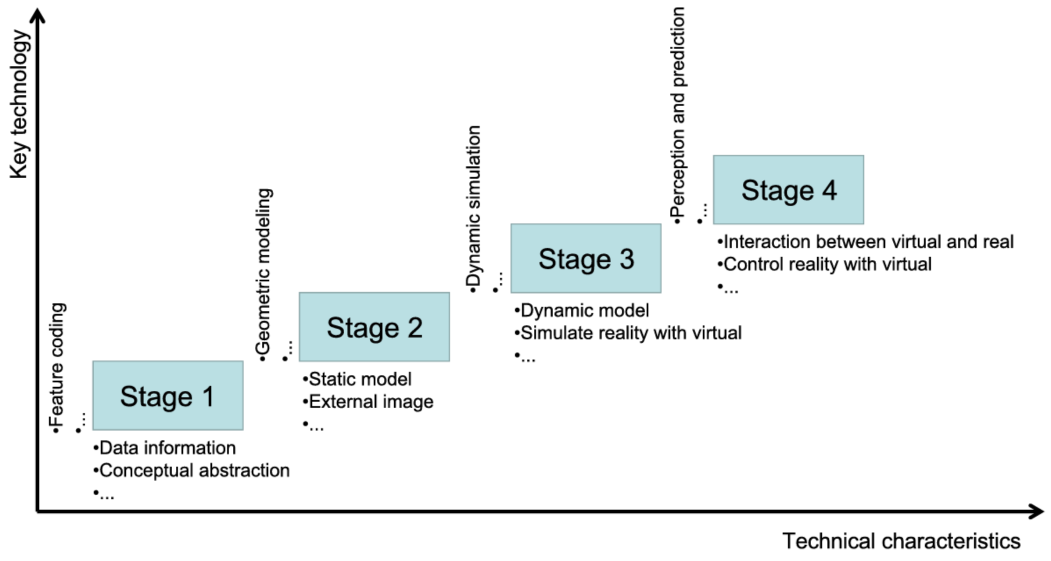


Figure 3-Four Stages in Development and Evolution Digital Technology

Stage 1: Conceptual Abstraction

The first electronic tube computer, developed in 1946, initiated the digital transformation in manufacturing. It enabled high-speed computing by digitizing objects, programming processes, and storing results. Without graphical tools, products were represented through abstract data, using numbers and letters to define their identity and attributes (Li et al., 2022).

Stage 2: External Resemblance

The emergence of CAD technology in the 1960s revolutionized product design by introducing graphical processing in computers. Using interactive graphic tools, designers could create geometric designs digitally. The shift from 2D to 3D technology allowed for intuitive visualization of ideas through 3D models, giving products both a digital identity and static geometric data resembling their physical counterparts (Li et al., 2022).

Stage 3: Simulate Reality with Virtual

The concept of Digital Mock-Up (DMU) introduced the use of digital models to evaluate product functions and performance. DMU represents a complete system or subsystem, reflecting not only geometric designs but also functional and performance characteristics in specific technical fields. By elevating 3D models from static representations to dynamic simulations, DMU laid the foundation for modern Digital Twin technology, enabling models to mimic both the shape and behavior of physical products (Li et al., 2022).

Stage 4: Interaction Between Virtual and Real

The concept of the "twin" was first applied in the manufacturing industry during NASA's Apollo program, where two identical spacecraft were built: one for the mission and the other, called the "twin," remained on Earth. This twin mirrored the spacecraft's status during the mission, enabling real-time simulation and supporting astronaut training and operational decisions. The use of the twin extended beyond the traditional role of prototypes, which were typically confined to the design and manufacturing stages, to include the operational phase of the product. By creating a realistic environment for the twin, it could simulate the actual running state of the spacecraft, offering critical insights and predictions to assist mission success. This pioneering approach laid the foundation for what we now know as Digital Twin Technology, a concept that has since been adopted across industries, including manufacturing, to optimize performance, enhance predictive maintenance, and improve decision-making through real-time data mirroring and simulations. In 2003, Dr. Michael Grieves, a professor at the University of Michigan, introduced the concept of a "virtual digital representation equivalent to a physical product" in a course document on Product Lifecycle Management (PLM). This concept encompassed not only the physical product in the physical space but also the virtual product in the digital space, along with the data, information, and process interfaces connecting the two. By linking the virtual product from the design stage to the entire lifecycle of manufacturing and operation, this idea laid the groundwork for the Digital Twin concept. However, due to the limitations of theoretical frameworks and technological advancements at the time, the Digital Twin concept did not gain widespread attention or development until later years, when advancements in data integration and simulation technologies enabled its full realization and application across industries (Li et al., 2022).

2.2.2 Characteristic and Elements of Digital Twin

The idea of Digital Twin was first introduced by Grieves, who described it as having three main elements: the real physical product, its digital version, and a data link that connects the two [17, 21]. The attributes of the physical systems are collected via sensors and mapped to the virtual system through the connection layer, which facilitates the exchange of data between the physical and virtual layers (Ramasubramanian et al., 2022). This connection is usually made possible by sensors that send data from the physical system to the virtual model. Thanks to this real-time data exchange, the virtual model stays up to date with the actual state of the physical asset and helps improve its performance through continuous monitoring and analysis (Ferko et al., 2022).
Later, Jones et al., in their systematic literature review, expanded this basic idea by identifying 13 important aspects of a Digital Twin**:** (1) Physical Entity/Twin, (2) Virtual Entity/Twin, (3) Physical Environment, (4) Virtual Environment, (5) Realization,(6) State, (7) Parameters, (8) Physical-to-Virtual Connection, (9) Virtual-to-Physical Connection, (10) Twinning, (11) Twinning Rate, (12) Physical Process, and (13) Virtual Process (Jones et al., 2020).
VanDerHorn and Mahadevan (2021) by looking at many definitions, proposed a more general one: a Digital Twin is "a virtual representation of a physical system (and its associated environment and processes) that is updated through the exchange of information between the physical and virtual systems". They also described the concept of Digital Twin in 3 main components and 8 subcomponents, which are summarized in the table 2 (VanDerHorn & Mahadevan, 2021).

COMPONENTS	**DESCRIPTION**
PHYSICAL REALITY	
PHYSICAL SYSTEM	The actual assets or components that is being modelled.
PHYSICAL ENVIRONMENT	Where the physical system of interest resides.
PHYSICAL PROCESSES	The activities or operations the system performs

VIRTUAL REPRESENTATION	
VIRTUAL SYSTEM	Digital version of the Physical system that we are studying like a robot, a machine, or a production line.
VIRTUAL ENVIRONMENT	The digital version of the space around the real system.
VIRTUAL PROCESSES	Simulated processes that mirror real-world operations.
INFORMATION INTERCONNECTION	
PHYSICAL-TO-VIRTUAL CONNECTION	This is the process of sending data from the real world to the Digital Twin to be synchronized with the real system.
VIRTUAL-TO-PHYSICAL CONNECTION	This is the process of sending back the instructions or decisions by Digital Twin to the real system.

Table 2-Three Main Components of Digital Twin

According to the Sharman et al. (2022) analysis they defined the elementary and imperative components of a DT as follows: The elementary components are physical asset, digital asset and information flow that without them the Digital Twin cannot exist. The latter components consist of IoT devices, Data, Machine Learning, Security and Evaluation metrics/Testing which make Digital Twin unique.

2.2.3 Digital Twin in Industry 4.0/5.0

Digital Twin performs unique functions at each phase of a product's life cycle, and many international companies have started to explore the application of it in product design, manufacturing and service (Liu et al., 2022). Using Digital Twin in the design phase brings benefits. It helps designers better understand customer needs, leading to a more accurate design of the product’s look and function. By introducing Digital Twin early in the product development process, companies can keep production costs lower by reducing the need for physical prototypes and avoiding costly redesigns. This early use improves both design precision and cost-efficiency (Alnowaiser & Ahmed, 2023). In the manufacturing stage, Digital Twin is designed to support real-time monitoring and optimization of manufacturing processes (Semeraro et al., 2021). Specifically, it helps to increase competitiveness, productivity and efficiency in several key areas of production systems such as production planning and control, maintenance and layout planning (Kritzinger et al., 2018). By continuously analyzing data from the physical system, they can predict future states, such as potential equipment failures or changes in production quality. This predictive capability allows Digital Twin to suggest adjustments in parameters to improve key performance metrics like efficiency, quality, and output. Additionally, Digital Twin helps with long-term planning by providing forecasts, enabling manufacturers to optimize resource allocation and maintenance schedules to minimize downtime and maximize productivity (Semeraro et al., 2021).

In the manufacturing sector, three primary applications of Digital Twin has been identified, as noted in (Scott, 2020):

1. Supervisory: Digital Twin offers real-time insights into the current state of the corresponding system, which informs and enhances the decision-making process.
2. Interactive: Digital Twin actively monitors the physical twin and can automatically modify one or several parameters when it detects an abnormal situation, ensuring optimal performance.
3. Predictive: Digital Twin not only tracks the existing conditions of the system but also forecasts its future behavior, facilitating the delivery of corrective actions and recommendations through automated systems or human-in-the-loop feedback

The last stage is service in which Digital Twin enhances product utilization and maintenance by analyzing user behavior and preferences, allowing companies to tailor their offerings to individual needs, ultimately leading to improved customer satisfaction and efficient service delivery (Semeraro et al., 2021).
Alamin et al. (2021) also mentioned that Digital Twin in the lifetime of a production line has an important role:

- It acts as a visual prototype of the production line which allows to evaluate its behavior before actual implementation.
- It helps to effective decision making and reduce possible sources of inefficiencies.
- It monitors operation lines at run time and by providing a model of its evolution for failure detection could create a possible optimization (Alamin Khaled et al., 2021).

2.2.4 Diverse Perspectives on Digital Twin Technology: Insights from Leading Global Companies

Digital Twin technology has become very popular and is transforming industries. It helps businesses improve operations, make better decisions, and create value. Many well-known companies are using Digital Twin in their work as it includes some positive changes toward the targeted outcomes like reducing costs and risk, improving efficiency, improving service offerings, security, reliability, resilience, and supporting decision-making (VanDerHorn & Mahadevan, 2021). For example, General Electric has built Digital Twin of jet engine that can achieve power optimization, monitoring and diagnostics of jet engine (Liau et al., 2018). In product design, Dassault has established a 3D experience platform based on Digital Twin, which continuously improves the product design model in the information world and implements it into the physical product improvement, by using the information from user interaction. In manufacturing, Simens

has built a production system model that integrates manufacturing processes based on this new technology. In terms of service, PTC, by establishing a real-time connection between the virtual world and the real world, enabled predictive maintenance and after sales service and support for customers (Liu et al., 2022).
Tabel 3 below provides some examples of how companies apply this technology in the real world (Qi et al., 2018).

General Electric (GE)	By combining physical machinery with analytical techniques, machines can be tested, debugged, and optimized in a virtual environment.
PTC	The PLM process extends into the next design cycle, creating a closed-loop system for product design and enabling predictive maintenance.
Siemens	Using a consistent data model throughout the product lifecycle, some real-world operations are accurately simulated.
Oracle	Virtual models of devices and products simulate the complexities of physical entities, providing insights for practical applications.
ANSYS	By combining advanced simulation with powerful data analysis, it helps enterprises gain strategic insights.
Dassault	Through the 3D experience platform, designers and customers can interact with the product during the design or manufacturing process to understand how it works.
SAP	By building digitized models, product development and innovation are driven by real-time data acquisition and analysis.
Altair	Using advanced virtual simulation technologies, virtual models are created with multiple physical properties, improving the product's characteristics.

Table 3-Companies and Digital Twin

2.3 Autonomous Mobile Robot (AMR) in Logistics

Autonomous Mobile Robots (AMRs) and Automated Guided Vehicles (AGVs) are one of the most important trends in internal logistic (Dobrzańska & Dobrzański, 2025). AGVs are used to perform task which required a lower degree of autonomy, while AMRs develop in dynamic environments, using autonomous navigation (Dobrzańska & Dobrzański, 2025).
The first known AGV which was integrated into warehousing and logistics activities using track-guided magnetic systems, optical sensors, and color bars as guidance technologies was introduced in 1953 (Zhang et al., 2023). The application of AGVs can be categorized in two sections. Firstly, they act as suppliers which transporting and delivering packages in warehouses. In the second section, they are integrated into assembly platforms supporting operation in the production process. By developments in sensors and robot control technology, these systems eventually created a new class of vehicles called Autonomous Mobile Robots (AMRs) (Pizoń et al., 2024b).

AMRs are the next generation of AGVs which are equipped with sensors and algorithms to avoid obstacles and calculate paths based on an internal map (Bamigbala et al., n.d.). Therefore, unlike AGVs, MiR AMRs don't rely on fixed infrastructure such as magnetic strips or rails. They use intelligent mapping and obstacle avoidance to adapt their routes in real-time, offering far greater flexibility and scalability (Mobile Industrial Robots – MiR, n.d.).
A key distinction is the AMR's ability to sense its environment and respond in real-time. It can dynamically plan routes and make decisions autonomously, with these functions seamlessly integrated into a single system. The AMR can calculate the optimal trajectory to reach its target, avoiding obstacles and navigating using maps it constructs on its own. These capabilities make AMRs cost-effective and straightforward to implement in automated logistics systems. Furthermore, AMRs can be controlled via user-friendly interfaces, such as tablets or mobile devices, requiring minimal programming skills. Users can remotely access the robot's dashboard to monitor parameters, check real-time status, and track its position on the map. This flexibility allows users to manage logistics tasks conveniently, regardless of their location. In addition to their autonomous functions, AMRs are equipped with proximity sensors, laser scanners, and cameras to gather environmental data. They are powered by batteries, and understanding battery capacity is essential for efficient operation. Predicting battery usage for specific routes can help enhance the performance and sustainability of AMRs in multi-robot systems, particularly in logistics and warehouse environments (Aliev et al., 2021).

3. State of Art

In this section of this study, a literature review is conducted to identify and analyze the relevant research and papers to the use of AGVs and AMR Vehicles in Intralogistics.
He, et al (2022) studied an energy-efficient open-shop scheduling problem with multiple AGVs and deteriorating jobs. They formulated a multi-objective model with four objectives, aiming to minimize the maximum ending time of all AGVs, the total idle time of machines and AGVs, the total tardiness of jobs and the total energy consumption of machines and AGVs. To solve this problem an improved population-based multi-objective differential evaluation (IMODE) was developed. The result of the experimental results shows that the IMODE is preferable to other well-known multi-objective algorithms at solving the problem being considered (L. He et al., 2022).
Antonelli and Aliev (2022) develop an intelligent monitoring framework for a mobile manipulator and introduce new challenges in managing the recharge cycles as the energy consumption of the mobile manipulator is also related to the overall tasks executed. They implemented an intelligent monitoring system in which gathered the data online and then key performance indicators (KPIs) calculated by Machine Learning (ML) to optimize energy consumption recharging cycles (Antonelli & Aliev, 2022).
Alamin et al. (2021) they believe that Digital Twin in production lines can focus on the management of the production process and monitor and optimize energy consumption and

communications. To do so, they extend models of energy consumption, that allow the monitoring of production line components throughout the production lifetime (Alamin Khaled et al., 2021).
Energy constraints are critical for mobile robots as they usually use batteries with limited capacity. Mei et al. (2006) investigated robot deployment for the coverage tasks by considering timing and energy constraint, which can be conflicting optimization goal. There is a tradeoff: while a robot moves at higher speed, it helps to satisfy timing constraints by completing tasks quickly, high speed simultaneously leads to higher power consumption. To demonstrate this relationship, they built power models for mobile robots and calculated the robots' power consumption at different speeds. According to their studies for the PPRK robot, power increases slowly at low speeds but then increases super-linearly (faster than linear) after a certain speed (around 0.1 m/s) (Mei et al., 2006).
Y. He et al. (2015) There is potentially a significant amount of energy savings that could be realized by reducing the idle energy consumption through better scheduling. They formulated a mathematical model of mixed integer programming. For solving this NP-hard problem, Nested Partitions algorithm was used (Y. He et al., 2015).
Energy-efficient scheduling attempts to decrease energy consumption while keeping the service level the same. In energy-efficient scheduling there are two main strategies, switching off resources while in idle mode and controlling the resources working speed (Speed Scalable).The former helps to balance energy savings from shutting down resources and energy requirements to start and warm them up, the latter has to balance energy consumption and productivity.
Homayouni and Fontes (2021) proposed a mixed-integer linear programming model that effectively solved the energy-efficient job shop scheduling problem and transport resource with speed scalable machines which are used in the production operations and vehicles which are used for job transporting. The numerical analysis shows that by using the speed scalable resources, energy consumption can decrease (Homayouni & Fontes, 2021).

4. Development of Digital Shadow and Digital Twin for MiR100

4.1 Motivation

Industry 4.0 technologies bring fundamental changes to manufacturing, including internal logistics. Among these technologies, in today's dynamic manufacturing environment, the integration of AMRs in this sector enables automated independent material transport within production plants, which eliminate many manual and repetitive operations.
AMRs, such as those developed by MiR, are proof of the successful integration of hardware and software innovation. Several studies highlight their potential to reduce in energy consumption in logistics operations. However, to fully harness the benefits and advantages of them, optimizing their energy usage is necessary.

In the evolution to Industry. 5.0, there is a significant shift toward human-centric systems, sustainability, and resilience. As a result of this shift, manufactures are trying to organize internal logistics that not only enhance efficiency but also support sustainability goals.
In this context, the performance of AMRs is influenced by several operational factors. While it might seem that higher speed improves productivity and guarantees better performance , this is not always the case.
In fact, factors such as bottlenecks, battery usage and the system balance should be considered to optimize operational efficiency. Since AMRs rely on battery; the task implementation and scheduling depend on the battery capacity and the rate of energy consumption.
Therefore, understanding how the key operational parameters of AMRs such as speed, acceleration, and payload handling affecting power consumption and energy consumption is necessary to optimize the process efficiency and energy consumption. Although researchers demonstrated that lower speeds in AMRs lead to slower battery depletion and lower power consumption, this relationship for MiR100 has not been verified. Therefore, to address this gap the first phase of study was conducted to study battery behavior and power consumption in both high speed and low speed. Furthermore, based on findings, this study proposed a framework that dynamically adjusted speed to contribute to the optimization operational KPI and energy consumption.

4.2 Description of Framework

To study battery behavior of MiR100 at different speeds, a Digital Shadow framework is proposed. In this framework as it can be seen in Figure 4, data flows from the physical entity to the virtual part, but there is no feedback or control on the opposite side. MiR100 real-time battery data is acquired with the use of Node-RED, which allows monitoring of the battery statues.

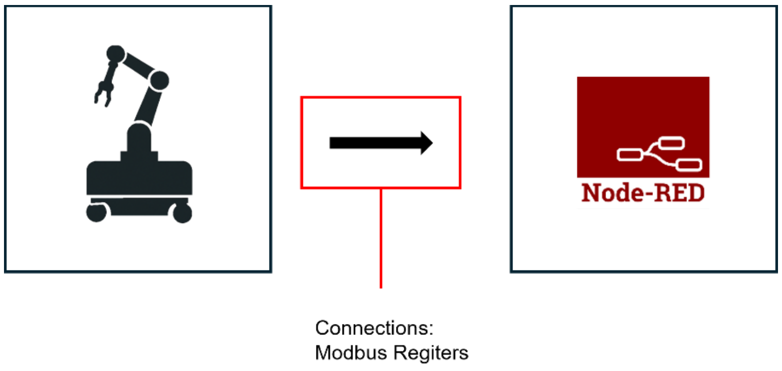


Figure 4-Digital Shadow Framework

In the next step the Digital Twin framework is introduced to optimize the MiR100 performance. Figure 5 shows the direction if this framework Unlike Digital Shadow, which only collects data in a one-way, Digital Twin enables two-way interaction between the physical entity and virtual counterpart. With this framework, the FlexSim simulation is not only updates the real-time data from MiR100, but also send back control signals such as adjusting the MiR100 speed in response to the real-time data. Moreover, throughout the process, FlexSim continuously monitors and calculates key performance indicators which will be introduced and discussed in the 5 chapter

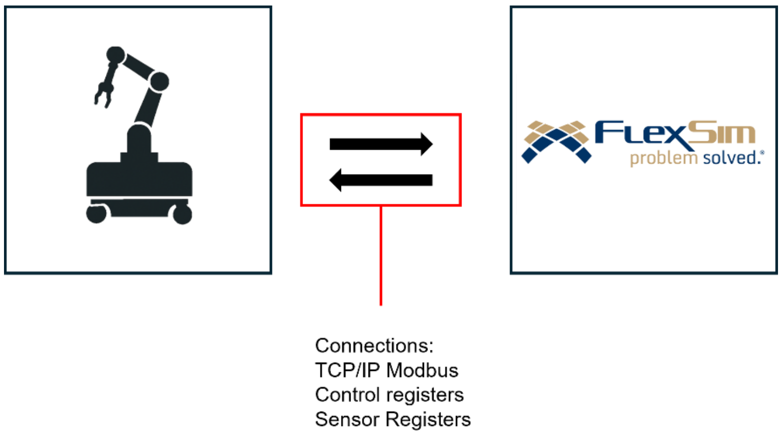


Figure 5-Digital Twin Framework

4.3 Node-RED

Node-Red is a flow-based programming tool, which provides the opportunity to collect, transform and visualize real-time data easily. Users with any background can use it as it has low-code nature. can be used to create JavaScript functions.

