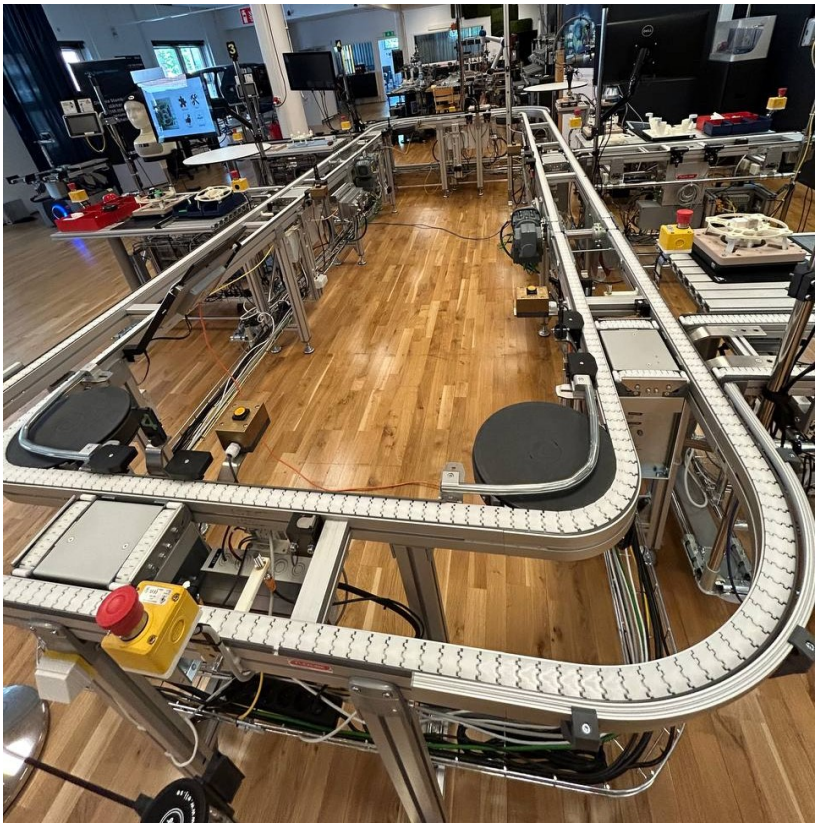
CHALMERS
UNIVERSITY OF TECHNOLOGY



Digital Twin Development and Long-Term Simulation of a Cyber-Physical Production System

Master's thesis in Master Information and Communication Technology

Saleena Punathikoyiloth

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Digital Twin Development and Long-Term Simulation of a Cyber-Physical Production System

Saleena Punathikoyiloth

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Chalmers University of Technology
Gothenburg, Sweden 2026

Digital Twin Development and Long-Term Simulation of a Cyber-Physical Production System
Saleena Punathi koyiloth

© Saleena Punathikoyiloth, 2026.

Supervisor: Silvan Marti, Department of Industrial and Materials Science
Examiner: Björn Johansson, Department of Industrial and Materials Science

Master's Thesis 2026
Department of Industrial and Materials Science
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover:

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Digital Twin Development and Long-Term Simulation of a Cyber-Physical Production System

Saleena Punathikoyiloth
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

This thesis presents the development and verification of a digital model for the WS Conveyor system located at the SII-Lab. The conveyor system, used for drone assembly, consists of six docking stations arranged along an oval belt where pallets carrying products circulate continuously between workstations. The digital model was implemented in Python as a discrete-time simulation that replicates the control logic specified in the original CODESYS PLC documentation, with every parameter traceable to a specific page in the reference document.

The simulation models pallet movement along a 9.4-meter belt at configurable speeds, sensor-based detection at each docking station, and timer-controlled auto-pass behavior. All physical measurements and operational details were confirmed through on-site visits and a semi-structured interview with the system architect. Verification was carried out through 95 automated tests using the pytest framework, achieving 100% code coverage across all simulation modules. Face validity was established through a structured review with the system architect following the procedure described by Sargent (2013).

A 26-scenario parameter-sensitivity analysis revealed that station dwell time is the active-period bottleneck governing system throughput: reducing the auto-pass timer from 10 to 2 seconds nearly doubled throughput, while varying belt speed from 40% to 100% had no measurable effect. Worker processing speed was found to be the dominant factor in shift production: an all-experienced workforce (2–2.5 min per pallet) produced 58% more output than an all-beginner workforce (3–5 min per pallet). The pallet-count relationship saturated beyond 8 pallets when workers became the binding constraint.