The Flow is the main way to organize nodes. Each flow has a specific name and description. The flows in Node-RED are managed by the different type of “nodes”. Also known as 'black-boxes', where each of them has a specific function or purpose. When data is transmitted to the node, it processes data according to the defined function and then it passes that data to the subsequent node in the flow. Nodes are connected by wires. The following Figure 6 shows the Node-Red flow with nodes and connections.

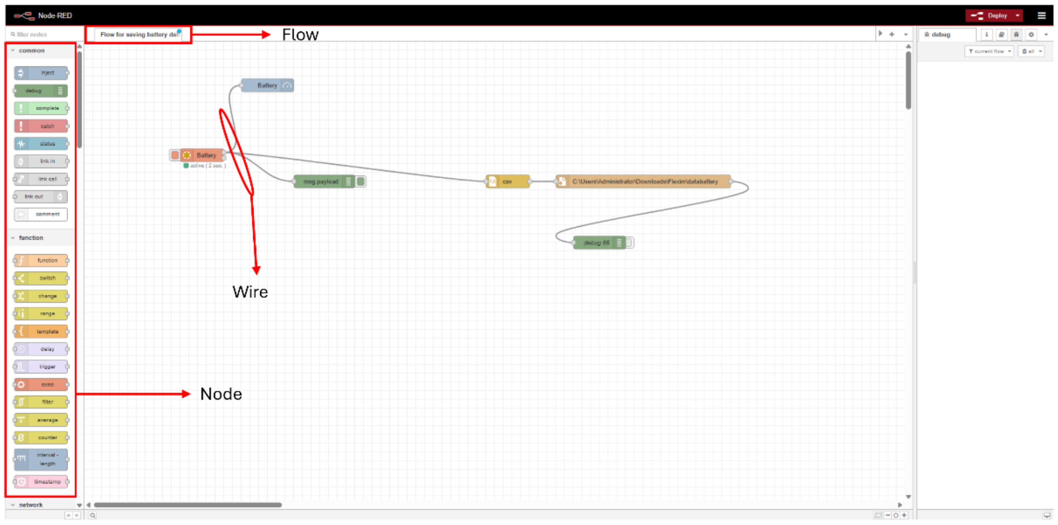


Figure 6-Node-RED Environment

4.4 Physical Representation

The system under study is an autonomous mobile robot (MiR100): The MiR100 is the mobile base of the system. It has two motorized wheels for movement and four caster wheels for stability. This robot uses sensors and mapping to move safely and efficiently in different environments. Once the environment is mapped, the robot autonomously navigates within the designated map which is defined on the portal. Figures 7 and 8 demonstrated the MiR100 robot and corresponding map where it is located in the Mind4Lab at Politecnico di Torino.

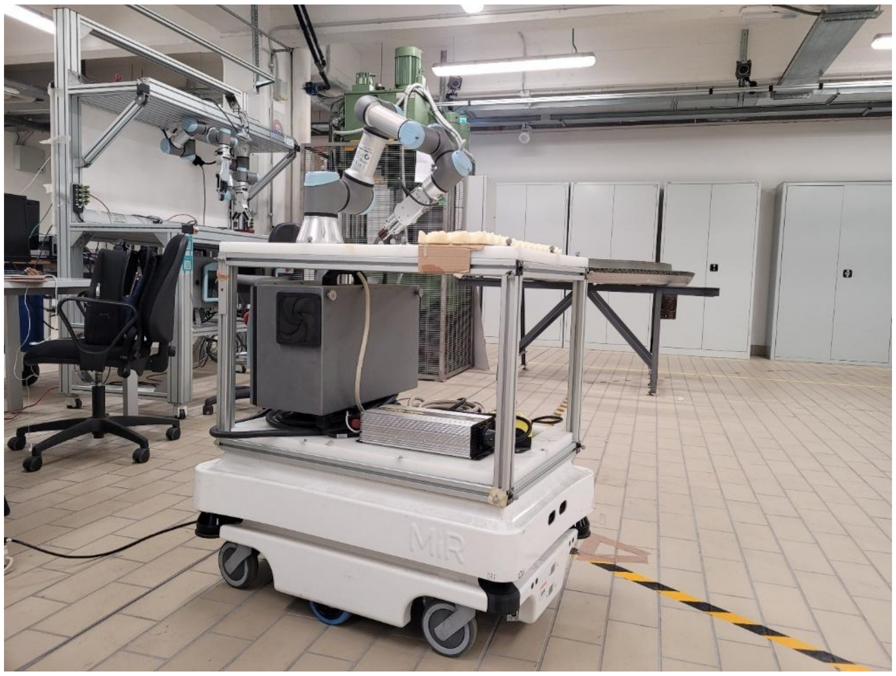


Figure 7-Physical Representation (MiR100)

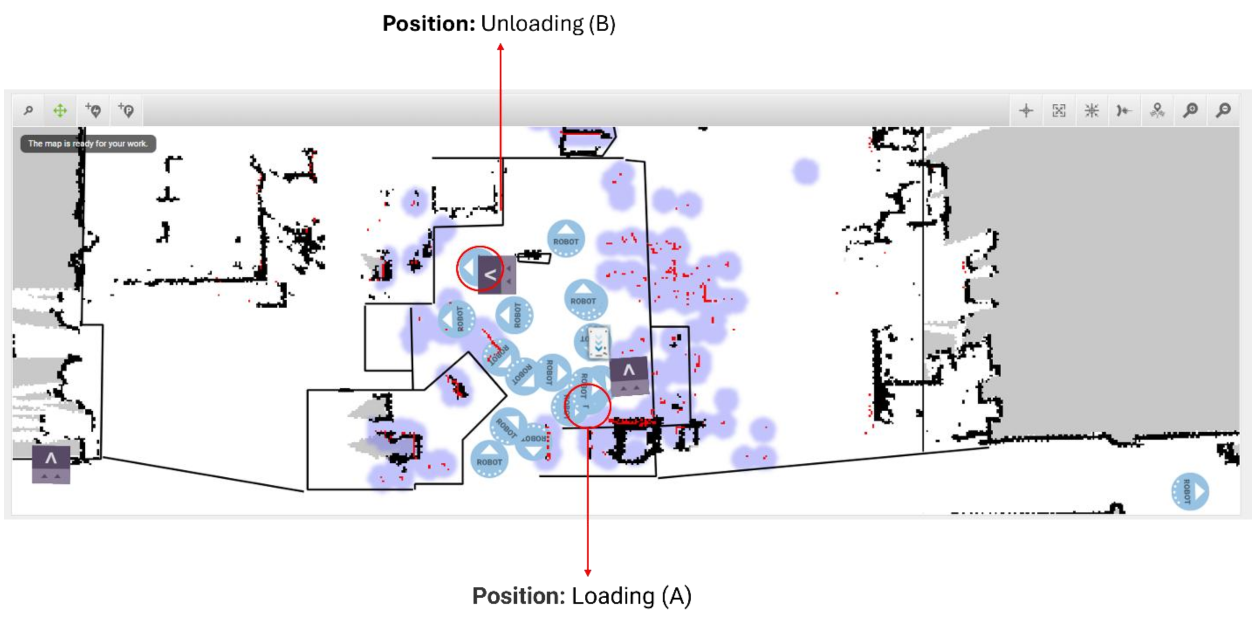
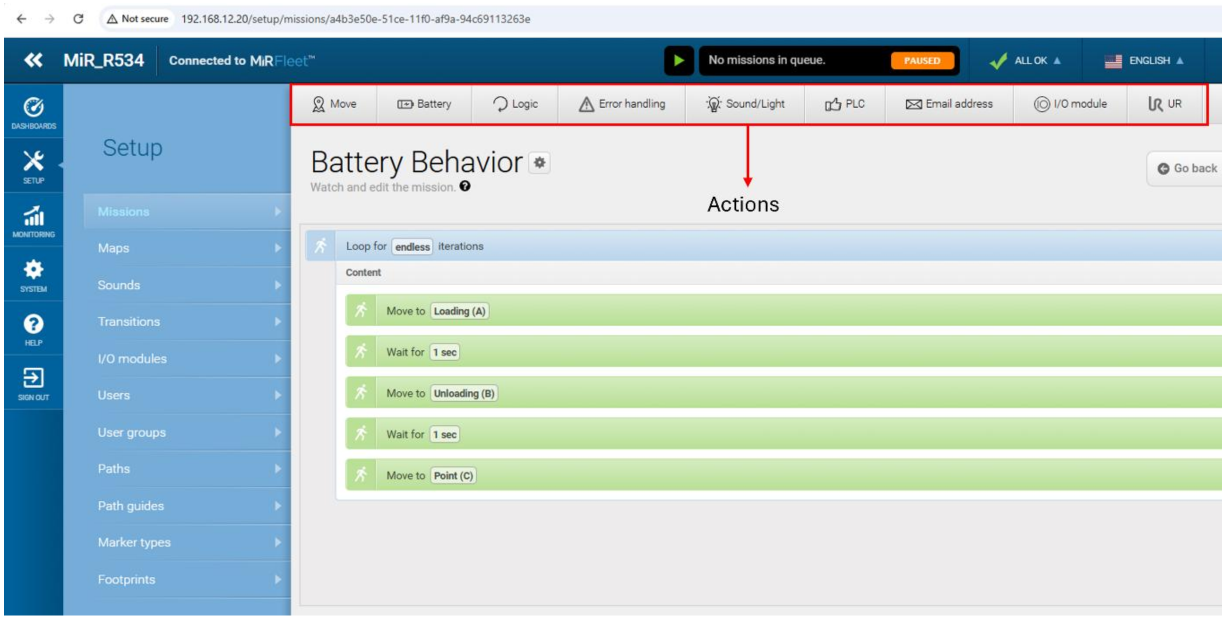


Figure 8-Live map of the Mind4Lab environment

The MiR100 features a remarkable payload capacity of up to 100 kg and a maximum linear speed of 1.5 m/s, making it an essential tool for transporting various materials, components, and finished products efficiently. Users can control and create missions through a web-based interface.

A mission consists of a series of actions, such as move, logic, docking, and sounds, that user can pick from the menus in the top bar. The main action groups are *Move*, *Battery*, *Logic*, *Error handling*, *Sound/Light*, *PLC*, *Email address*, *I/O module*, *Safety system*, *Cart*, *Shelf* and *UR*, as illustrated in Figure 9. The actions are executed in a top-to-bottom order.



Battery Behavior

Watch and edit the mission.

Loop for endless iterations

Content

Move to Loading (A)

Wait for 1 sec

Move to Unloading (B)

Wait for 1 sec

Move to Point (C)

Figure 9-MiR100 Platform

In this thesis to build a mission the following actions are used:

Move

1. Move: This action defines a specific position on the map which the robot should move to.
2. Planner Settings: This action allows the robot to set the desired speed while it runs its mission.

Logic

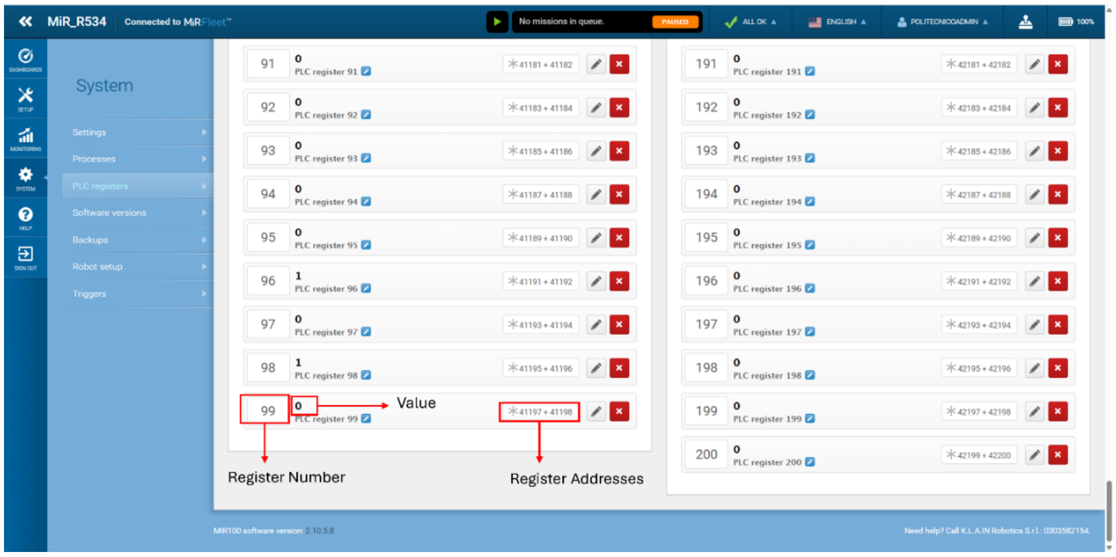
1. Wait: This action pauses the mission for a given period of time.
2. Loop: This action provide the opportunity for the robot to repeat a mission for a specified number of times or indefinitely.
3. While: This action check battery level, number of pending missions, PLC registers or input from I/O modules and then while these conditions are true actions or missions should be executed. In this thesis this action is used just for PLC register, therefore it checks continuously if the register is set to certain value. As an example if it is defined While Register = 1, it means that the robot pause until register 6 has the value 1.

PLC registers

The PLC registers feature is accessible through the System menu. In this section users can create new registers or change the default value of them. Registers 1-100 are 32 bit integers, that is whole positive or negative numbers and Registers 101-200 are 64 bit floating point numbers, that is positive or negative decimal numbers.
It is important to know that one PLC register uses two holding register addresses. This register address should be inserted in the Modbus control register and Modbus sensor register.
The following actions are commonly used in relation to PLC registers:

1. Set PLC Register: This action is used to set a value in a register.
2. Wait For PLC Register: This action is used to pause the mission until a specified value is found in a given PLC register.

In Figure 10, shows the MiR portal interface for registering PLC values. The right side shows float registers, while the left side illustrates for integer registers.



91	0	PLC register 91	41181 + 41182
92	0	PLC register 92	41183 + 41184
93	0	PLC register 93	41185 + 41186
94	0	PLC register 94	41187 + 41188
95	0	PLC register 95	41189 + 41190
96	1	PLC register 96	41191 + 41192
97	0	PLC register 97	41193 + 41194
98	1	PLC register 98	41195 + 41196
99	0	PLC register 99	41197 + 41198
191	0	PLC register 191	42181 + 42182
192	0	PLC register 192	42183 + 42184
193	0	PLC register 193	42185 + 42186
194	0	PLC register 194	42187 + 42188
195	0	PLC register 195	42189 + 42190
196	0	PLC register 196	42191 + 42192
197	0	PLC register 197	42193 + 42194
198	0	PLC register 198	42195 + 42196
199	0	PLC register 199	42197 + 42198
200	0	PLC register 200	42199 + 42200

MIR100 software version: 2.10.5.8

Need help? Call K.L.A.IN Robotics S.r.l.: 0303582154.

Figure 10-PLC Registers

4.5 Virtual Representation

FlexSim is a powerful object-oriented simulation software which is used to model, simulate, and analyze complex systems. It is particularly useful for industries such as material handling, healthcare, warehousing, mining, logistics, and more. It is both powerful and user-friendly.
For advanced users, the software supports deeper customization through programming languages like FlexScript and C++. FlexSim also provides features for performing "what-if" analysis without needing to change things in the real world, which can help identify optimal process improvements and anticipate potential bottlenecks in system workflows.

4.5.1 3D Model

The software allows users to develop 3D models to visualize and optimize their processes. They are more intuitive and realistic, so they reduce the number of errors that could be made in the process of building a simulation model. Moreover, as they are more visual, they can sometimes be more effective than graphs, statistics when there is need to communicate with key decision makers who do not necessarily have technical or engineering background (*FlexSim Documentation: Welcome*, n.d.).
One of the core features of FlexSim is its intuitive drag-and-drop interface, which allows users to build models by simply placing predefined 3D objects into the simulation. It also offers robust logic-building tools with minimal programming requirements, making it accessible even to users with limited technical knowledge. Moreover, FlexSim is increasingly recognized for its ability to simulate various industries beyond traditional manufacturing, such as healthcare, mining, and transportation, by handling both time and spatial dynamics in real-world operations FlexSim. This

makes it a versatile tool for improving efficiency, reducing costs, and increasing operational sustainability (FlexSim, n.d.).
In FlexSim, objects serve as the fundamental building blocks for creating simulation models. They can be broadly categorized into Flow Items, Fixed Resources and Task Executers. In the Simulation Model, Flow Items interact with Fixed Resources and Task Executers. Among the standard resources available for model construction, the most widely used are Source, Queue, Processor, and Sink. These components represent the entry point, waiting zone, processing unit, and exit point of flow items in a simulated system. These components can be strategically combined to build dynamic simulations that model processes across a spectrum of complexity.
Figure 11 presents an example of the layout in FlexSim simulation software.

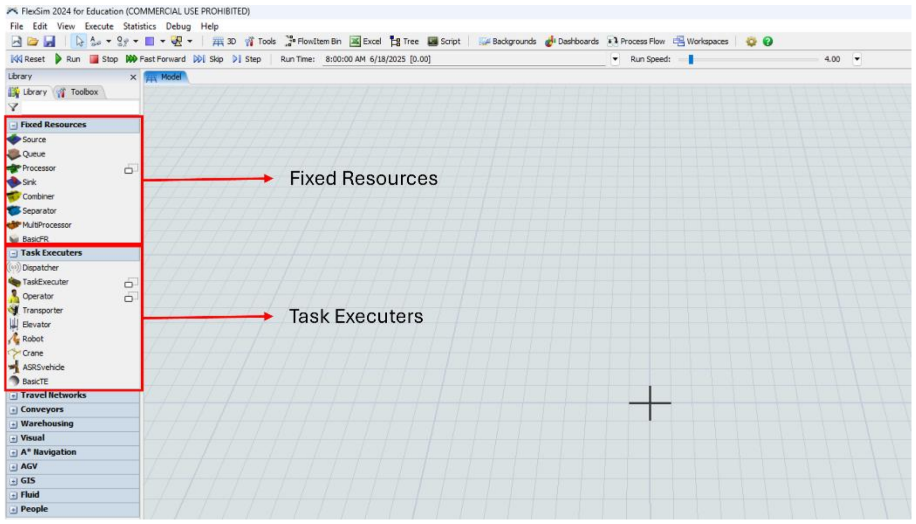


Figure 11-3D Model

Flow items

Flow items are objects that move through the simulation model, typically from one station (often a fixed resource) to another. They can represent various entities, such as products, customers, paperwork, or parts, and are used to simulate the movement of items throughout the system. While flow items are initially represented as boxes by default, their appearance can be customized to resemble people, products, or other shapes as needed.

Fixed Resources

Fixed resources are objects that remain stationary within the model. Typically, they interact with flow items in the simulation by storing or modifying them. These resources can represent various processes or stages in the system, such as processing stations, machines, or storage locations.

- *Source*: It generates Flow Items in FlexSim, serving as the process flow's starting point. Items can be created at a set rate, on a schedule, or through other methods.
- *Queue*: It temporarily holds Flow Items before processing. In FlexSim, queues can model physical storage or virtual buffers, with configurable exit rules (e.g., FIFO, LIFO, or custom logic). They also support capacity limits, priority settings, and routing controls.
- *Processor*: It (or workstation) is where Flow Items undergo transformation or processing. In FlexSim, it simulates tasks like assembly, machining, or inspection, with configurable processing times, resource needs, and schedules.
- *Sink*: It serves as the exit point for Flow Items, removing them from the simulation upon arrival. It can represent completed products, waste materials, or any process outputs.

Task Executor

Task executers are objects that move within the 3D model and perform specific tasks, such as transporting flow items or operating machines. The most common type of task executor is the operator, which typically represents an employee performing tasks in the simulation model.

4.5.2 Process Flow Model

The Process Flow model in FlexSim constructs the logical framework of a simulation by focusing on key procedures and decision-making processes rather than visual representation. Acting as the "brain" of the simulation, it defines how objects move, interact with resources, and undergo processing based on logical rules and conditions. In the process flow objects are represented as tokens (small green circles), abstract entities that can symbolize items, people, orders, calls, or grouped pallets. When a simulation model runs, the token moves through the process flow activities, executing each activity's logic. To build logic in FlexSim, there are predefined activity blocks, which allows users to develop logic. The following Figure shows an example of process flow in FlexSim.

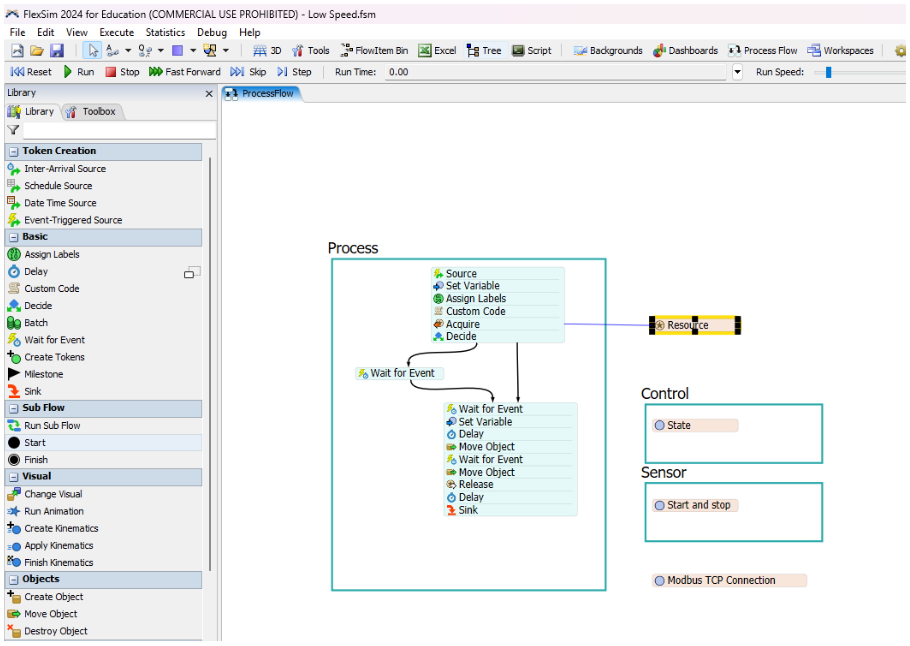


Figure 12-Process Flow Model

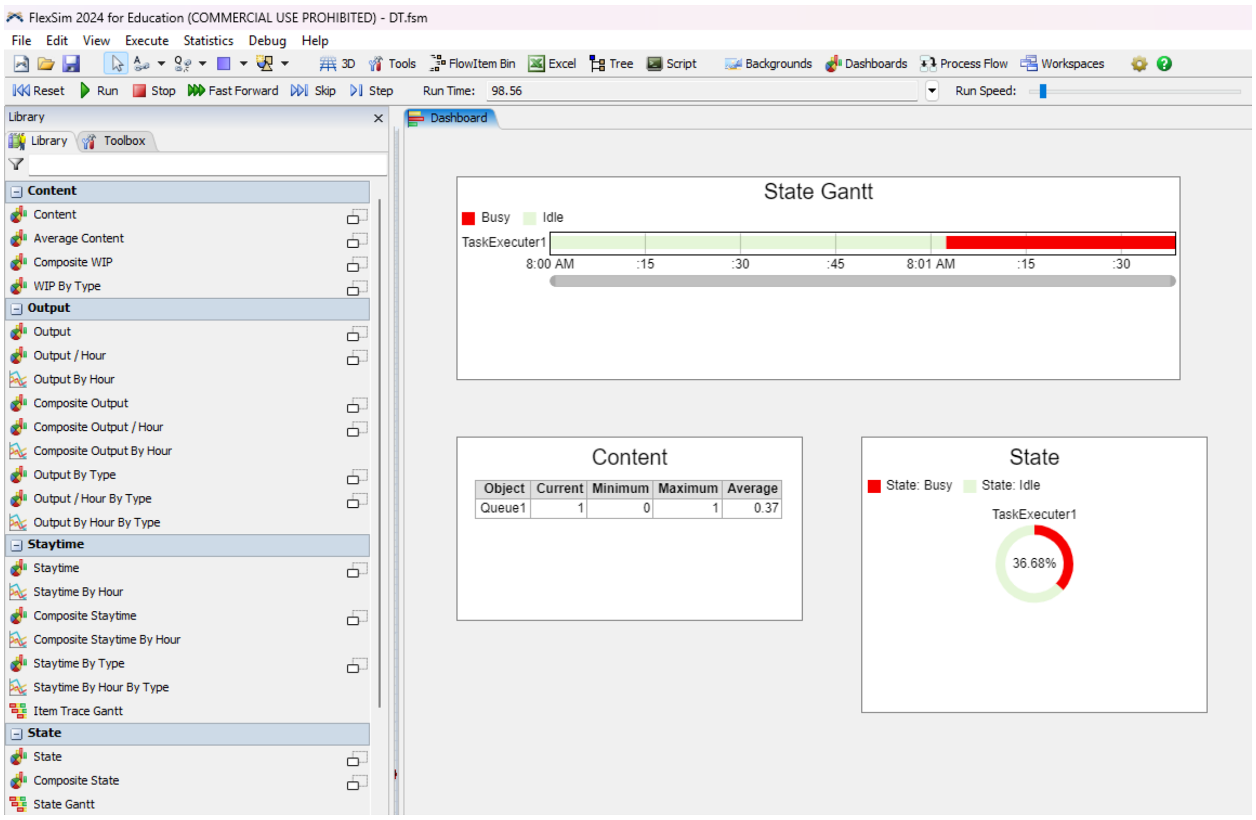
In this thesis to build the process flow the following activity blocks are used:

- *Source*: Similar to sources in the 3D model in process flow, these activities create the token. There are different types of source including Inter-Arrival, Schedule, Date Time, and Event-Triggered Sources. In this study, the last one is used to create tokens in response to an event while simulation is running. When the event occurs, the token will be created.
- *Assign label*: Labels can be used to attach data to tokens and store important data about various object or store the emulation variables.
- *Custom code*: It allows users to create a custom behavior through predefined custom codes or writing their own code in FlexScript. When a token enters this block activity, it will evaluate the user-defined code and then release to the next block activity. In this study, a predefined custom code is used to add assigned label values to the global tables, enabling collecting all data.
- *Resource:* This shared asset represents a limited supply of resources that can be acquired and then released. In this study, MiR100 is considered as a resource. Resources are useful for modeling timing constraints, where multiple tokens are needed to use a limited number of shared assets.
- *Acquire:* It is used to acquire a resource when it is needed during process flow. Once the resource has been acquired, nothing else can acquire it until the acquiring token has released it.
- *Release:* This activity block is used to release or return a resource during a process flow when there is no longer needed. When a resource has been released, it increases the availability of that resource by a specified amount.

- *Decide:* Based on conditions that users define, it sends a token to one of two or more possible activities. If the specified condition is met, tokens are routed accordingly. The condition has various forms, including condition-based logic, case selection, time-based routing, percentage splits, statistical distributions, global table lookups, and random or round-robin allocation.
- *Wait for event:* it will hold the token until a certain event occurs. Once that event occurs, the token is released and continues the process flow.
- *Delay:* It holds a token for a certain amount of time.
- *Move Object*: Moves an object or multiple objects to another place in 3D simulation model. Therefore, the process flow and 3D model can be synchronized together.
- *Sink:* This activity block destroys tokens, removing all data stored on those tokens.
- *Variable:* The value of variables in process flow can be assigned through emulation connections or Emulation Variable. Therefore, it is possible to reference an Emulation Connection and an Emulation Variable or create them. To reference them first it is necessary to create them on the emulation through toolbox, which will be explained in part 4.5.4 Emulation tool.
- *Set Variable:* This block assigns the value of a Variable shared asset.

4.5.3 Dashboard

Dashboards are blank windowpanes that users can customize with anything that they need to access during a simulation run. One of the most common uses of dashboards is providing many easy-to-use charts. While simulation run, 3D model and process flow activities collect a standard set of statistics continuously and it can be represented by charts. As a result, dashboards provide various data visualization tools which allow users to track and evaluate different key performance indicators (KPI). Figure 13 shows an example of a dashboard with various charts in the FlexSim environment. To extract the real-time state of the MiR100, indicating whether the robot is available or currently transporting, a custom code was implemented within the process flow model. This logic, which enables accurate calculation of the robot's utilization, is detailed in Section *5.3.4: Logic within Process Flow in FlexSim*.



Object	Current	Minimum	Maximum	Average
Queue1	1	0	1	0.37

Figure 13-Dashboard

4.5.4 Emulation Tool

The emulation tool is used to connect FlexSim simulation to real-world control systems like Programmable logic controllers (PLC) or external software systems, which enable the simulation model to receive and send real-time signals. The emulation tool is accessible from the toolbox as it is shown on Figure 14. This tool provides different types of protocols, including OPC, UA, OPC DA and Modbus. Moreover, multiple simultaneous connections can be supported to integrate multiple PLCs or client-server systems. When real-time communication with a PLC is not needed, it can be possible to uncheck the “active” option. Therefore, the data would be retrieved from the theorical simulation instead of the external source.

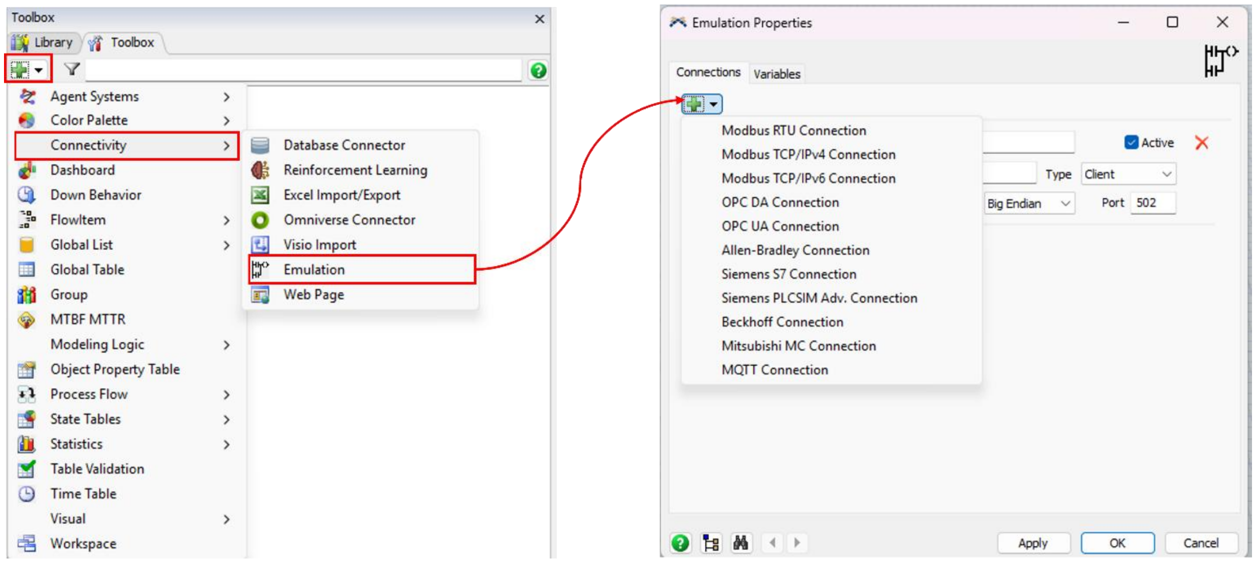


Figure 14-Emulation Tool for Creating Connection

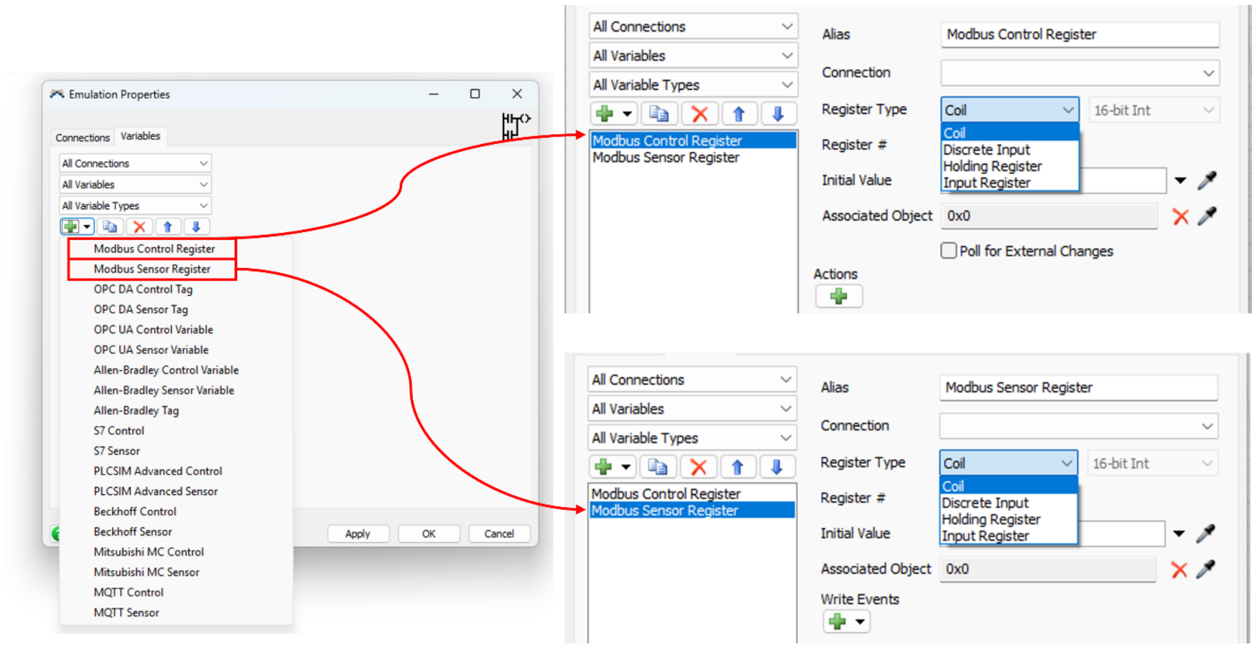
Emulation allows users to define variables for each of these connections. The term variable refers to any inputs or outputs that are received or sent by the PLC. There are two types of variables that can be added, Control Register (output) and Sensor Register (input). The properties of these variables are described in detail below, and Figure 15 illustrates the properties of emulation, control and sensor variables.

Control Register (PLC Outputs)

Control is the output of the PLC. Based on the inputs (values) it receives, the PLC will issue controls that will tell the system what actions to take in the simulation model. In FlexSim, control variables are used to read data from the PLC, server or data variable. If "Poll for External Changes" option is enabled, FlexSim will regularly ask for changes from the external system using events (polling). Moreover, control variables can also be connected to an object of 3D model.

Sensor Register (PLC Inputs)

Sensor is the inputs to the PLC. Sensors provide environmental data from inputs to the PLC and then the PLC will decide what actions to take based on that data. Sensor Variables are the variables which send data from the simulation FlexSim to the PLC.



Alias: Modbus Control Register
Connection:
Register Type: Coil, 16-bit Int
Register #
Initial Value
Associated Object: 0x0
Poll for External Changes

Alias: Modbus Sensor Register
Connection:
Register Type: Coil, 16-bit Int
Register #
Initial Value
Associated Object: 0x0

Figure 15-Modbus Register

In the following part a detailed explanation of each Modbus Register Type in FlexSim is provided:

- *Coil:* The coil register in Modbus is a binary writable, which means that the simulation or controller can set a value 1 (On) or 0 (Off) to activate or deactivate the physical system.
- *Discrete Input:* This type of register is a binary input register which is used to read only the current state of external devices or systems.
- *Holding Register*: This type of register is writable numeric registers that can store complex data typically 16-bit or 32-bit integers. Therefore, it can have a range of numeric values rather than just 1 or 0.
- *Input Register*: This type of register is similar to input register, but it reads numeric values instead of binary one.

5. Case Study

5.1 Experiment Implementation

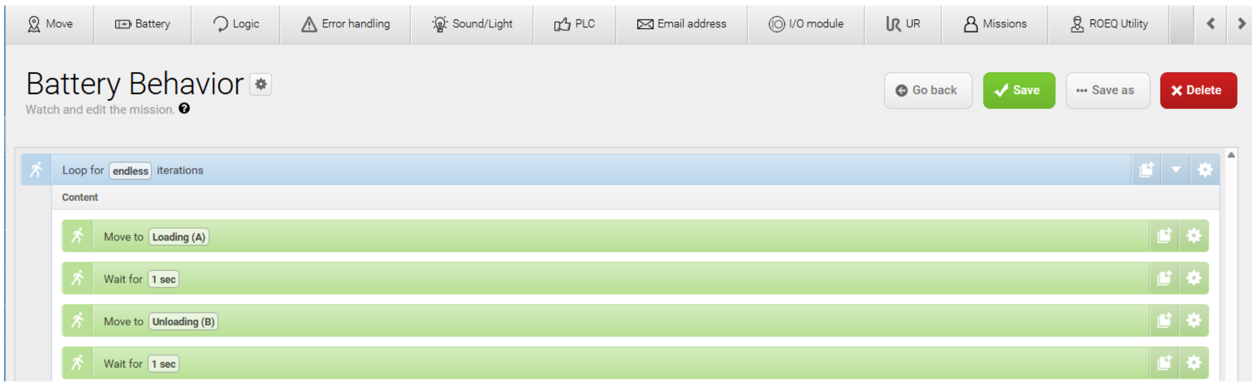
The experiment is divided into two main phases: the first phase is preliminary phase, which focuses on developing a data acquisition system under digital shadow framework. By collecting and storing data under digital shadow, it becomes possible to perform more in-depth analysis and gain a clearer understanding of the robot's status and performance over time. The second phase is designed for the optimization operational KPI and energy consumption under Digital Twin framework.

5.2 Preliminary Phase of Study

This phase was conducted without implementing a Digital Twin since it is a data-driven analysis of the battery draining of MiR100 robot operating under different speed conditions. To collect the relevant data Node-RED platform is used. The objective of this study is to understand the relationship between the speed of the MiR100 robot and its battery consumption, particularly focusing on how varying speeds affect the battery. It has been assumed that no failures and interruptions occur, therefore it represents a static and ideal framework in which it operates. In the first test, the robot operated at a lowest speed (0.1 m/s), and in the second test, it operated at a highest speed (1.5 m/s).

5.2.1 Logic within MiR100

The Battery Behavior mission is for analyzing the MiR100 battery consumption. As shown in Figure 16, in this mission, the robot moves to **Loading (A)**, wait for 1 second, move to **Unloading (B)**, wait for 1 second. This mission is continuing in endless iterations.



Battery Behavior

Watch and edit the mission.

Loop for endless iterations

Content

Move to Loading (A)

Wait for 1 sec

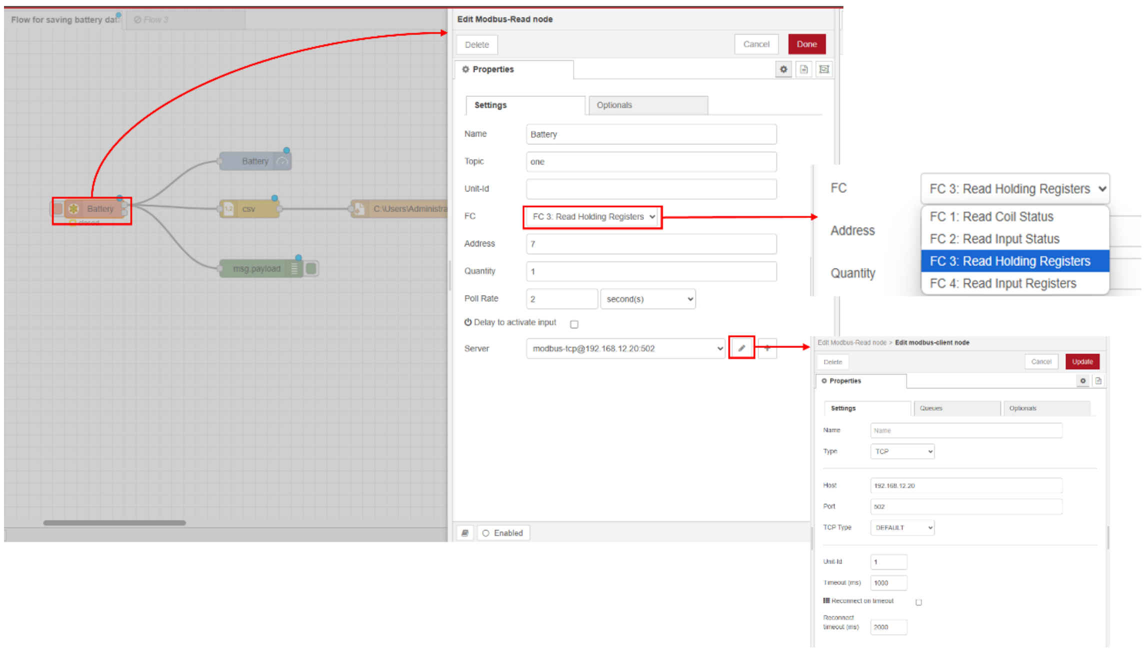
Move to Unloading (B)

Wait for 1 sec

Figure 16- MiR100's Actions for Monitoring Battery Behavior

5.2.2 Logic within Node-RED

The Node-RED is using Modbus-Read Node to acquire battery data from the MiR100. In the Figure below it can be seen how this node is created and configured. The panel allows users to choose the IP address and the port, which are important parts of the connection. According to the MiR100 robot's Modbus TCP data sheet, the battery level (remaining charge) is available in holding register 4008 with type 16int. The Node-RED is using zero-based indexing, therefore register 4008 corresponds to register address 7. Every 2 seconds the Node-RED connects to MiR100 at IP 192.168.12.20. It reads the battery level from Modbus register 7. Then battery values will be saved into CSV file.