The system is classified as a digital model in the Kritzinger et al. (2018) sense, with manual data transfer from the physical system. The thesis discusses the path toward a fully connected digital twin using the ISA-95 automation-pyramid framework, and identifies camera-based pallet tracking, process mining of event logs, and predictive maintenance as future extensions.

Keywords: digital twin, digital model, discrete-time simulation, conveyor system, PLC, Industry 4.0, worker behavior, bottleneck analysis, verification, Python, SII-Lab.

Acknowledgements

I would like to express my sincere gratitude to **Silvan Marti** and **Björn Johansson** for warmly welcoming me into the Industrial and Material Science section and for their continuous guidance throughout my master's journey. Their patience, encouragement, and willingness to provide clear direction whenever I needed clarification have been invaluable from the very beginning until the successful completion of this thesis. Their academic support and professional insights greatly contributed to both my research work and personal development.

I am especially thankful to **Sven Ekered** for the practical assistance and support provided during the course of this research, particularly in arranging real-time data essential for my thesis work. His support and cooperation played a significant role in the successful execution of this study.

I would also like to acknowledge all my former teachers and mentors whose encouragement, knowledge, and inspiration motivated me to continue learning and growing throughout my academic journey. A special note of appreciation goes to **Lena Sommarström**, my study advisor, whose constant motivation, valuable advice, and timely guidance always encouraged me during challenging times.

My deepest and most heartfelt gratitude goes to my husband, **Firoz Mohmed**, who has been my strongest pillar of support throughout this journey. His unwavering encouragement, technical assistance, financial support, and dedication in managing our family responsibilities gave me the strength and confidence to successfully pursue and complete my master's Thesis.

Finally, I would like to sincerely thank my family and friends for always believing in me, motivating me, and filling my journey with positivity and energy.

I would also like to acknowledge the assistance of AI tools such as *Claude* and *Grok*, *Codex* which helped refine sentence structure and improve the clarity and readability of my writing throughout this thesis.

Saleena Punathikoyiloth, Gothenburg, june 2026

List of Acronyms

AMR	Autonomous Mobile Robot
CSV	Comma-Separated Values
DES	Discrete-Event Simulation
DS	Docking Station
GVL	Global Variable List
HMI	Human-Machine Interface
PLC	Programmable Logic Controller
POU	Program Organization Unit
RFID	Radio-Frequency Identification
SCADA	Supervisory Control and Data Acquisition
SFC	Sequential Function Chart
SII-Lab	Smart Industry and Innovation Laboratory
WS	Workstation

Contents

List of Figures

List of Tables

1
Introduction

Manufacturing systems have grown increasingly complex over the past decades, incorporating automated conveyors, programmable logic controllers, and sensor based feedback loops to manage production flow. As these systems become more sophisticated, understanding their behavior under different operating conditions becomes correspondingly more difficult. Physical experimentation on a running production line is often impractical it is expensive, disruptive, and in some cases carries safety risks. This has motivated the adoption of simulation based approaches where a software representation of the physical system can be used to explore scenarios, identify inefficiencies, and test changes without interfering with real operations.
The concept of a digital twin provides a structured framework for this kind of simulation. A digital twin is a virtual replica of a physical system that mirrors its behavior, structure, and operational logic. Depending on the level of integration, it may range from a standalone simulation model with manual data input to a fully connected system with automatic, bidirectional data exchange between the physical and virtual domains [2]. Lopes et al. [16] identify simulation as the enabling technology for digital twins, positioning what if scenario analysis as the canonical use case through which digital twins generate operational value. The term has gained considerable traction within the Industry 4.0 discourse, where it is positioned as a foundational technology for smart manufacturing, predictive maintenance, and production optimization.