Edit Modbus-Read node

Name: Battery
Topic: one
Unit-Id:
FC: FC 3: Read Holding Registers
Address: 7
Quantity: 1
Poll Rate: 2 second(s)
Delay to activate input: ☐
Server: modbus-tcp@192.168.12.20:502

FC: FC 3: Read Holding Registers
- FC 1: Read Coil Status
- FC 2: Read Input Status
- FC 3: Read Holding Registers
- FC 4: Read Input Registers

Edit Modbus-Read node > Edit modbus-client node

Name: Name
Type: TCP
Host: 192.168.12.20
Port: 502
TCP Type: DEFAULT
Unit-Id: 1
Timeout (ms): 1000
Reconnect on timeout: ☐
Reconnect timeout (ms): 2000

Figure 17-Logic within Node-RED for Battery Behavior

5.2.3 Discussion

After collecting and saving data with a resolution of two seconds, a deep analysis is conducted. The first column of data set indicates the timestamp, while the second one represents the corresponding battery level. The results revealed a clear difference in battery drain between the two tests, which is visually represented in Figure 18. When operating at high speed (1.5 m/s), the robot consumed battery at a much faster rate, leading to a steeper decline in battery percentage over time. Conversely, the low-speed test showed slower, more gradual battery consumption. To more conclusively show that speed impacts battery consumption.

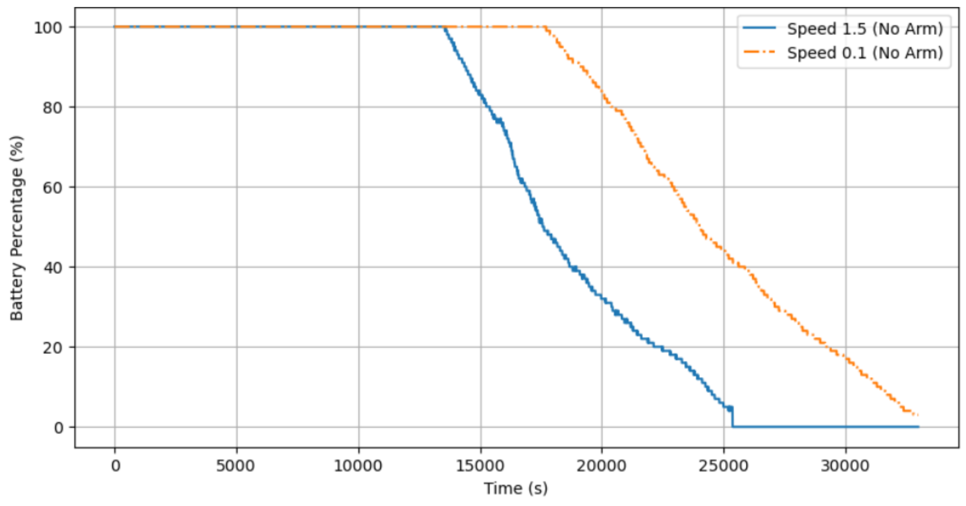


Figure 18-Battery Discharge Over Time

5.3 Second Phase of Study

To evaluate how Digital Twin can enhance the performance of production systems in terms of operational and sustainability three scenarios are defined. During the experiment all data related to current and voltage until completion of 10 items were monitored in order to assess power consumption and energy consumption under three scenarios. The first two scenarios run at the lowest speed 0.1 M/S and at the highest speed 1.5 M/S, while the last scenario is dynamic speed adjustment that has been developed to respond to the limitation in the warehouse storage capacity. This solution aims to solve a key challenge in internal logistics regarding the combination of satisfying operational KPIs and sustainability paradigm.

The last scenario is proposed because using only high speed is not a good option in this specific thesis study. Although it reduces transportation time, at the same time it increases power and energy consumption, which means the battery will be drained faster. Moreover, when the floor storage becomes full, the robot must wait until at least one item goes out. Since the transportation time at high speed is shorter than low speed, the floor storage fills up faster, and the robot ends up being idle more often, just waiting for one item to go out.

On the other hand, transporting all the items at low speed reduces power and energy consumption, but increases the cycle time of each transportation and overall cycle time for transporting all the items. Therefore, there is a trade-off between energy efficiency and operational performance.

To address this, a dynamic logic scenario was introduced that adjusts the robot's speed based on the real-time situation. This solution is specifically designed to the condition of this study, where the number of items is limited and the storage has capacity limitations and other assumptions that are going to be taken in account is going to be explained in section *5.3.1 Assumptions and Characteristics*. Therefore, the third scenario aims to strike a balance between minimizing energy consumption and maintaining acceptable operational performance.

Once all 3 scenarios were conducted, the values of Current 1, Current 2 and voltage were continuously recorded. These metrics are the robot's electrical performance while completing the mission. To calculate average power consumption, first the sum of Current 1 and Current 2 was computed at each data point to represent the total Current. Then, power consumption (in Watt) was calculated by multiplication of the total current (in Ampere) and the corresponding voltage value (in Volt). The average power consumption was obtained by taking the mean of all calculated power values. Finally, to calculate energy consumption (in watt-hours) the average power consumption is multiplied by the total transportation time. Energy consumption is an important factor in evaluating sustainability, as it not only depends on power drawn by robot but also on the duration of operation. Time is an important factor as the time required for transporting items at lower speed is greater than the time required using a higher speed, directly affecting energy usage.

Before explaining these scenarios in detail, it is necessary to get in depth with assumptions, logic within 3D model and process flow in FlexSim, the mission design within MiR100 and connections that enable real-time synchronization between the physical and virtual systems.

5.3.1 Assumptions and Characteristics

All three scenarios follow the same assumptions and characteristics:

1. The inter-arrival time is not deterministic, it is dependent on uniform distribution between 60 and 150 seconds, therefore it varies by item by item, but is the same within three scenarios.
2. A total of 10 items is processed in each scenario.
3. Each scenario followed the same sequence of missions to ensure consistency in data collection and comparability of results.
4. A maximum of 5 items is allowed in the floor storage at any given time.
5. Each item will remain in floor storage for up to 500 seconds.
6. The pick-and-place operations are performed manually by a human operator, maintaining a constant working pace throughout the process.
7. To ensure synchronization between the FlexSim model and the robot, a 5-second delay block is implemented within the process flow.

5.3.2 Logic within 3D Model in FlexSim

The following Figure shows the 3D model of virtual system, consisting of Source, Queue, Task Executor, Floor Storage and Sink. In the following the properties of each component will be discussed.

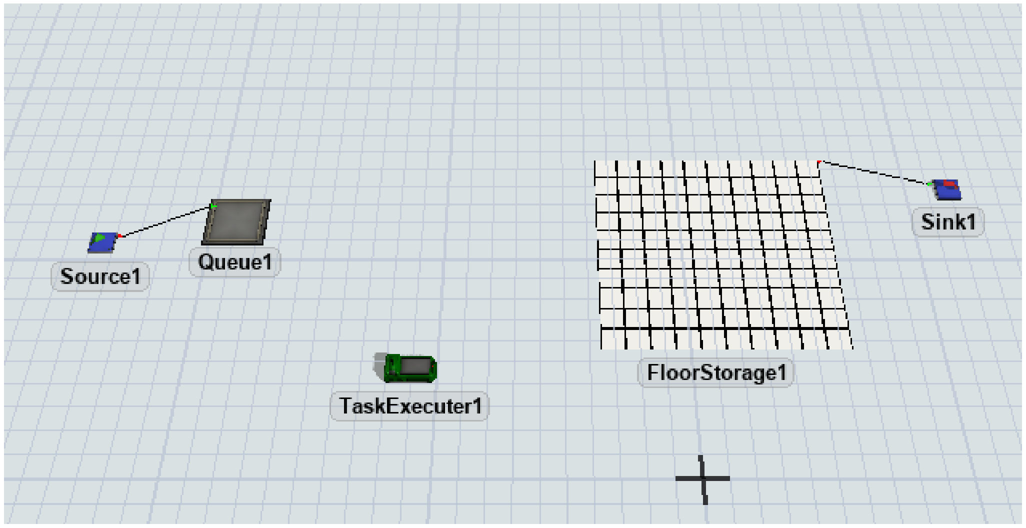
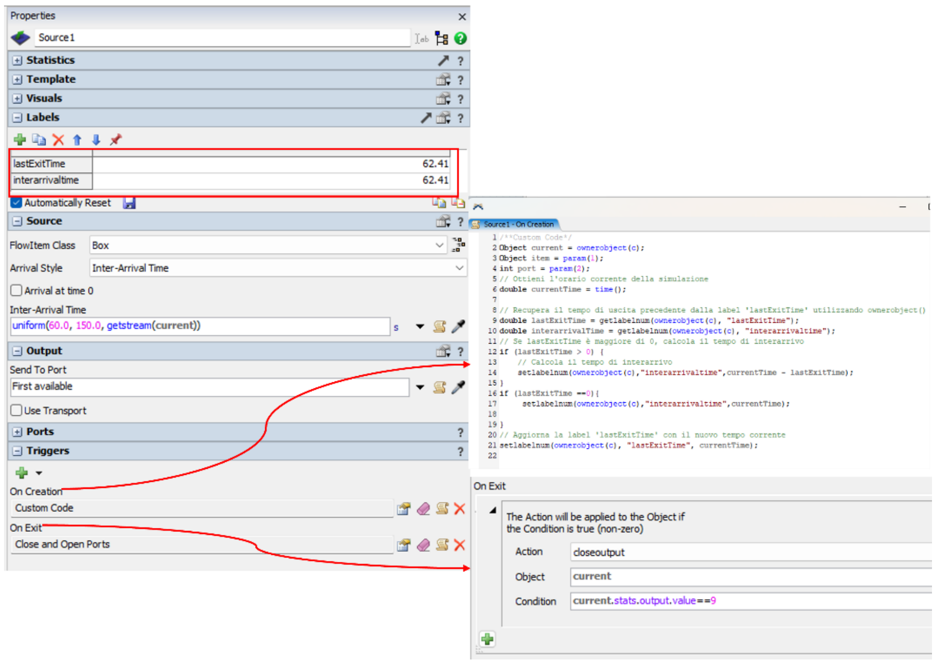


Figure 19-Logic within 3D Model

The process starts with the arrival of 10 items to the system with uniform distribution. The inter-arrival time is distributed uniform between 60 seconds and 150 seconds. Figure 20 illustrates the properties of source. To track the timing of arrivals two labels “lastExitTime” and “interarrivaltime” are defined. The former records the time when the previous item was created, while the latter shows the inter-arrival time.

In the triggers session, a custom code “on creation” is defined, which is responsible for updating those labels each time a new item is created. The “on exit” trigger is used to control when the source should not create items. This triggers check the condition how many items are currently in the output port, once the condition is true, then this action will close the output port, which will not all additional items exit the source. Therefore, with this logic the total number of created items are limited to 10.



lastExitTime	62.41
interarrivaltime	62.41

Automatically Reset

FlowItem Class: Box
Arrival Style: Inter-Arrival Time
Arrival at time 0
Inter-Arrival Time: uniform(60.0, 150.0, getstream(current)) s
Send To Port: First available
Use Transport
On Creation: Custom Code
On Exit: Close and Open Ports

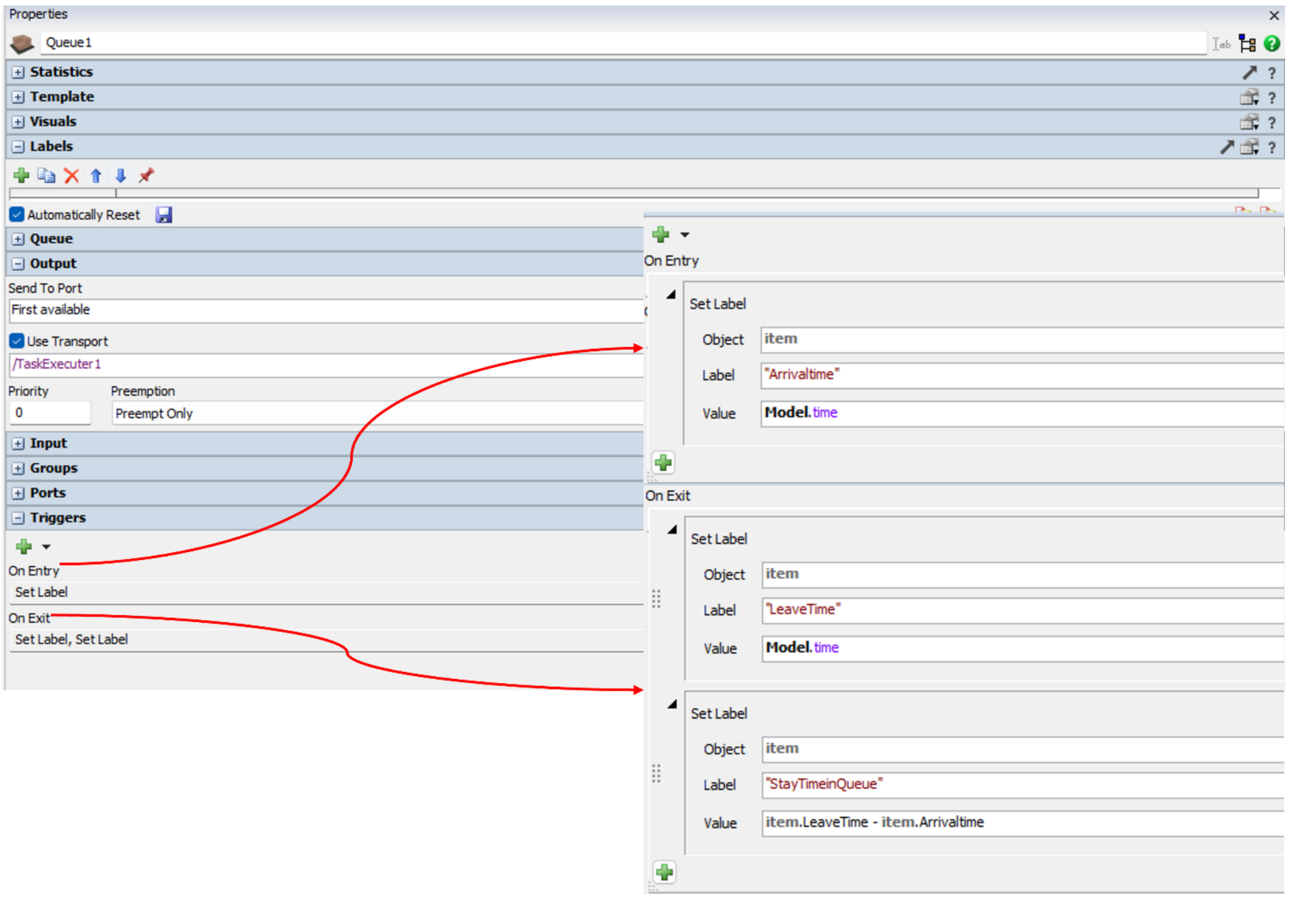
```
/**Custom Code*/
Object current = ownerobject(c);
Object item = param(1);
int port = param(2);
// Ottieni l'orario corrente della simulazione
double currentTime = time();

// Recupera il tempo di uscita precedente dalla label 'lastExitTime' utilizzando ownerobject()
double lastExitTime = getlabelnum(ownerobject(c), "lastExitTime");
double interarrivalTime = getlabelnum(ownerobject(c), "interarrivaltime");
// Se lastExitTime è maggiore di 0, calcola il tempo di interarrivo
if (lastExitTime > 0) {
    // Calcola il tempo di interarrivo
    setlabelnum(ownerobject(c),"interarrivaltime",currentTime - lastExitTime);
}
if (lastExitTime ==0){
     setlabelnum(ownerobject(c),"interarrivaltime",currentTime);

}
// Aggiorna la label 'lastExitTime' con il nuovo tempo corrente
setlabelnum(ownerobject(c), "lastExitTime", currentTime);
```

The Action will be applied to the Object if
the Condition is true (non-zero)

Action: closeoutput
Object: current
Condition: current.stats.output.value==9

Figure 20- Source Properties

Items temporarily are stored in Queue1 while waiting for the MiR100 to transport them to floor storage one by one. In the triggers section, “on entry” the label “Arrivaltime” is assigned to item with the value of Model.time, to capture exact time of arrival to the queue. When the item exits the queue (when it is picked up by task executor) two labels are defined in “on exit” trigger: “LeaveTime” with the value of Model.time and “StayTimeinQueue” which is calculated by subtracting the Arrivaltime from LeaveTime. These two labels can later be viewed by clicking on the box when it is being transported by the task executor, which in this case represent MiR robot. The following Figures illustrate queue and box properties.


Properties

Queue1

Statistics

Template

Visuals

Labels

Automatically Reset

Queue

Output

Send To Port: First available

Use Transport: /TaskExecuter1

Priority: 0 | Preemption: Preempt Only

Input

Groups

Ports

Triggers

On Entry: Set Label

On Exit: Set Label, Set Label

On Entry

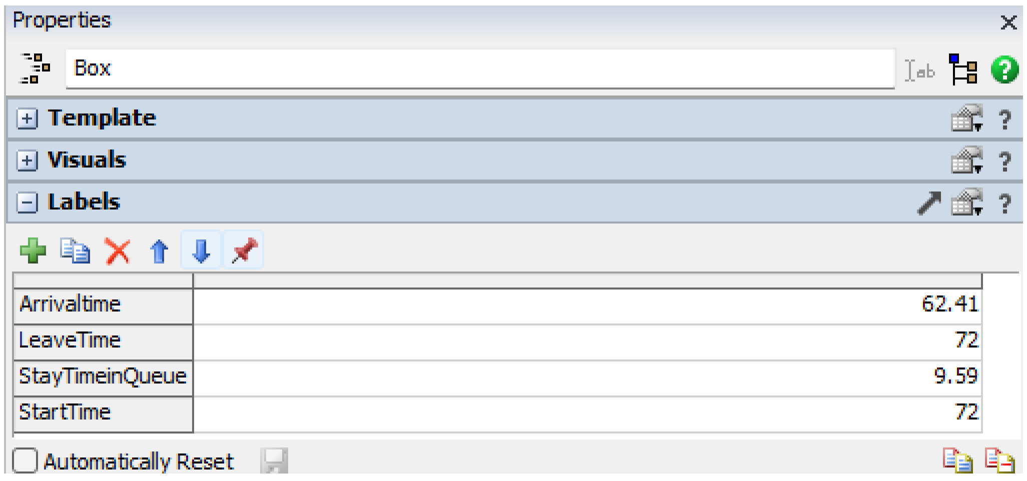
Set Label
- Object: item
- Label: "Arrivaltime"
- Value: Model.time

On Exit

Set Label
- Object: item
- Label: "LeaveTime"
- Value: Model.time

Set Label
- Object: item
- Label: "StayTimeinQueue"
- Value: item.LeaveTime - item.Arrivaltime

Figure 21-Queue Properties


Properties

Box

Template

Visuals

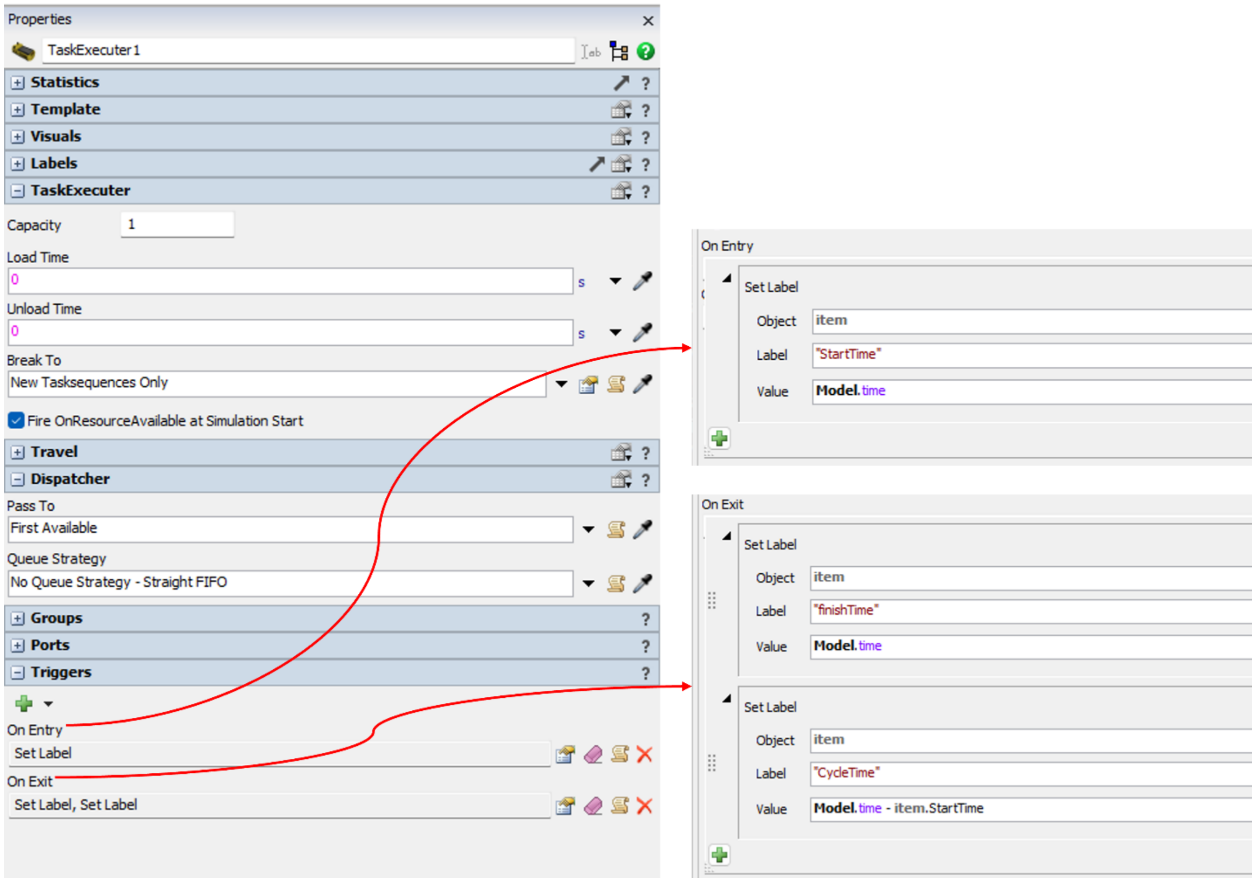
Labels

Arrivaltime	62.41
LeaveTime	72
StayTimeinQueue	9.59
StartTime	72

Automatically Reset

Figure 22-Box Properties

The boxes one by one are transporting with task executor. In the triggers section, “on entry” the label “StartTime” is assigned to item with the value of Model.time, to capture exact time transportation begins. When the item exits the task executor (when it reaches floor storage) two labels are defined in “on exit” trigger: “finishTime” with the value of Model.time and “CycleTime” which is calculated by subtracting the StartTime from finishTime or model.time. This helps to keep track of the transportation duration. Figure 23 illustrates the configuration of the Task Executor's triggers which are used to calculate the cycle time of transported items.