1.1 Background

This thesis is situated within the STENA INDUSTRY INNOVATION LAB (SII), which houses a workstation conveyor system referred to as the WS Conveyor used for the manual assembly of small drones. The system consists of an oval conveyor belt approximately 9.4 meters in length, along which pallets circulate between six docking stations. At each station, a worker may perform assembly tasks on the product carried by the pallet, after which the pallet continues to the next station. The entire system is controlled by a PLC programmed in CODESYS, with sensor based logic governing pallet detection, mission assignment, and timing.
Despite its relatively compact size, the WS Conveyor exhibits behaviors representative of larger industrial conveyor systems: queuing, timing dependent throughput, station utilization imbalances, and sensitivity to worker performance. Salunkhe et al. [12] argue that laboratory scale cyber physical production system (CPPS) testbeds are an appropriate research vehicle for digital twin development, providing

a controlled environment where system behavior can be fully observed and parameters freely varied. The SII Lab conveyor system fits this description, and its compact scale is therefore a methodological asset rather than merely a limitation.

1.2 Aim and Objectives

The aim of this thesis is to develop, verify, and analyze a digital twin of the WS Conveyor system using Python. The work is organized around four objectives:

1. **System logic implementation.** Translate the control logic documented in the PLC reference into executable Python code, modeling the conveyor belt, docking stations, sensors, pallets, and timing mechanisms.
2. **Verification and testing.** Establish confidence in the simulation through systematic automated testing, ensuring that every implemented rule corresponds to a documented behavior in the PLC specification.
3. **Scenario analysis.** Use the simulation to explore how changes in system parameters such as the number of pallets, belt speed, and auto pass timer duration affect throughput and station utilization.
4. **Worker behavior modeling.** Extend the simulation to incorporate human workers with empirically derived processing times, experience dependent performance, and shift based scheduling including breaks.

The system analysis that underpins the simulation drew on three complementary sources: the PLC program documentation, on site observation during factory visits, and a semi structured interview with the system architect. This three source approach is described in detail in Chapter 3.

1.3 Scope and Limitations

The simulation developed in this thesis replicates the sensor based control logic of the WS Conveyor system as documented in the CODESYS PLC program [3]. Physical quantities such as belt length and station spacing were confirmed through on site measurement, and worker behavior parameters were collected through a semi structured interview with the system architect.

Several aspects of the real system fall outside the scope of this work. The simulation does not model the pneumatic subsystem, the RFID based pallet identification, or the HMI operator interface. Queue collision behavior between pallets on the belt is simplified pallets are assumed not to physically interact. The system currently operates as a digital model with manual data transfer; no live data connection to the PLC exists. These limitations are discussed further in Chapter 6, along with recommendations for future work. As noted above, the testbed nature of the SII Lab system is treated as an asset for controlled experimentation [12] rather than purely as a constraint on generalizability.

1.4 Research Questions

The thesis is guided by the following research questions:

1. How accurately can the WS Conveyor system behavior be replicated using a discrete time simulation based on PLC documentation?
2. How do changes in system parameters pallet count, belt speed, and auto pass timer affect system throughput and station utilization?
3. What is the impact of worker processing speed and experience level on overall production output during an 8 hour shift?

1.5 Report Structure

The remainder of this report is organized as follows. Chapter 2 provides background on digital twins, simulation modeling, bottleneck analysis, and the Industry 4.0 context. Chapter 3 presents the analysis of the WS Conveyor system based on PLC documentation, factory visits, and the system architect interview. Chapter 4 describes the implementation of the digital twin in Python, including the software architecture and design decisions. Chapter 5 presents the results from verification testing, baseline simulation, and multi scenario analysis. Chapter 6 discusses the findings in relation to the research questions and identifies limitations. Chapter 7 concludes the thesis and outlines directions for future work.

2
Background and Theory

This chapter presents the theoretical foundations underpinning the thesis. It begins with an overview of the digital twin concept and its classification, followed by a discussion of discrete time simulation as the modeling approach used in this work. The chapter also situates the project within the broader Industry 4.0 framework, describes the PLC control environment, and reviews related work on conveyor system simulation and digital twin development.

2.1 Digital Twin

The term digital twin was originally introduced in the context of product lifecycle management, referring to a virtual representation of a physical product that could be used for simulation and analysis throughout its lifecycle [1]. Since then, the concept has been broadly adopted in manufacturing, infrastructure management, and process engineering, though its definition varies across domains and authors. Lopes et al. [16] identify simulation as the enabling technology for digital twins, arguing that what if scenario analysis is the canonical use case through which digital twins generate value in production contexts.
A widely cited classification was proposed by Kritzinger et al. [2], who distinguish three levels of integration between a physical system and its virtual counterpart:

- **Digital model.** A virtual representation with no automatic data exchange. Data is transferred manually between the physical and digital systems.
- **Digital shadow.** A one directional automatic data flow exists from the physical system to the digital model, allowing the model to reflect the current state of the real system.
- **Digital twin.** Automatic data flow exists in both directions. Changes in the physical system are reflected in the digital model, and insights from the digital model can be fed back to control or optimize the physical system.

It should be noted that the Kritzinger classification is one of several maturity models available in the literature; others include the frameworks proposed in ISO 23247 and by Grieves and Vickers [1]. My thesis adopts the Kritzinger model for its clarity and widespread adoption in the manufacturing digital twin literature.
A complementary perspective is offered by Subramaniyan and Skoogh [14], who describe a three element digital twin architecture consisting of (1) real time data acquisition from the physical system, (2) a continuously updated simulation model, and (3) a decision support system that feeds insights back to operations. The work presented in this thesis covers element (2) a simulation model built from documen-

tation and verified against the system architect's knowledge. Elements (1) and (3) automatic data acquisition and decision support are identified as future work in Chapter 7.
The work presented in this thesis falls within the first category of Kritzinger's classification a digital model. The simulation was built using documentation and on site observations, and data was transferred manually rather than through a live connection. The path toward a digital shadow or a fully connected digital twin is discussed in Chapter 6 as part of future work.

2.2 Simulation Modeling

Simulation refers to the process of designing a computational model of a system and conducting experiments with that model to understand the system's behavior or evaluate strategies for its operation [**?**]. In manufacturing contexts, simulation is commonly used to analyze throughput, identify bottlenecks, evaluate resource allocation, and test what if scenarios without disrupting real production.

2.2.1 Discrete Time vs. Discrete Event Simulation

In manufacturing simulation, the dominant paradigm is discrete event simulation (DES), in which the simulation advances from one event to the next without processing intermediate idle periods. DES is the default approach in most commercial manufacturing simulation tools and is well suited to systems where events are irregularly spaced in time [15].
The simulation approach used in this thesis is instead discrete time simulation (DTS), where the system state is updated at fixed time intervals. At each time step, the simulation advances all moving components, evaluates sensor conditions, updates timers, and logs events.
DTS was selected over DES for a specific technical reason: the PLC that controls the WS Conveyor executes a fixed scan cycle, evaluating all sensor inputs and updating all actuator outputs at regular intervals. DTS mirrors this execution semantics, exactly every input is evaluated on the same period, and all inter event intervals are captured, not only the events themselves.
A DES implementation would introduce artificial event ordering that does not correspond to the PLC's actual execution model. A time step of 0.1 seconds was chosen to ensure that pallet movement between sensor evaluations remains smaller than the sensor detection tolerance of 0.1 meters, preventing pallets from passing through sensor zones undetected.
The choice between DTS and DES has direct real world consequences for how faithfully a simulation reflects an actual PLC controlled system. A PLC operates as a clock-driven device: at each fixed scan cycle (typically every 10–100 ms), it samples every input, evaluates its logic, and writes every output regardless of whether any event has occurred.
DTS preserves this behavior by ticking at a fixed rate, capturing the in between intervals during which timers count, sensors are re evaluated, and state continues to evolve.

DES, by contrast, advances directly from event to event and skips these intervals, which would distort timer accuracy, sensor sampling, and the exact ordering of co incident events. For the WS Conveyor, this difference matters in practice: the 5 second auto-pass timer, the sensor activation tolerance, and the propagation of pallets between stations all depend on uniform, scan cycle based time progression.
A DES implementation would either need artificial event scheduling that does not correspond to the PLC's real execution model, or would suffer timing errors that would invalidate operational comparisons against the physical system.
Adopting DTS therefore aligns the simulation semantics with the real world control system and preserves the fidelity required for parameter sensitivity analysis and future validation against live PLC traces.