Properties
TaskExecuter1
Statistics
Template
Visuals
Labels
TaskExecuter
Capacity: 1
Load Time: 0 s
Unload Time: 0 s
Break To: New Tasksequences Only
☑ Fire OnResourceAvailable at Simulation Start
Travel
Dispatcher
Pass To: First Available
Queue Strategy: No Queue Strategy - Straight FIFO
Groups
Ports
Triggers
On Entry: Set Label
On Exit: Set Label, Set Label

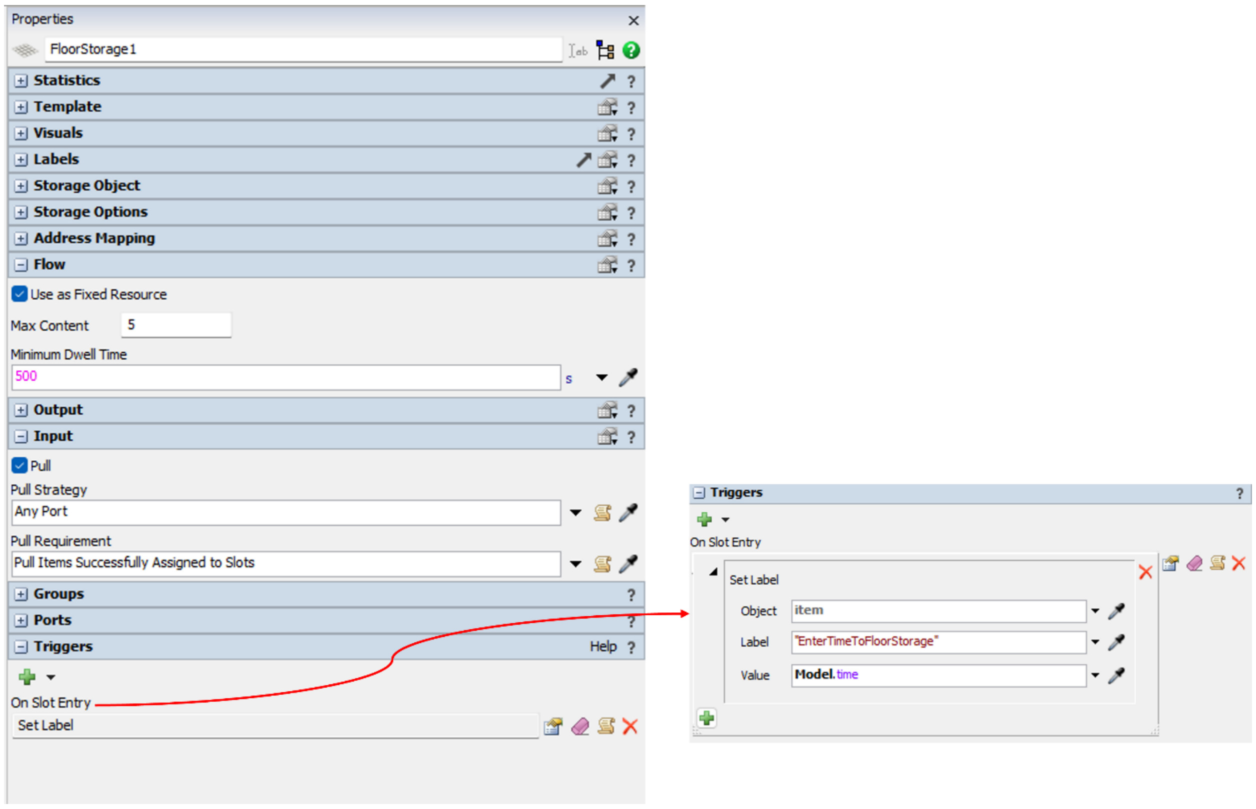
On Entry
Set Label
Object: item
Label: "StartTime"
Value: Model.time

On Exit
Set Label
Object: item
Label: "finishTime"
Value: Model.time

Set Label
Object: item
Label: "CycleTime"
Value: Model.time - item.StartTime

Figure 23- Task Executor Properties

The transported items are kept in FloorStorage for 500 seconds. In the triggers section, “on entry” the label “EnterTimetoFloorStorage” is assigned to item with the value of Model.time, to capture exact time entering the floor storage. Figure 24 illustrates the trigger configuration and the assigned label used to track the item's entry time into the storage, while Figure 25 demonstrates all the labels that an item has while it is kept in floor storage. After remaining there for 500 seconds, the items are removed from the system through a sink.


Properties

FloorStorage1

Statistics

Template

Visuals

Labels

Storage Object

Storage Options

Address Mapping

Flow

Use as Fixed Resource

Max Content 5

Minimum Dwell Time

500 s

Output

Input

Pull

Pull Strategy

Any Port

Pull Requirement

Pull Items Successfully Assigned to Slots

Groups

Ports

Triggers Help

On Slot Entry

Set Label

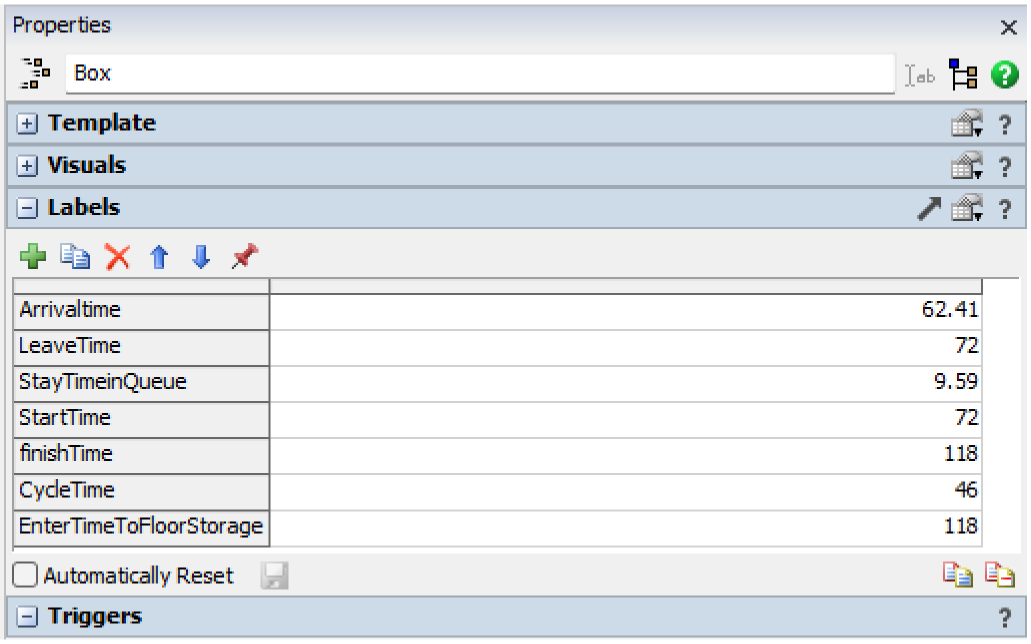
Triggers

On Slot Entry

Set Label

Object	item
Label	"EnterTimeToFloorStorage"
Value	Model.time

Figure 24- Floor Storage Properties


Properties

Box

Template

Visuals

Labels

Arrivaltime	62.41
LeaveTime	72
StayTimeinQueue	9.59
StartTime	72
finishTime	118
CycleTime	46
EnterTimeToFloorStorage	118

Automatically Reset

Triggers

Figure 25- Box Properties at the end of the model

5.3.3 Connections

To understand how the logic within process flow works, it is necessary to explain the connection between the physical system (MiR100) and virtual model (Flexsim). The connections are made through the variables in the process flow. In this study five emulation variables and one emulation connection is used as illustrated in Figure 26.

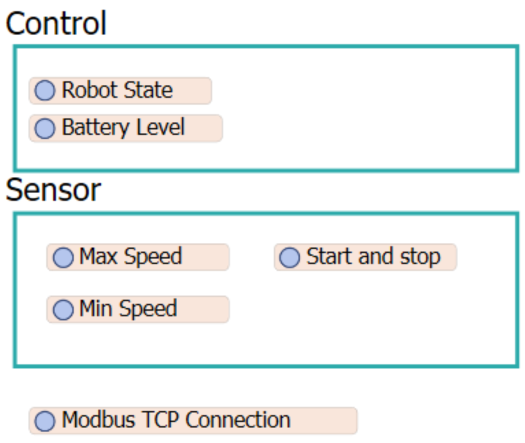
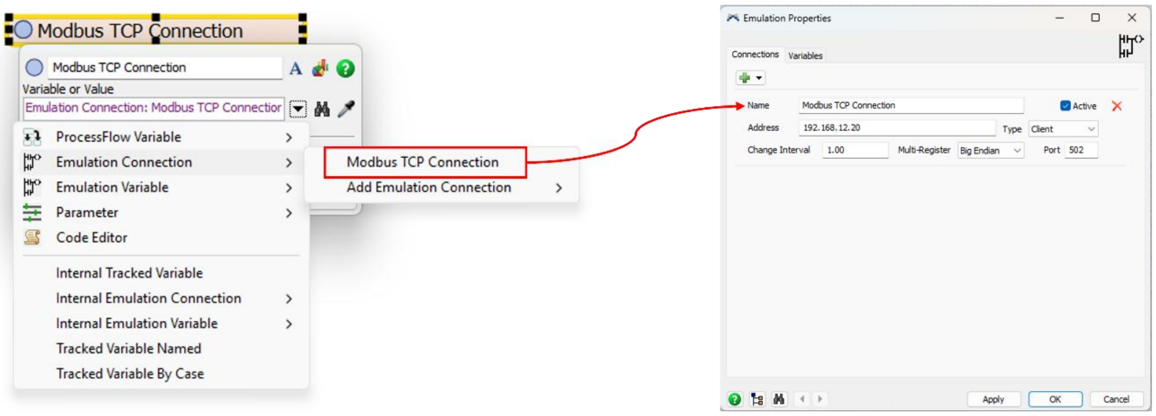


Figure 26-Process Flow Variables

In this study, Modbus TCP protocol is used to make this communication. As shown in the Figure below, the connection is configured by setting the IP address of the MiR100 robot (192.168.12.20), client type and communication port 502. The configuration steps are defined within the emulation properties as described in section *4.5.4 Emulation Tool*. Once this connection is built, in the process flow it is possible to add a variable and link it to the previously defined Modbus TCP Connection through the emulation connection toolbox as it illustrates in Figure 27.



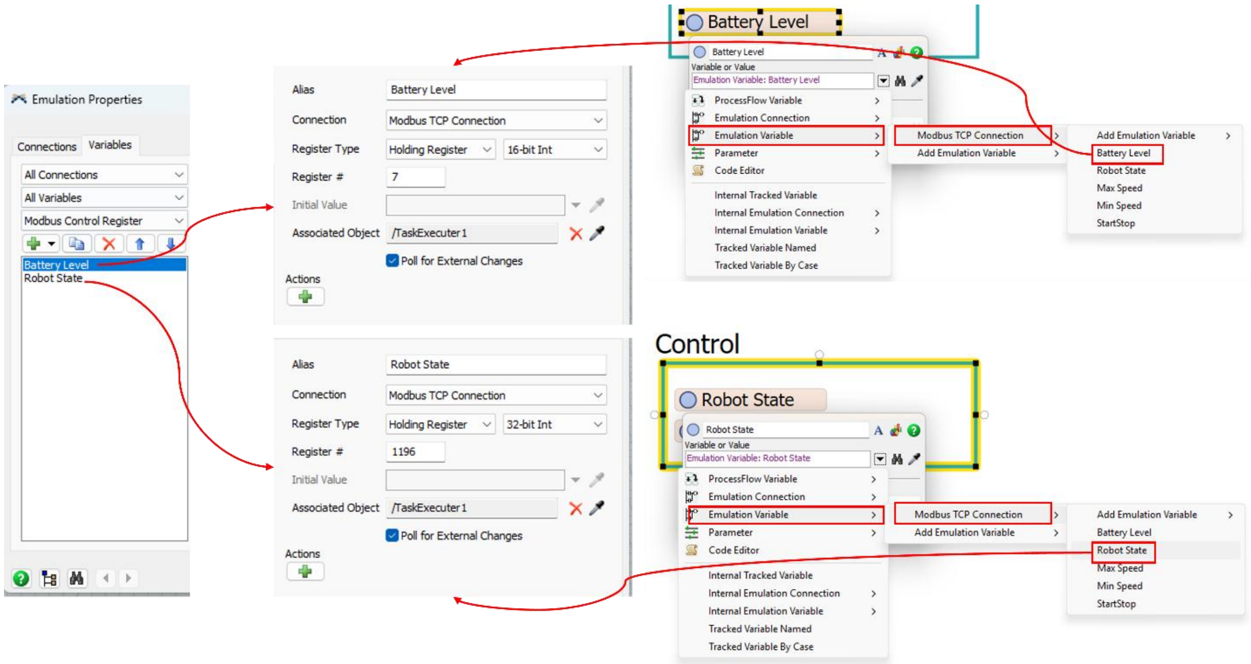
Name: Modbus TCP Connection ✓ Active
Address: 192.168.12.20 Type: Client
Change Interval: 1.00 Multi-Register: Big Endian Port: 502

Figure 27-Emulation connection

To complete this connection, control and sensor variables should be defined.
Control Variables are used to receive real-time data from the MiR100, specifically the battery level and robot state. Figure 28 shows Modbus control register and control variable in detail.

- *Battery Level variable*: In this setup, PLC register 4008 within MiR100 portal corresponds to register 7 in FlexSim, which helps to take the real-time battery level.

- *Robot State variable*: PLC register 99 within MiR100 portal corresponds to register 1196 in FlexSim. When PLC register 99 has a value of 0, this condition is mirrored in FlexSim through control variable 'Robot State' which takes the value 0. This indicates that MiR100 is available at the loading point (Ponit A). In contrast, if the register 99 is 1, it means the robot is busy and control variable 'Robot State' takes 1 in FlexSim. Control variable 'Robot State' is responsible for synchronization between the physical part (MiR100) and virtual model (FlexSim).


Emulation Properties

Connections | Variables

All Connections
All Variables
Modbus Control Register
Battery Level
Robot State

Alias: Battery Level
Connection: Modbus TCP Connection
Register Type: Holding Register | 16-bit Int
Register #: 7
Initial Value:
Associated Object: /TaskExecuter1
☑ Poll for External Changes
Actions

Alias: Robot State
Connection: Modbus TCP Connection
Register Type: Holding Register | 32-bit Int
Register #: 1196
Initial Value:
Associated Object: /TaskExecuter1
☑ Poll for External Changes
Actions

Battery Level
Variable or Value: Emulation Variable: Battery Level

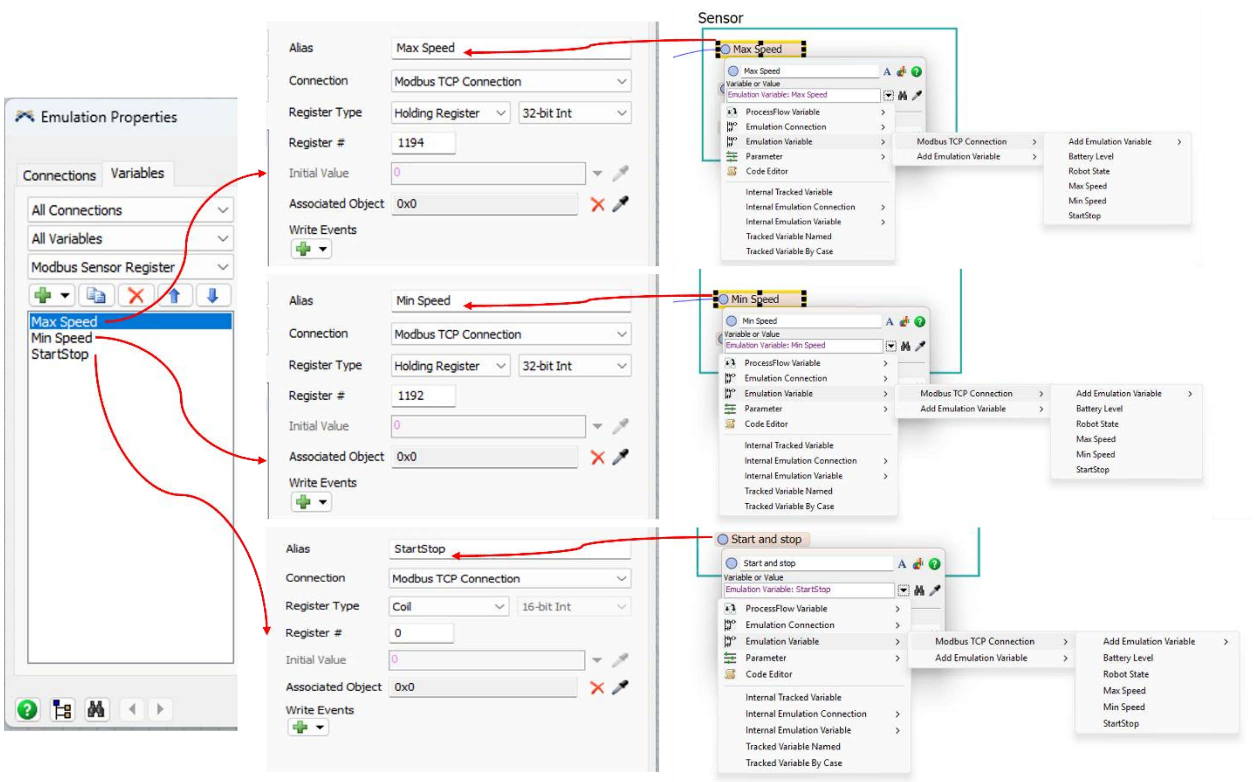
Control
Robot State
Variable or Value: Emulation Variable: Robot State

Figure 28-Modbus Control Register & Control Variable

Sensor Variables are used to send real-time data from virtual system (FlexSim) to physical system (MiR100) specifically Start and Stop, Max Speed and Min Speed. Figure 29 presents the configuration of the Modbus sensor register along with the sensor variable in detail.

- *Start and Stop variable*: Register 1 in MiR100 portal corresponds to register 0 in FlexSim. This variable is configured as a Coil type register, which means that it is a binary signal register (0 or 1). It is responsible for the activation and deactivation of the MiR100. Unlike variables that are initialized at simulation start, in this set up, this is done through the Set Variable block in the process flow. Assigning a value of 1 through this block to this sensor variable sends start command to MiR100.

- *Max Speed variable*: This variable is sensor variable in FlexSim to control the speed of the MiR100 in real-time by sending signal to it. Register 98 in MiR100 portal corresponds to register 1194 in FlexSim. This variable is configured as a Holding Register. In a process flow the Set Variable block will set the value of 1 to the Max Speed variable when a certain condition is met. This will act a command signal to the MiR100 to operate at high-speed mode.
- *Min Speed variable*: This variable functions similar to the Max Speed variable, but it is used to command the MiR100 to operate at low speed. This variable is configured as a Holding Register, Register 97 in the MiR100 portal and Register 1192 in FlexSim. When in the process flow, a Set Variable block assigns a value of 1 to the Min Speed variable under certain conditions, the MiR100 will receive it as command to reduce its speed.