2.2.2 Simulation Verification and Validation

Verification and validation (V&V) are distinct but complementary activities in simulation development [4]. Verification asks whether the simulation has been built correctly, that is, whether the code accurately implements the intended model. Validation asks whether the right model has been built or whether the simulation adequately represents the real system for the intended purpose.
Sargent [4] describes several validation techniques, including comparison testing, face validity, and historical data validation. Face validity involves asking individuals knowledgeable about the real system to examine the simulation's behavior and outputs and judge whether they are reasonable. This technique is particularly applicable when access to real time operational data is limited.
In this thesis, verification was carried out through automated unit and integration testing using the pytest framework, achieving 100% statement coverage across all simulation modules. This approach aligns with the NASA Systems Engineering Handbook's recommendation for unit test based verification of simulation software, where statement coverage provides a minimum confidence level that all code paths have been exercised. Validation was approached through two complementary methods: parameter by parameter comparison with the PLC documentation, and a face validity review session with the system architect (Sven Ekered), who assessed whether the simulation outputs were consistent with his operational experience of the real system.

2.3 Bottleneck Analysis in Manufacturing Systems

The identification of bottlenecks the resources or processes that constrain overall system throughput is a central concern in manufacturing systems engineering. Subramaniyan and Skoogh [13] define the active period bottleneck as the station or resource whose active period (the time it is occupied or processing) most constrains the throughput of the system. Their framework distinguishes between momentary bottlenecks, which shift between stations over time, and long term bottlenecks, which persistently limit throughput.
This concept is directly relevant to the scenario analysis in this thesis. The simulation results demonstrate that station dwell time whether governed by an auto

pass timer or by worker processing speed dominates the cycle time and therefore determines throughput. Belt speed, by contrast, has negligible effect because transit time between stations is a small fraction of the total cycle. In the terminology of Subramaniyan and Skoogh, the docking stations are the active period bottleneck, and the conveyor belt is not a constraining resource. This framing is applied to the results in Chapter 5 and discussed further in Chapter 6.

2.4 Industry 4.0 and Smart Manufacturing

Industry 4.0 refers to the ongoing transformation of manufacturing through the integration of digital technologies, including the Internet of Things, cyber physical systems, cloud computing, and artificial intelligence [5]. Within this framework, digital twins are positioned as a means of bridging the gap between physical production systems and their digital counterparts, enabling data driven decision making, predictive maintenance, and autonomous optimization. Achouch et al. [22] provide a broader view of the Industry 4.0 technology landscape, identifying digital twins alongside predictive maintenance and smart sensors as key enablers of next generation manufacturing.
The WS Conveyor system at the SII Lab serves as a laboratory scale demonstrator for Industry 4.0 concepts. Salunkhe et al. [12] argue that laboratory scale cyber physical production system (CPPS) testbeds are an appropriate and effective research vehicle for investigating digital twin technologies, as they provide a controlled environment where system behavior can be fully observed and parameters freely varied. The SII Lab conveyor system fits this description: it is complex enough to exhibit realistic production dynamics while remaining manageable in scope for a thesis project. This reframes the system's relatively small scale from a limitation to a methodological asset.
While the current setup does not include automatic data exchange between the PLC and the simulation, the architecture of the digital model has been designed with extensibility in mind. The separation of configuration, logic, and simulation components allows future integration with live PLC data feeds, SCADA interfaces, or machine learning modules.

2.5 PLC and CODESYS

A Programmable Logic Controller (PLC) is an industrial computer designed to control manufacturing processes through sensor inputs and actuator outputs. The WS Conveyor system is controlled by a PLC programmed in CODESYS, a widely used development environment for industrial automation that supports the IEC 61131 3 standard [6].
The PLC program for the WS Conveyor uses Sequential Function Charts (SFC) to define the state machine behavior at each docking station. SFC is the IEC 61131 3 equivalent of GRAFCET, the graphical functional chart language widely used in European industrial automation [18]. This connection situates the WS Conveyor's control logic within the broader industrial control literature beyond a single vendor

implementation. Each station progresses through a defined sequence of states idle, pallet detected, safety pause, elevator operation, workspace transfer, and return with transitions governed by sensor inputs and timer conditions.
The complete PLC program is documented in a 39 page reference (Kod_WSConv_Sven2022.pdf) [3], which served as the primary specification for the simulation developed in this thesis.