Emulation Properties
Connections | Variables
All Connections
All Variables
Modbus Sensor Register
Max Speed
Min Speed
StartStop

Alias	Max Speed
Connection	Modbus TCP Connection
Register Type	Holding Register, 32-bit Int
Register #	1194
Initial Value	0
Associated Object	0x0
Write Events	

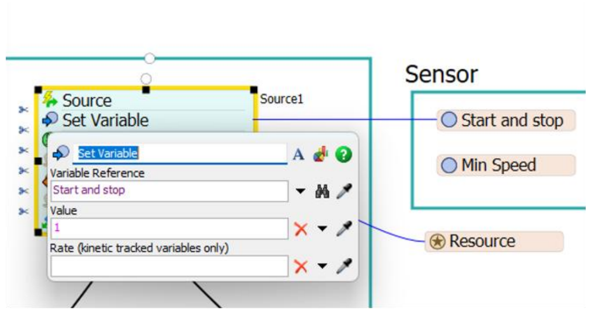
Alias	Min Speed
Connection	Modbus TCP Connection
Register Type	Holding Register, 32-bit Int
Register #	1192
Initial Value	0
Associated Object	0x0
Write Events	

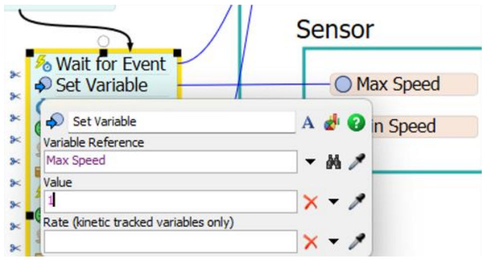
Alias	StartStop
Connection	Modbus TCP Connection
Register Type	Coil, 16-bit Int
Register #	0
Initial Value	0
Associated Object	0x0
Write Events	

Sensor
Max Speed — Variable or Value: Emulation Variable: Max Speed
Min Speed — Variable or Value: Emulation Variable: Min Speed
Start and stop — Variable or Value: Emulation Variable: StartStop

Figure 29-Modbus Sensor Register & Sensor Variable

Figure 30 illustrates the speed control logic configuration, where sensor variables such as Start and Stop, Max Speed, and Min Speed are set using "Set Variable" blocks. These variables are connected to sensor logic, enabling dynamic adjustment of the robot's speed based on defined conditions.





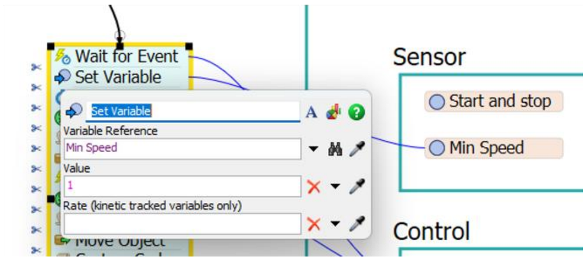


Figure 30-Speed Control Logic Configuration

5.3.4 Logic within Process Flow in FlexSim

The following Figure illustrates the whole process flow that monitors the system performance through collecting real-time data through control variables and controlling the operational behavior of the system through sensor variables.

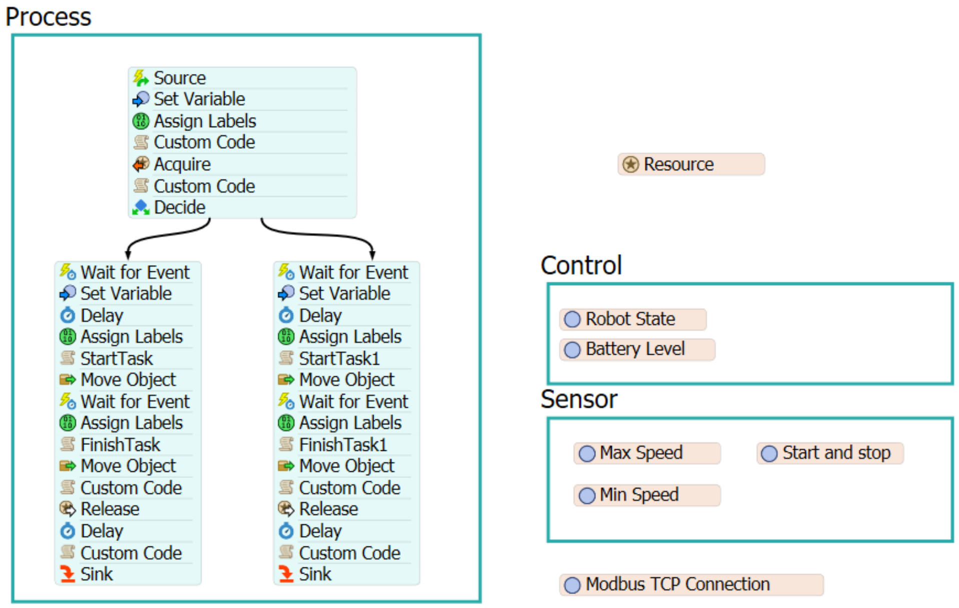
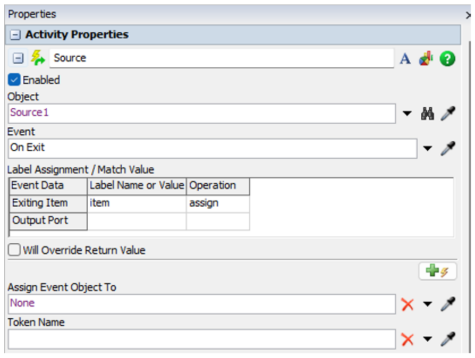


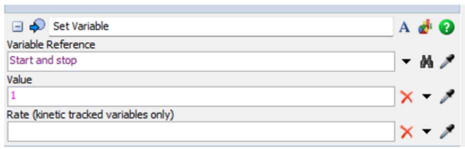
Figure 31- Digital Twin Process Flow

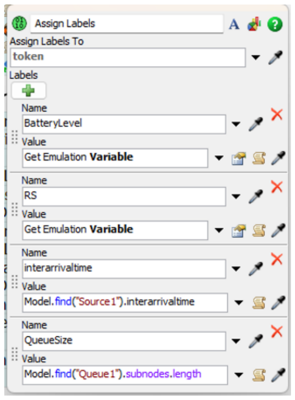
The process starts with the **Event-Triggered Sources** block. This block is connected on Source 3D model on exit, so whenever an item in 3D model created and exits from the source object in 3D model, an event-based trigger activates the source block in process flow. As a result, a token will be appeared in the process flow. The token passes through the **Set Variable** block to assign value 1 to *start and stop sensor variable* acts as a trigger, sending a signal to MiR100 to begin its task. Then the token enters the **Assign Labels** block to capture real-time data: the robot's battery level and robot state via get emulation variable (linked to the control variables Battery Level and Robot State). These labels indicate the current battery level of MiR100 and whether it is still processing a task or has finished and it is waiting for the next mission. Additionally, two other labels are assigned to the token, the interarrival time of the item, and the current queue size. These labels retrieve the current inter-arrival time from Source1 and the current number of items in Queue1 within the 3D model. The following figure shows the properties of the blocks discussed above.


Properties
Activity Properties
Source
Enabled: ✓
Object: Source1
Event: On Exit
Label Assignment / Match Value

Event Data	Label Name or Value	Operation
Exiting Item	item	assign
Output Port		

Will Override Return Value: ☐
Assign Event Object To: None
Token Name:

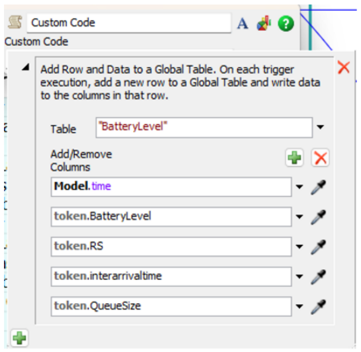

Set Variable
Variable Reference: Start and stop
Value: 1
Rate (kinetic tracked variables only):


Assign Labels
Assign Labels To: token
Labels
Name: BatteryLevel
Value: Get Emulation **Variable**
Name: RS
Value: Get Emulation **Variable**
Name: interarrivaltime
Value: Model.find("Source1").interarrivaltime
Name: QueueSize
Value: Model.find("Queue1").subnodes.length

1 2 3

Figure 32-Source, Set Variable and Assign Labels Properties

All this information captured through these labels is passed to **Custom Code** block**,** which is stores in a global table named "BatteryLevel" for monitoring and analysis. Next, the **Acquire** block reserves the task executor (as a resource) to ensure only one task is done at a time. the **Custom Code** block sets the robot's status to "busy", which is later used in the dashboard to calculate the robot's utilization. The token moves to the **Decide** block that evaluates the number of items already in the floor storage using the subnode.lenght function which counts the number of items in the 3D model. If there are 5 or more than 5 items in floor storage, indicating that there is no more space there, the flow shifts to the path 1 which corresponds to low-speed path. This speed adjustment helps to allow time for items in storage to be processed and prevents the accumulation of items in the queue area. These configurations are demonstrated in Figure 33.


Custom Code

Custom Code

Add Row and Data to a Global Table. On each trigger execution, add a new row to a Global Table and write data to the columns in that row.

Table "BatteryLevel"

Add/Remove Columns

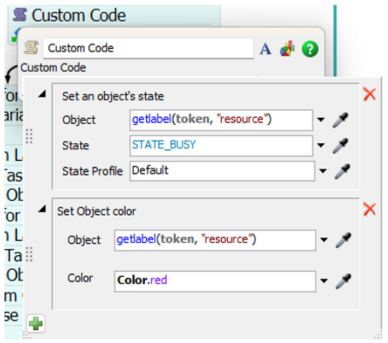
Model.time

token.BatteryLevel

token.RS

token.interarrivaltime

token.QueueSize


Custom Code

Custom Code

Custom Code

Set an object's state

Object getlabel(token, "resource")

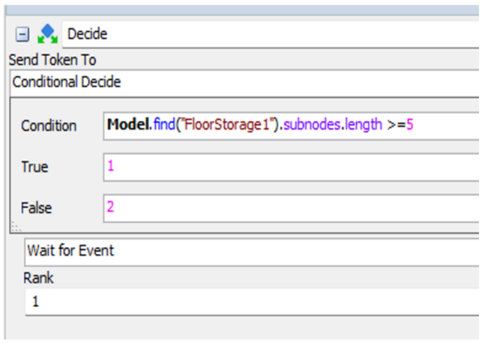
State STATE_BUSY

State Profile Default

Set Object color

Object getlabel(token, "resource")

Color Color.red


Decide

Send Token To

Conditional Decide

Condition Model.find("FloorStorage1").subnodes.length >=5

True 1

False 2

Wait for Event

Rank

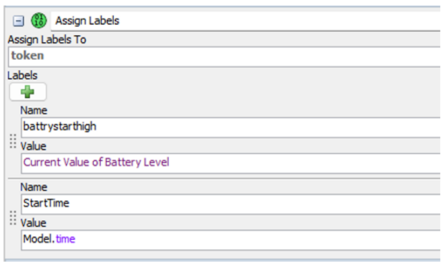
1

4 5 6

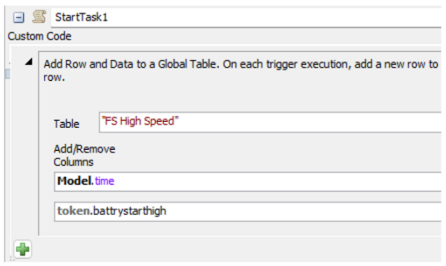
Figure 33-Custom Code and Decide Properties

The flow then continues along two parallel branches, both are following the same structure, the only difference between them is the applied speed, which is determined based on the number of items in the floor storage. The **Wait for Event** block is connected to the control variable named Robot Satet and is configured to trigger on change. This allows FlexSim to monitor and detect the robot's state changes. If the robot's state changes from 0 (Unloading point) to 1 (Loading point), the condition is met and the token continues along the process flow. Until this condition is met the token will be kept here. In this study, the event being monitored is the signal received from the physical system MiR100 robot, which updates the robot state variable to 0 once it has returned to the loading point and ready to transport a new item. This synchronization ensures that the virtual model waits for the robot's real-time availability before assigning the next mission. Once this condition is true, the system proceeds by **Set Variable** block to assign value 1 to a sensor variable (either *High Speed* or *Low Speed*) to determine the MiR100 speed. Following this, the 5-second **delay** is introduced to simulate the observed latency between sending the speed signal and the robot beginning its movement.

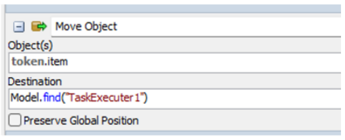
Next the **Assign Labels** block attach important real-time data to the token: current battery level (under the label batterystarthigh) and start time of task executor (under the label StartTime). These labels then are used in the Custom code block named **StartTask1**, which records these in the global table named FS High Speed. To synchronize the process flow and 3D model together, the **Move Object** block is used, which moves items from the queue to task executor. Figure 34 illustrates this step in the process.


Assign Labels
Assign Labels To: token
Labels
Name: battrystarthigh
Value: Current Value of Battery Level
Name: StartTime
Value: Model.time

7


StartTask1
Custom Code
Add Row and Data to a Global Table. On each trigger execution, add a new row to row.
Table: "FS High Speed"
Add/Remove Columns
Model.time
token.battrystarthigh

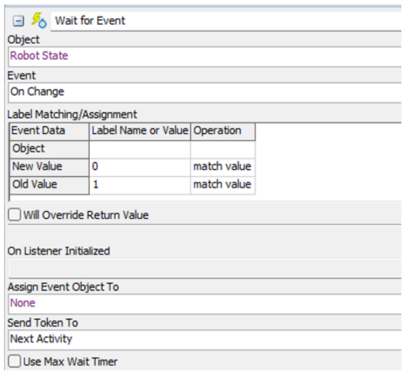
8


Move Object
Object(s): token.item
Destination: Model.find("TaskExecuter1")
Preserve Global Position

9

Figure 34-Assign Labels, Custom Code and Move Object Propertiess

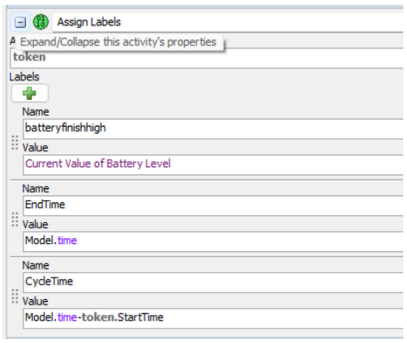
The **Wait for Event** block is connected to the control variable named Robot State to monitor the MiR100 state change: when the Robot State changes from 1 (Loading point) to 0 (Unloading point). The event is triggered only when this exact changes occur, allowing the flow to continue. If this condition is not met, then the token remains here until the robot finishes its task and becomes available in the loading point. Therefore, this block ensures proper synchronization between the physical robot and the process flow. Next, the **Assign Labels** attach important real-time data to the token: current battery level (under the label batteryfinishhigh), finish time of task executor (under the label EndTime) and total duration of process which is calculated as cycle time (under the label CycleTime). Then these labels are inserted into the same global table where the StartTime was previously recorded, using a custom code block named **FinishTask1**. The following Figure demonstrates how these functions are implemented within the process flow.


Wait for Event
Object: Robot State
Event: On Change
Label Matching/Assignment

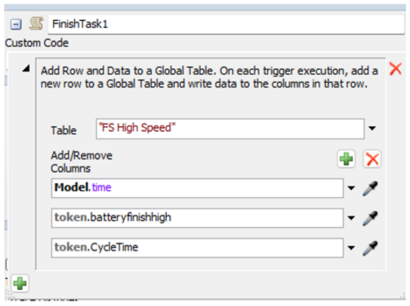
Event Data	Label Name or Value	Operation
Object		
New Value	0	match value
Old Value	1	match value

Will Override Return Value
On Listener Initialized
Assign Event Object To: None
Send Token To: Next Activity
Use Max Wait Timer

10


Assign Labels
Expand/Collapse this activity's properties
token
Labels
Name: batteryfinishhigh
Value: Current Value of Battery Level
Name: EndTime
Value: Model.time
Name: CycleTime
Value: Model.time-token.StartTime

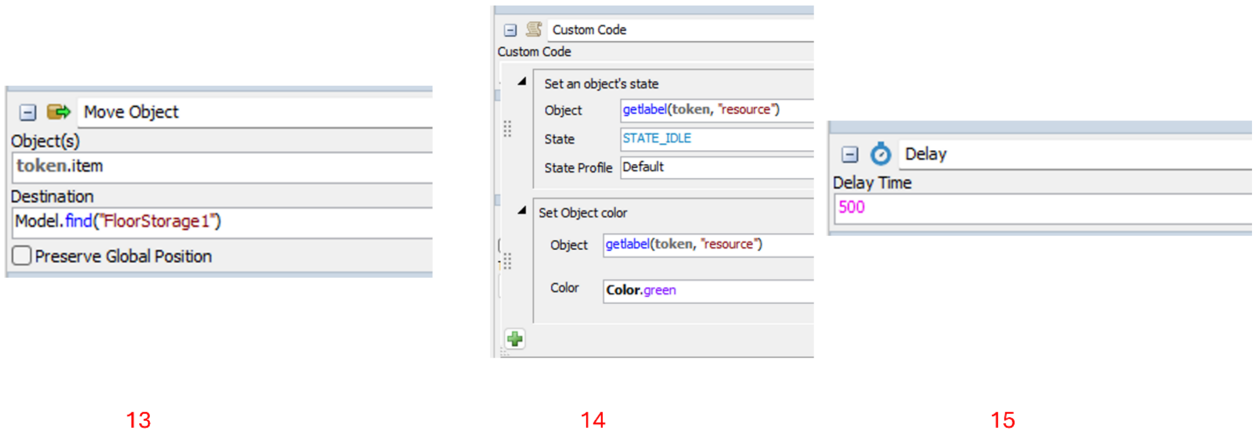
11


FinishTask1
Custom Code
Add Row and Data to a Global Table. On each trigger execution, add a new row to a Global Table and write data to the columns in that row.
Table: "FS High Speed"
Add/Remove Columns
Model.time
token.batteryfinishhigh
token.CycleTime

12

Figure 35-Wait for Event, Assign Labels and Custom Code Properties

As can be seen in Figure 36, the **Move Object** moves the item from task executor to the floor storage, synchronizing the process flow and 3D model together. The **Custom Code** block assigns the robot's state back to "idle" and the **Release** block frees the robot for the next task. Finally, the item stays in floor storage for **500 seconds** before being removed from system via the **Sink** block.

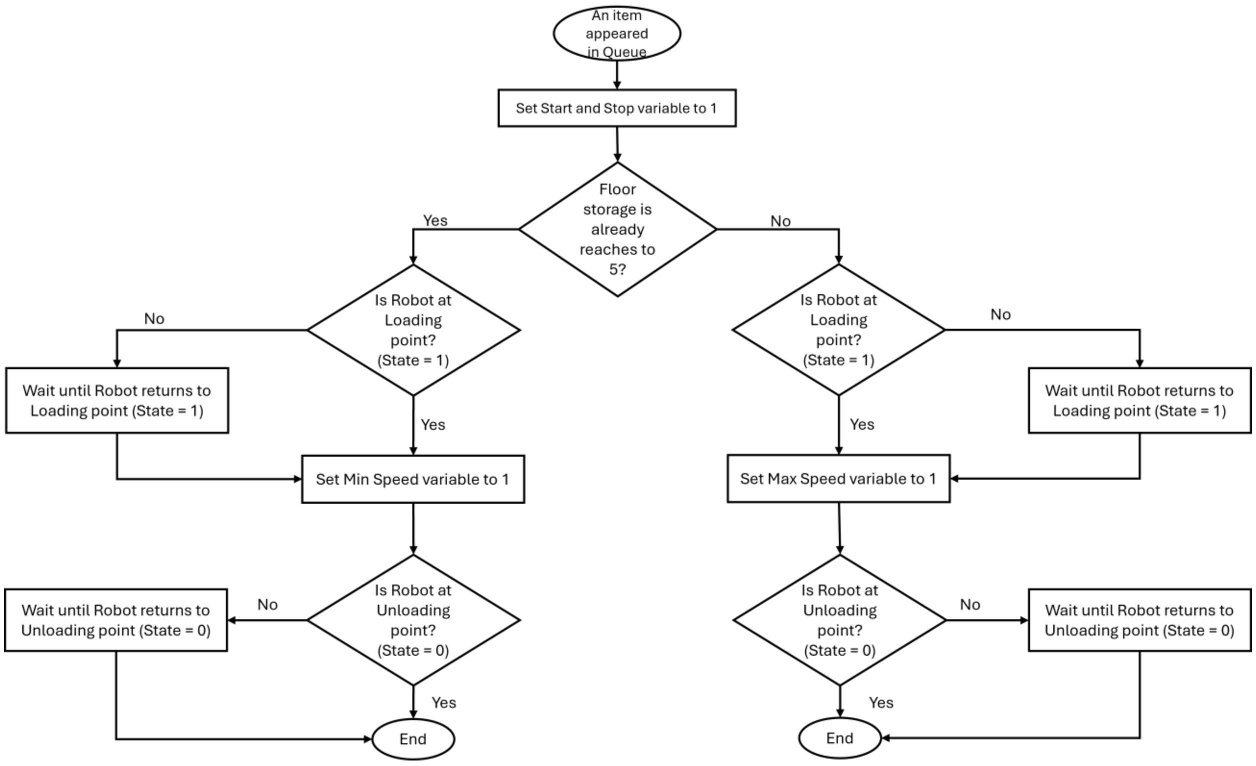

Move Object
Object(s): token.item
Destination: Model.find("FloorStorage1")
Preserve Global Position

Custom Code
Set an object's state
Object: getlabel(token, "resource")
State: STATE_IDLE
State Profile: Default
Set Object color
Object: getlabel(token, "resource")
Color: Color.green

Delay
Delay Time: 500

Figure 36-Move Object, Custom Code and Delay Properties

The following Figure provides simplified process flow that shows how sensor variables wroks between the FlexSim simulation and the real system.



An item appeared in Queue
Set Start and Stop variable to 1
Floor storage is already reaches to 5?
Yes
No
Is Robot at Loading point? (State = 1)
No
Wait until Robot returns to Loading point (State = 1)
Yes
Set Min Speed variable to 1
Is Robot at Unloading point? (State = 0)
No
Wait until Robot returns to Unloading point (State = 0)
Yes
End
Is Robot at Loading point? (State = 1)
No
Wait until Robot returns to Loading point (State = 1)
Yes
Set Max Speed variable to 1
Is Robot at Unloading point? (State = 0)
No
Wait until Robot returns to Unloading point (State = 0)
Yes
End

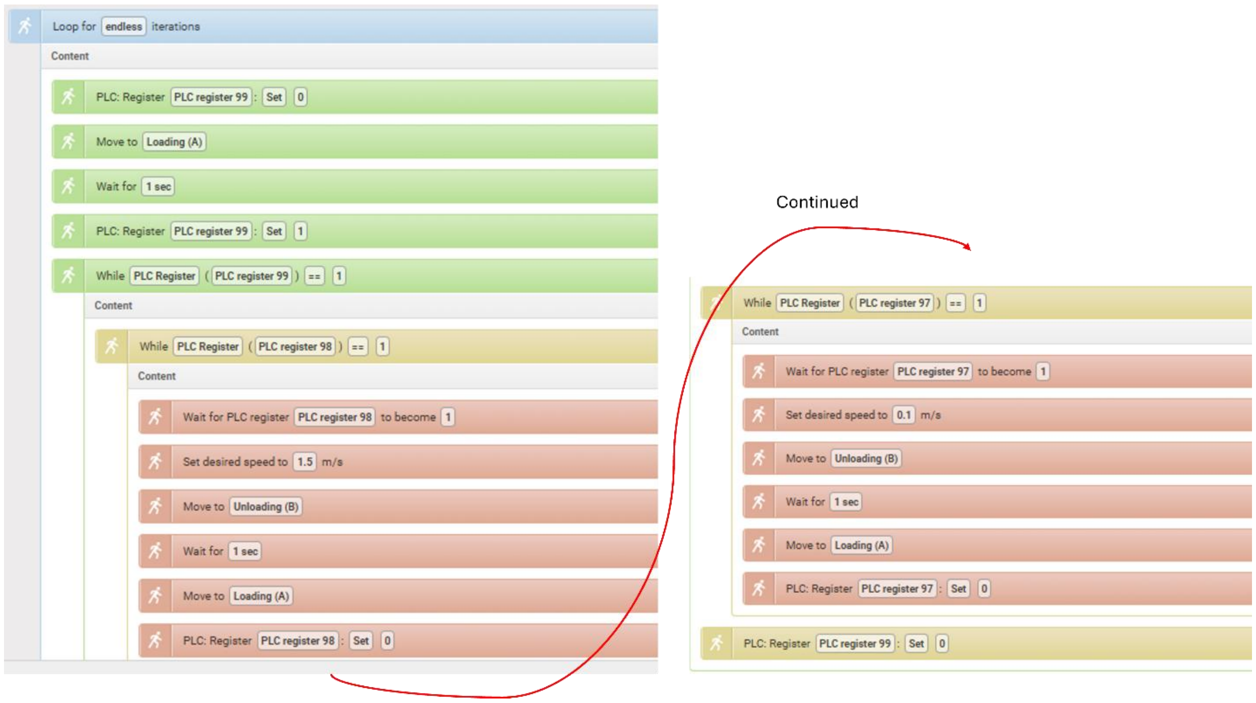
Figure 37-Process Flow of Communication via Sensor Variables Between FlexSim and MiR100

5.3.5 Logic within MiR100

The MiR100’s operation is programmed within the portal. In the Mission tab, users can create a new mission through dragging a series of actions. Figure 38 shows the logic configuration for controlling the MiR100's motion through the actions. In the following there is an explanation of the logic in MiR100, which ensuring real-time synchronization between the MiR100 and FlexSim. The mission in MiR100 is running in a *loop*, which means it repeats the mission for unlimited iterations. First, the robot sets *PLC register 99* to 0 which indicates the MiR100 is available. The MiR100 *moves to Loading Point (A)* and *waits for 1 second*, which simulates loading time. Then, *PLC Register 99* is set to 1, which indicates to the FlexSim that the robot has a Mission (now is busy).

From this point, *PLC register 97* (for low speed) and *PLC register 98* (for high speed) are constantly monitored through the *While* action. Only one of these *While* action will become activate (become 1), which depends on the signal received from the FlexSim. If *Register 98* is 1, then the MiR100 will go with high speed. It *waits for PLC Register 98* to become 1, then the *speed sets to 1.5 m/s*, *moves to Unloading point (B)*, *waits for 1 second* and then returns to the *Loading Point (A)*. sets *PLC Register 98* to 0 to reset the high-speed path.

If *Register 97* is 1, then the MiR100 will go with low speed. It waits for *PLC Register 97* to become 1, then the *speed sets to 0.1* m/s, *moves to Unloading point (B)*, *waits for 1 second* and then returns to the *Loading Point (A)* and set low speed (*PLC Register 97)* back to 0 to reset it. Finally, to indicate the robot is ready for the next task, *PLC Register 99* is set to 0.



```
Loop for endless iterations
Content
  PLC: Register PLC register 99 : Set 0
  Move to Loading (A)
  Wait for 1 sec
  PLC: Register PLC register 99 : Set 1
  While PLC Register ( PLC register 99 ) == 1
  Content
    While PLC Register ( PLC register 98 ) == 1
    Content
      Wait for PLC register PLC register 98 to become 1
      Set desired speed to 1.5 m/s
      Move to Unloading (B)
      Wait for 1 sec
      Move to Loading (A)
      PLC: Register PLC register 98 : Set 0
```

```
    While PLC Register ( PLC register 97 ) == 1
    Content
      Wait for PLC register PLC register 97 to become 1
      Set desired speed to 0.1 m/s
      Move to Unloading (B)
      Wait for 1 sec
      Move to Loading (A)
      PLC: Register PLC register 97 : Set 0
  PLC: Register PLC register 99 : Set 0
```

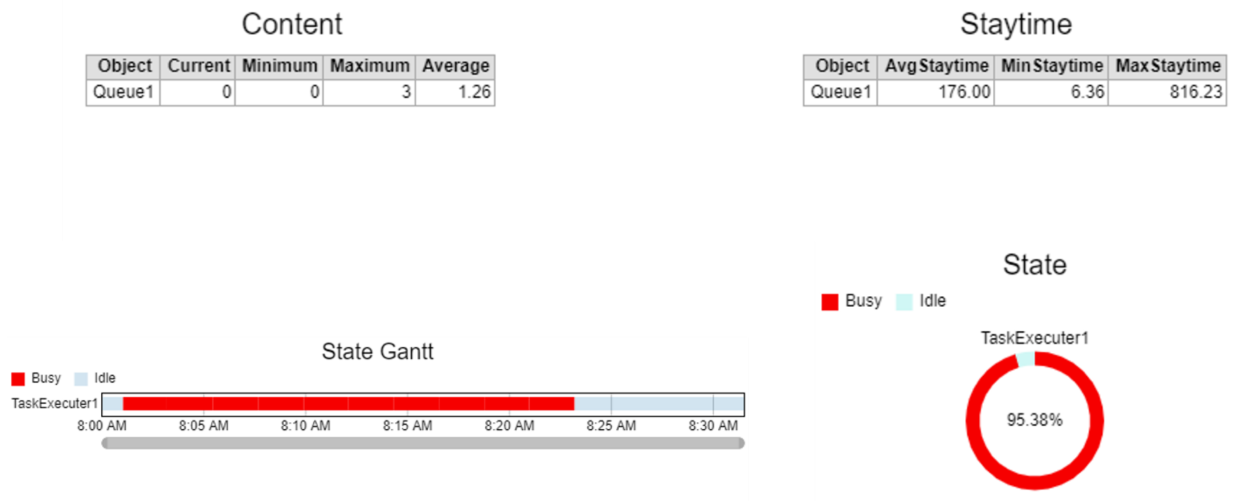
Figure 38-MiR100’s Actions

5.4 Scenario Description

Once the simulation starts, items are created based on a uniform inter-arrival time. As soon as an item enters Queue1, the MiR100 takes the item from the loading point (A), waits for 1 second, then goes to the unloading point (B), where there is the storage. Here the robot stays 1 second to unload the item. After unloading it returns to the loading point (A) to repeat this process for the rest of the 10 items. All the assumptions that are mentioned in section 5.3.1 Assumptions and characteristics are applied for three scenarios.

5.4.1 First Scenario: Transporting at Low Speed (0.1 m/s)

In this scenario, MiR100 performs all transportation processes at the lowest speed (0.1 m/s). If the warehouse storage reaches full capacity, the robot must wait until at least one item goes out of the system. The transportation cycle starts from loading point (A) to reach unloading point (B) and then returns to loading point (A), with an average cycle time of 118 seconds. Items, on average, spent 176 seconds waiting in the queue before being transported with an average queue length of 1.26 items. The utilization in this scenario is 95.38%, indicating that the robot is working most of the time. The Figure below shows the operational system performance generated by FlexSim software.



Content

Object	Current	Minimum	Maximum	Average
Queue1	0	0	3	1.26

Staytime

Object	Avg Staytime	Min Staytime	Max Staytime
Queue1	176.00	6.36	816.23

Figure 39-First Scenario Operational Performance Indicator

Figure 40 shows power consumption changes overtime for transporting each item at low speed. There are some fluctuations during the transportation of each item, which is normal as the current and voltage depends on movement dynamics and pauses during the mission.

Power consumption is calculated by multiplying the current by voltage. In the Figure below, the red dashed line indicates the mean power consumption which is around 7.95 W. However, power consumption alone is not a good KPI. Therefore, energy consumption is also defined, calculated by multiplying the average power consumption by the duration of the simulation which is 1374 seconds.

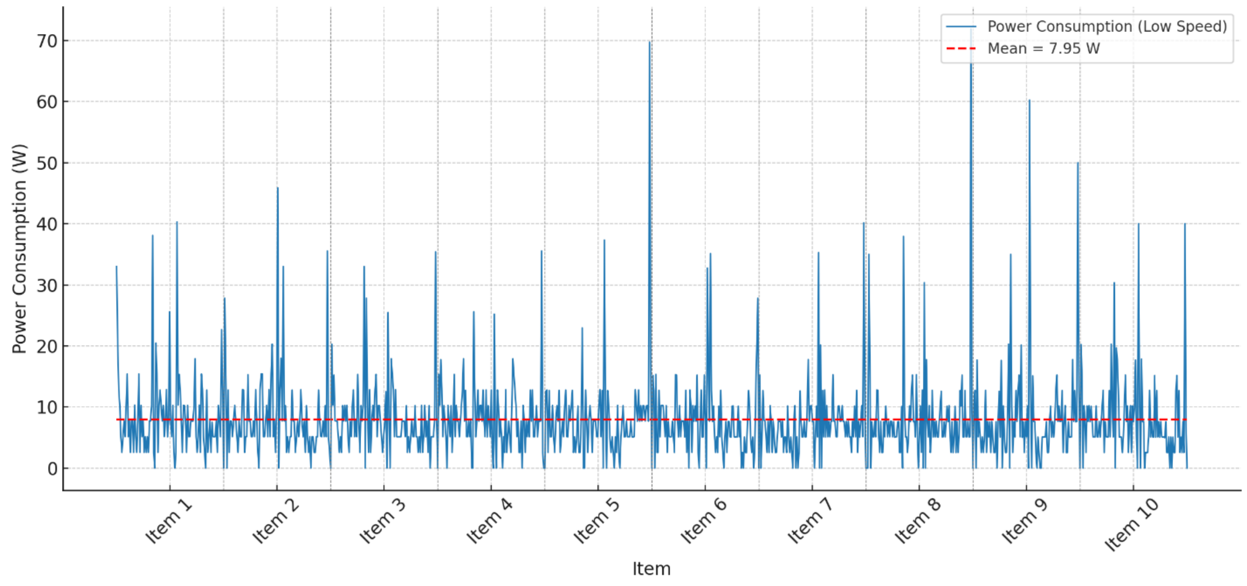
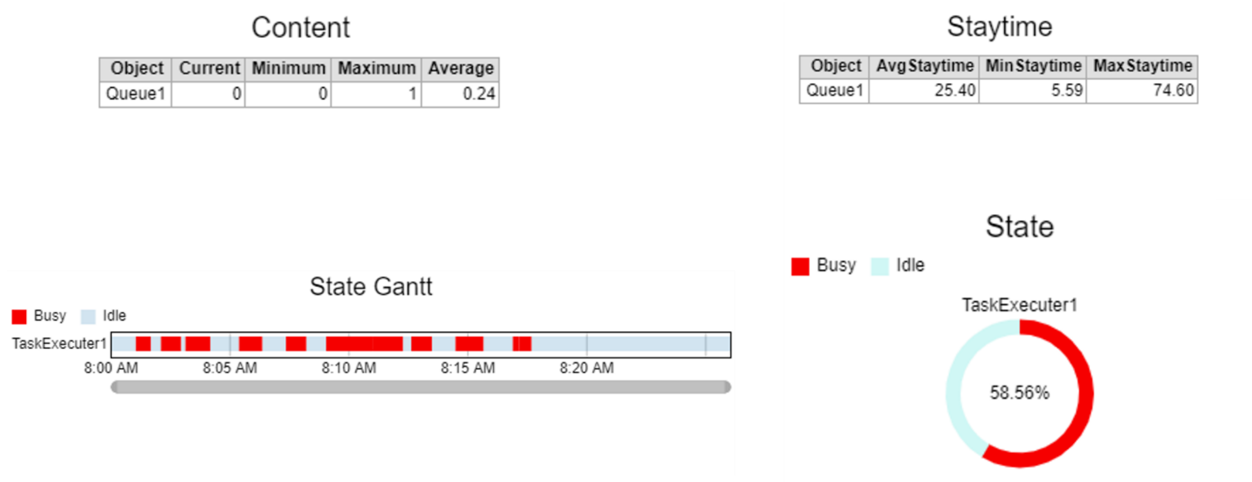


Figure 40-First Scenario Power Consumption Indicator

5.4.2 Second Scenario: Transporting at High Speed (1.5 m/s)

In this scenario, the robot performs all transport tasks at its maximum speed of 1.5 m/s, enabling much faster cycle times in comparison to slower speed. The average cycle time takes only 42 seconds, moving from the loading point (A) to the unloading point (B) and returning. In this scenario, items experience significantly shorter waiting times in the queue, with an average of 25.40 seconds and a queue length of 0.24 items. However, despite transporting items fast, the robot's utilization drops to 58.56%, which means that the robot spends more time idle and often waiting for the next item to be available. Therefore, while robot's speed increases alone, does not improve the performance when downstream capacity (floor storage) is restricted. The following Figure shows the results of this scenario in terms of operational performance.


Content

Object	Current	Minimum	Maximum	Average
Queue1	0	0	1	0.24

Staytime

Object	Avg Staytime	Min Staytime	Max Staytime
Queue1	25.40	5.59	74.60

Figure 41-Second Scenario Operational Performance Indicator

In this scenario, the robot transports each item faster, but this comes with a noticeable trade-off in energy behavior. As shown in Figure 42, there are some sharp changes due to the robot's fast acceleration, deceleration and short pauses. The average power consumption during the scenario reaches 34.46 W, which is significantly higher than the low-speed scenario. This increase reflects the pressure on the battery system while robot is moving at high speed. As a result, the battery drains much faster, as it showed in section *5.2.3 Discussion*. Like the first scenario, energy consumption is calculated by multiplying the average power consumption by the toral duration of the simulation which is 1074. Although high speed reduces task completion time, it leads to much higher energy consumption, which is a concern in sustainability approach.

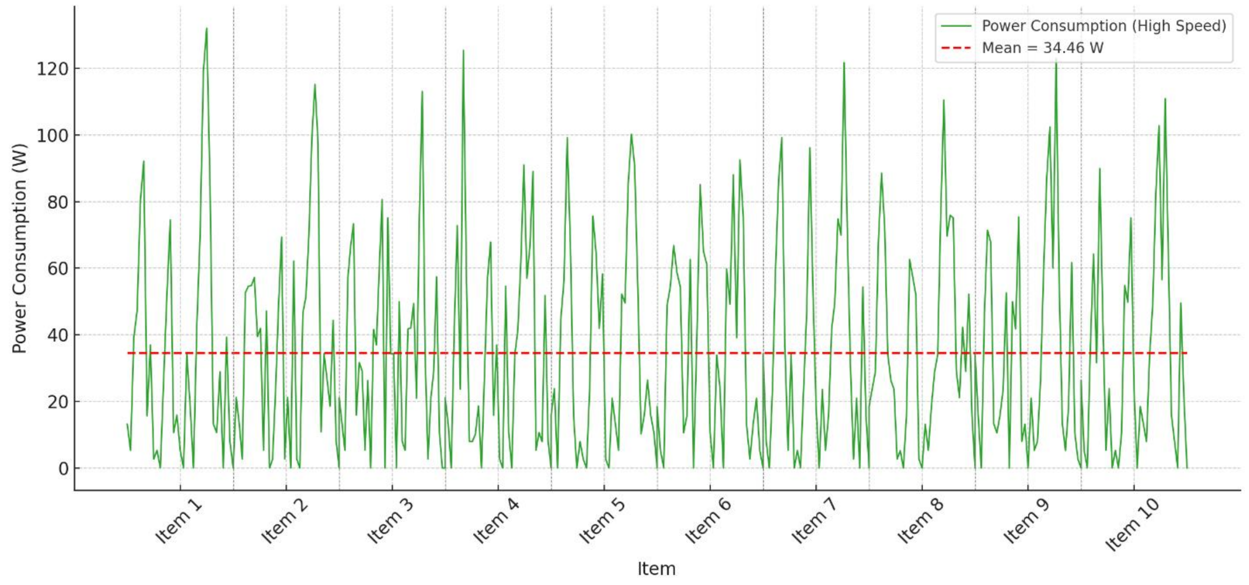
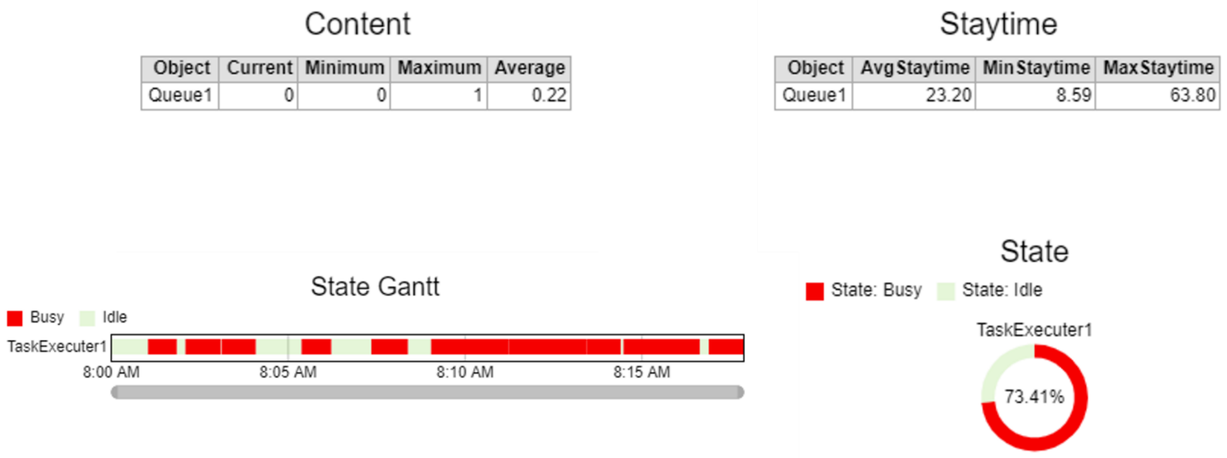


Figure 42-Second Scenario Power Consumption Indicator

5.4.3 Third Scenario: Transporting with Dynamic Speed (1.5 m/s or 0.1 m/s)

In this proposed solution, the robot performs transport at high speed (1.5 m/s), but then when the floor storage reaches the maximum capacity (5 items), the robot will not stopped its work, it starts to transport the item at lowest speed (0.1 m/s).
In this scenario, items experience significantly shorter waiting times in the queue, with an average of 23.20 seconds and a queue length of 0.22 items. However, despite transporting items fast, the robot's utilization increases to 73.41%, which means that the robot does not spend more time idle. Figure 43 provides an overview of the results related to operational performance indicators.



Content

Object	Current	Minimum	Maximum	Average
Queue1	0	0	1	0.22

Staytime

Object	AvgStaytime	MinStaytime	MaxStaytime
Queue1	23.20	8.59	63.80

Figure 43-Third Scenario Operational Performance Indicator

As is shown in the Figure below, item numbers 6, 7 and 9 are transported at a low speed because at those moments the floor storage was already full. To avoid the robot being idle, it reduces its speed. The red boxes clearly highlight these intervals, during which power consumption decreases significantly compared to other high-speed intervals. Power consumption during the transport of items 6, 7 and 9 is mostly below the average, which is 19.57 W (Red dashed lines). Therefore, it supports the idea that transporting at low speed results in lower power consumption, reflecting energy saving behavior. In contrast, for items 1 to 5 and 8 and 10 which are corresponding to high speed transportation in some points reaching over 120 W. This pattern confirms that the dynamic speed adjustment logic helps to optimize the power consumption based on real-time system constraints, showing the balance between operational indicators and sustainability.

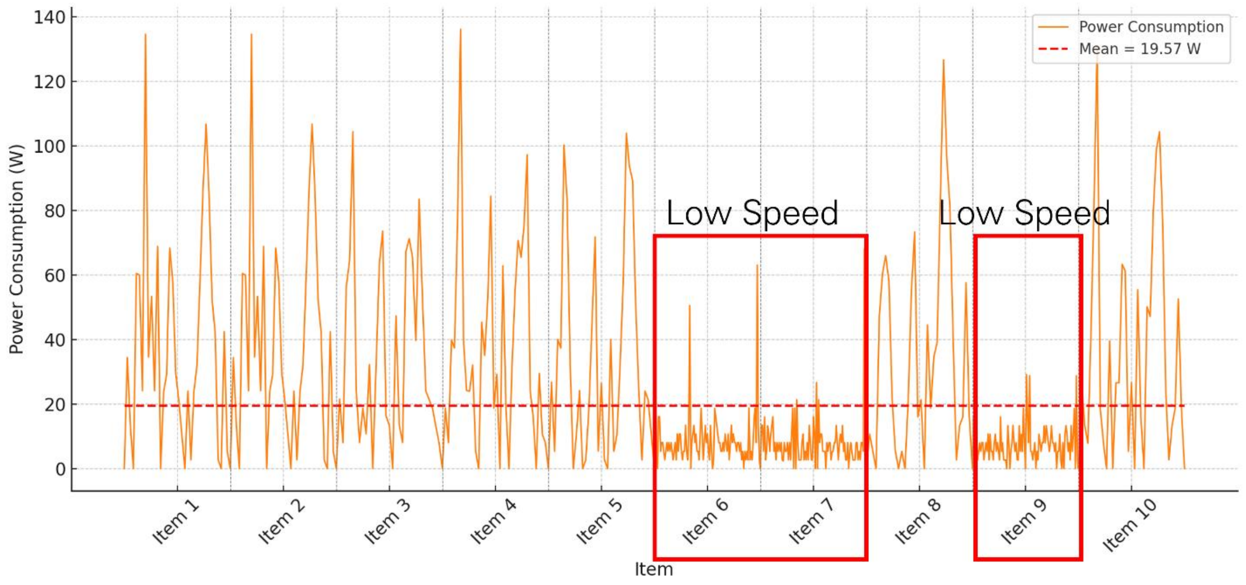


Figure 44- Third Scenario Power Consumption Indicator

5.4.4 Discussion

In this section, a comprehensive comparison among three scenarios is presented. The Figure 45 shows the transportation timelines of ten items under three scenarios: Scenario 1 (Low Speed, 0.1 m/s), Scenario 2 (High Speed, 1.5 m/s), and Scenario 3 (Digital Twin, dynamic speed). The red vertical lines represent the arrival times of each item into the system. While the two Tables below represent the operational and sustainability indicators' results.
In scenario 1, from operational perspective, each item is transported at low speed, resulting in the longest average time (118.3 seconds) per item and also longest cycle time of 1374 seconds for all 10 items. The utilization rate in this scenario also is the highest at 95.38%, which indicates that the robot is rarely idle. In this scenario, there is no problem in terms of reaching full capacity at the floor storage as the robot moves at lowest speed, the transported items will have time to leave the system. However, transporting at the low-speed results in long queues, with an average of 1.26 items in the queue and an average queue stay time of 176 seconds. In terms of sustainability, this scenario achieves the lowest average power consumption (7.95 W) and lowest energy consumption (3.03 Wh), which is calculated through dividing the power consumption by the cycle time.
In contrast, in scenario 2, in terms of operational, it shows significantly shorter transportation time for each item and shorter cycle time for transporting all 10 items. The robot's utilization rate drops to 58.56%, which means that the robot has substantial idle time. This is confirmed by the Gantt chart, where after item 5, the robot waits because the floor storage is full. The average number of items in the queue is only 0.24 and stay time in queue is reduced to 25.4 seconds, showing that while the robot transports items quickly, the limited capacity of the floor storage causes delay, which makes robot to wait and become idle. This can be considered as a bottleneck at the unloading point. In terms of sustainability, scenario 2 consumes the most power (34.45 W average power) and energy (10.27 Wh), which makes it the least sustainable one.

Scenario 3 (Digital Twin), using dynamic speed adjustment, shows a balanced approach. It dynamically changes to low speed when the floor storage is full. As it can be seen in the Figure below for items 6, 7 and 9. This approach gives time to the floor storage to clear the items from it. The average utilization is 73.41%, which is higher than the high-speed scenario, but lower the low-speed scenario. The queue content and stay time are the lowest (0.22 items, 23.2 seconds), showing excellent flow management. Although the average transport time varies, the total cycle time is 1072 seconds, nearly matching the high-speed scenario, but with much better balance. In terms of sustainability, scenario 3 again makes a balance with 19.56 W average power and 5.82 Wh total energy consumption, which is substantially better than high speed, while maintaining strong operational results.

In conclusion, Scenario 3 (Digital Twin) is the most efficient approach in this case study. It dynamically adapts to system limitations, avoiding unnecessary idling and battery waste while maintaining nearly optimal cycle time and queue management. It successfully balances operational KPIs and sustainability goals within the specific conditions of this thesis.

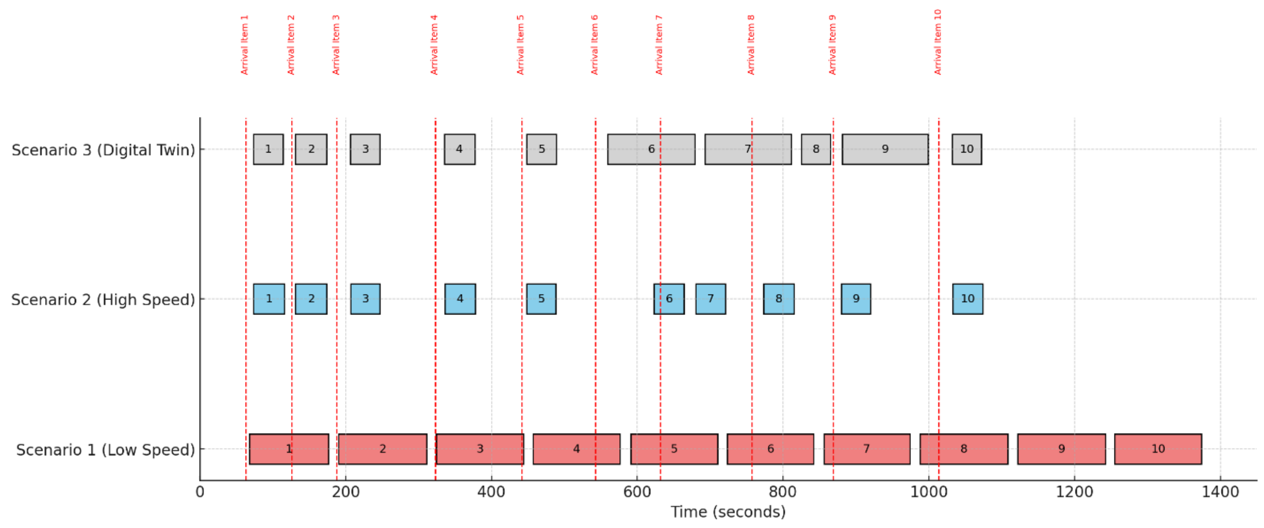


Figure 45-Item Transportation Timeline in Low-Speed, High-Speed, and Digital Twin Scenarios

Operation Performance Indicators	Low Speed	High Speed	DT
Average Time per each item	118.3	42.2	Dynamically changed
Utilization Rate	95.38%	58.56%	73.41%
Cycle Time (Seconds)	1374	1074	1072
Average Items in the Queue	1.26	0.24	0.22
Stay Time in the Queue (Seconds)	176	25.40	23.20

Table 4-Operational Performance Results Comparison

Sustainability Performance Indicators	Low Speed	High Speed	DT
Average power consumption (W)	7.95	34.45	19.56
Energy Consumption (Wh)	3.03	10.27	5.82

Table 5-Sustainability Performance Result Comparison

6. Conclusion and Future Works

This thesis explored the application of Digital Twin technology in order to improve internal logistics operations by synchronizing a virtual model in FlexSim with a physical system specifically MiR100 Autonomous Mobile Robot. The main objective of the study was to develop and evaluate speed adjustment by considering both operational efficiency and sustainability metrics, particularly where there is constrained storage. By integrating real-time data from MiR100 into the model built in FlexSim via Modbus TCP, the system was able to monitor and dynamically respond to bottlenecks.

Three different scenarios were analyzed: constant low speed, constant high speed and dynamic speed adjustment representing Digital Twin. The results demonstrated that while robot transport items at a high speed, transportation cycle time is reduced, which leads to idle periods due to warehouse capacity limitation which is creating a bottleneck at the unloading point, resulting in greater both power and energy consumption. Conversely, operating at low speed preserved battery and reduced energy consumption, but it resulted in longer cycle times and increased queue buildup.

The proposed Digital Twin scenario achieved a balanced solution. By adapting the robot’s speed based on real-time system status, such as storage availability, this method successfully managed to reduce the idle time while optimizing both operational KPIs and sustainability indicators. Power consumption analysis was based on real-time data including current and voltage reading obtained from the MiR portal. By calculating average power and total cycle time, energy consumption is assessed for three scenarios. The results showed that the developed model confirms the potential of Digital Twin framework in supporting smarted control logic and data-driven decisions in internal logistics.

However, this study also has some limitations. In this experiment, the system was limited to ten items and assumed uniform interarrival times. While this setup effectively demonstrated how different speed strategies impact energy and operational performance, the limited number of missions may not fully capture the complexities of a real logistics environment. In larger scale operations, the benefits of dynamic speed adjustment may become even more significant. Moreover, the pick and place task was performed by an operator at a fixed pace across all tasks.

In reality, an operator can have different working speeds, then introducing this variability could improve the realism of the model

For future work, the system could be extended to include multiple robots transporting collaboratively or explore the integration of a manipulator mounted on the MiR100 for pick and place operation for full automation exploring more dynamic arrival patterns to evaluate long term performance. Moreover, logic could be improved by using predictive algorithms or machine learning models that learn from historical patterns to optimize speed adjustments and energy usage.

In conclusion, this research confirms that Digital Twin implementation, when properly integrated with real-time data, can improve system performance and energy efficiency in internal logistics. As internal logistics systems continue to become more complex, Digital Twin will become an increasingly important tool in developing and implementing flexible and responsive solutions.

Bibliography

Aheleroff, S., Huang, H., Xu, X., & Zhong, R. Y. (2022). Toward sustainability and resilience with Industry 4.0 and Industry 5.0. *Frontiers in Manufacturing Technology*, *2*. https://doi.org/10.3389/fmtec.2022.951643

Aliev, K., Traini, E., Asranov, M., Awouda, A., & Chiabert, P. (2021). Prediction and estimation model of energy demand of the AMR with cobot for the designed path in automated logistics systems. *Procedia CIRP*, *99*, 116–121. https://doi.org/10.1016/j.procir.2021.03.036

Alnowaiser, K. K., & Ahmed, M. A. (2023). Digital Twin: Current Research Trends and Future Directions. *Arabian Journal for Science and Engineering*, *48*(2), 1075–1095. https://doi.org/10.1007/s13369-022-07459-0

Antonelli, D., & Aliev, K. (2022). Intelligent Energy Management for Mobile Manipulators Using Machine Learning. *FME Transactions*, *50*(4), 752–761. https://doi.org/10.5937/fme2204752A

Antonelli, D., Aliev, K., Soriano, M., Samir, K., Monetti, F. M., & Maffei, A. (2024). Exploring the limitations and potential of digital twins for mobile manipulators in industry. *Procedia Computer Science*, *232*, 1121–1130. https://doi.org/10.1016/j.procs.2024.01.110

Bamigbala, T., Onkamo, M., Safonova, I., & Rahman, T. (n.d.). *Towards Adoption of Autonomous Mobile Cobots in Intralogistics Picking Process: Review of Current Development 4 PUBLICATIONS 4 CITATIONS SEE PROFILE*. https://doi.org/10.13140/RG.2.2.13001.01127

Barata, J., & Kayser, I. (2024). How will the digital twin shape the future of industry 5.0? In *Technovation* (Vol. 134). Elsevier Ltd. https://doi.org/10.1016/j.technovation.2024.103025

Dobrzańska, M., & Dobrzański, P. (2025). Simulation Model as an Element of Sustainable Autonomous Mobile Robot Fleet Management. *Energies*, *18*(8). https://doi.org/10.3390/en18081894

Fera, M., Greco, A., Caterino, M., Gerbino, S., Caputo, F., Macchiaroli, R., & D'amato, E. (2020). Towards digital twin implementation for assessing production line performance and balancing. *Sensors (Switzerland)*, *20*(1). https://doi.org/10.3390/s20010097

Ferko, E., Bucaioni, A., & Behnam, M. (2022). Architecting Digital Twins. *IEEE Access*, *10*, 50335–50350. https://doi.org/10.1109/ACCESS.2022.3172964

Golovianko, M., Terziyan, V., Branytskyi, V., & Malyk, D. (2022). Industry 4.0 vs. Industry 5.0: Co-existence, Transition, or a Hybrid. *Procedia Computer Science*, *217*, 102–113. https://doi.org/10.1016/j.procs.2022.12.206

He, L., Chiong, R., & Li, W. (2022). Energy-efficient open-shop scheduling with multiple automated guided vehicles and deteriorating jobs. *Journal of Industrial Information Integration*, *30*. https://doi.org/10.1016/j.jii.2022.100387

He, Y., Li, Y., Wu, T., & Sutherland, J. W. (2015). An energy-responsive optimization method for machine tool selection and operation sequence in flexible machining job shops. *Journal of Cleaner Production*, *87*(C), 245–254. https://doi.org/10.1016/j.jclepro.2014.10.006

Hermann, M., Pentek, T., & Otto, B. (2016). Design principles for industrie 4.0 scenarios. *Proceedings of the Annual Hawaii International Conference on System Sciences*, *2016-March*, 3928–3937. https://doi.org/10.1109/HICSS.2016.488

Homayouni, S. M., & Fontes, D. B. M. M. (2021). A MILP Model for Energy-Efficient Job Shop Scheduling Problem and Transport Resources. *IFIP Advances in Information and Communication Technology*, *630 IFIP*, 378–386. https://doi.org/10.1007/978-3-030-85874-2_40

Javaid, M., Haleem, A., & Suman, R. (2023). Digital Twin applications toward Industry 4.0: A Review. In *Cognitive Robotics* (Vol. 3, pp. 71–92). KeAi Communications Co. https://doi.org/10.1016/j.cogr.2023.04.003

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, *29*, 36–52. https://doi.org/10.1016/j.cirpj.2020.02.002

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Li, L., Lei, B., & Mao, C. (2022). Digital twin in smart manufacturing. *Journal of Industrial Information Integration*, *26*. https://doi.org/10.1016/j.jii.2021.100289

Liau, Y., Lee, H., & Ryu, K. (2018). Digital Twin concept for smart injection molding. *IOP Conference Series: Materials Science and Engineering*, *324*(1). https://doi.org/10.1088/1757-899X/324/1/012077

Liu, X., Jiang, D., Tao, B., Jiang, G., Sun, Y., Kong, J., Tong, X., Zhao, G., & Chen, B. (2022). Genetic Algorithm-Based Trajectory Optimization for Digital Twin Robots. *Frontiers in Bioengineering and Biotechnology*, *9*. https://doi.org/10.3389/fbioe.2021.793782

Mei, Y., Lu, Y. H., Hu, Y. C., & Lee, C. S. G. (2006). Deployment of mobile robots with energy and timing constraints. *IEEE Transactions on Robotics*, *22*(3), 507–522. https://doi.org/10.1109/TRO.2006.875494

Mei, Y., Lu, Y.-H., Charlie Hu, Y., & Georg, C. (n.d.). *Energy-Efficient Motion Planning for Mobile Robots*.

Ouahabi, N., Chebak, A., Kamach, O., Laayati, O., & Zegrari, M. (2024). Leveraging digital twin into dynamic production scheduling: A review. In *Robotics and Computer-Integrated Manufacturing* (Vol. 89). Elsevier Ltd. https://doi.org/10.1016/j.rcim.2024.102778

Pizoń, J., Wójcik, Ł., Gola, A., Kański, Ł., & Nielsen, I. (2024a). Autonomous Mobile Robots in Automotive Remanufacturing: A Case Study for Intra-Logistics Support. *Advances in Science and Technology Research Journal*, *18*(1), 213–230. https://doi.org/10.12913/22998624/177398

Pizoń, J., Wójcik, Ł., Gola, A., Kański, Ł., & Nielsen, I. (2024b). Autonomous Mobile Robots in Automotive Remanufacturing: A Case Study for Intra-Logistics Support. *Advances in Science and Technology Research Journal*, *18*(1), 213–230. https://doi.org/10.12913/22998624/177398

Proceedings of the 2021 Design, Automation & Test in Europe (DATE 2021) : 01-05 February 2021, virtual conference. (2021). EDAA.

Qi, Q., Tao, F., Zuo, Y., & Zhao, D. (2018). Digital Twin Service towards Smart Manufacturing. *Procedia CIRP*, *72*, 237–242. https://doi.org/10.1016/j.procir.2018.03.103

Ramasubramanian, A. K., Mathew, R., Kelly, M., Hargaden, V., & Papakostas, N. (2022). Digital Twin for Human-Robot Collaboration in Manufacturing: Review and Outlook. *Applied Sciences (Switzerland)*, *12*(10). https://doi.org/10.3390/app12104811

Rizqi, Z. U., Chou, S. Y., & Cahyo, W. N. (2024a). A simulation-based Digital Twin for smart warehouse: Towards standardization. *Decision Analytics Journal*, *12*. https://doi.org/10.1016/j.dajour.2024.100509

Rizqi, Z. U., Chou, S. Y., & Cahyo, W. N. (2024b). A simulation-based Digital Twin for smart warehouse: Towards standardization. *Decision Analytics Journal*, *12*. https://doi.org/10.1016/j.dajour.2024.100509

Rosen, R., Von Wichert, G., Lo, G., & Bettenhausen, K. D. (2015). About the importance of autonomy and digital twins for the future of manufacturing. *IFAC-PapersOnLine*, *28*(3), 567–572. https://doi.org/10.1016/j.ifacol.2015.06.141

Scott, R. (2020). *Untangling the requirements of a Digital Twin Foreword 2 Untangling the requirements of a Digital Twin*.

Semeraro, C., Lezoche, M., Panetto, H., & Dassisti, M. (2021). Digital twin paradigm: A systematic literature review. *Computers in Industry*, *130*, 103469. https://doi.org/10.1016/j.compind.2021.103469ï

Stączek, P., Pizoń, J., Danilczuk, W., & Gola, A. (2021). A digital twin approach for the improvement of an autonomous mobile robots (AMR's) operating environment—A case study. *Sensors*, *21*(23). https://doi.org/10.3390/s21237830

Steinmetz, C., Schroeder, G. N., Binotto, A., Panikkar, S., Papenfuß, B., Schmidt, C., Rettberg, A., & Pereira, C. E. (2022). Digital Twins modeling and simulation with Node-RED and Carla. *IFAC-PapersOnLine*, *55*(19), 97–102. https://doi.org/10.1016/j.ifacol.2022.09.190

VanDerHorn, E., & Mahadevan, S. (2021). Digital Twin: Generalization, characterization and implementation. *Decision Support Systems*, *145*. https://doi.org/10.1016/j.dss.2021.113524

Wu, M., Su Lee Ming, E., Zheng, Y., Dong, B., Fai Yeong, C., & Holderbaum, W. (n.d.). *Green Automation: Empowering a Sustainable Future with Energy-Efficient Autonomous Mobile Robots in Manufacturing*.

Yin, C., & McKay, A. (2018). *Introduction to Modeling and Simulation Techniques*. https://eprints.whiterose.ac.uk/135646/

Zafar, M. H., Langås, E. F., & Sanfilippo, F. (2024). Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review. In *Robotics and Computer-Integrated Manufacturing* (Vol. 89). Elsevier Ltd. https://doi.org/10.1016/j.rcim.2024.102769

Zhang, J., Yang, X., Wang, W., Guan, J., Ding, L., & Lee, V. C. S. (2023). Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering. In

Automation in Construction (Vol. 146). Elsevier B.V. https://doi.org/10.1016/j.autcon.2022.104699

Zoubek, M., & Simon, M. (2021). Evaluation of the level and readiness of internal logistics for industry 4.0 in industrial companies. *Applied Sciences (Switzerland)*, *11*(13). https://doi.org/10.3390/app11136130