2.6 Python Mocking and Simulation

The simulation in this thesis uses a mock up approach, where the real PLC hardware and its I/O signals are replaced by Python objects that replicate the same logical behavior. Each physical component docking station, conveyor motor, pallet is represented as a Python dataclass with attributes corresponding to the sensor states and control variables documented in the PLC program.
This approach is conceptually related to the software testing practice of mocking, where external dependencies are replaced with controlled stand ins to enable isolated testing [7]. In this case, the PLC hardware is the external dependency, and the Python classes serve as its mock replacement. The advantage is that the simulation can be run, tested, and modified without access to the physical system or the PLC development environment. Compared to commercial simulation tools such as Siemens Plant Simulation or AnyLogic, this approach offers three practical advantages: the code is fully open and version controlled, every function is individually testable via pytest, and the simulation replicates PLC control logic at scan cycle resolution rather than at a graphical abstraction level.

2.7 Related Work

This section reviews prior work in three areas relevant to the thesis: digital twin implementations for manufacturing systems, simulation approaches for conveyor based production, and worker behavior modeling.

2.7.1 Digital Twin Implementations

Martinez et al. [11] present a digital twin implementation for a laboratory scale production system consisting of a conveyor and workstations a setup structurally similar to the WS Conveyor at the SII Lab. Their work uses the ISA 95 automation pyramid to frame the integration between the physical system and its digital counterpart, progressing from the shop floor control level through MES to enterprise level decision support. The present thesis addresses the shop floor simulation level; the ISA 95 framing is revisited in Chapter 7 as a roadmap for future integration.
Lee et al. [17] propose a CPPS architectural framework structured around sensors, a cyber model, IoT connectivity, and MES integration. Their framework provides a useful reference for understanding where the current digital model sits within a larger system of systems architecture. Negri et al. [8] review the roles of digital twins in CPS based production systems, identifying simulation based analysis and

what if scenario testing as primary applications both of which are realized in this thesis.

2.7.2 Simulation Approaches for Manufacturing

Bokrantz and Skoogh [15] establish discrete event simulation as the dominant paradigm in manufacturing systems research, a context important for understanding why the discrete time approach taken in this thesis is a deliberate methodological choice rather than a default. As discussed in §2.2.1, the DTS approach is motivated by the PLC's fixed scan cycle execution model.
Liu et al. [9] review digital twin technologies and industrial applications, noting the growing use of open source and Python based tools for manufacturing simulation. The SimPy library provides a process based DES framework that has been applied to production line modeling; the approach taken in this thesis differs in both paradigm (DTS rather than DES) and motivation (PLC logic replication rather than abstract process modeling).
Dahmen et al. [21] describe a scenario based virtual testing methodology for simulation models, in which multiple configurations are systematically varied and the results compared. The 26 scenario analysis conducted in this thesis follows a similar approach, varying pallet count, belt speed, auto pass timer, and worker experience level to characterize system sensitivity.

2.7.3 Worker Behavior in Assembly Systems

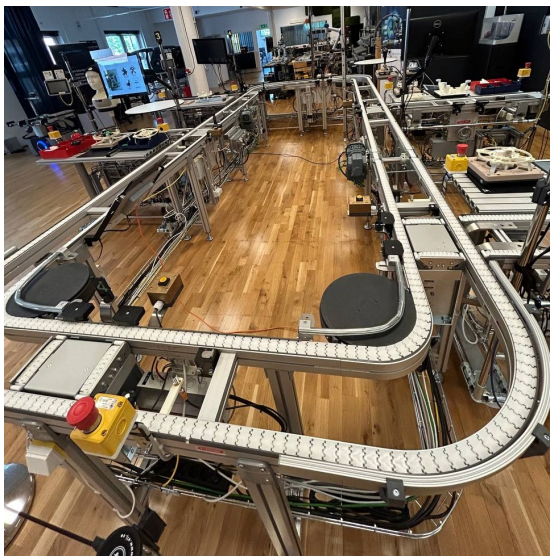
Processing time variability due to worker experience, fatigue, and task complexity has been shown to significantly affect throughput predictions in assembly line simulation [10]. Most studies model worker performance using assumed statistical distributions; the present thesis instead incorporates empirically collected timing data from a semi structured interview with the system architect at the SII Lab. The reported processing times (2 5 minutes) and the learning period (approximately 4 hours to reach experienced performance) are used directly as simulation parameters rather than as inputs to a fitted distribution. This empirical grounding is a methodological strength, though it is based on a single informant and should be interpreted as a calibration input rather than a generalizable finding about worker performance.
Laftchiev and Sun [20] demonstrate how PLC event relationship tables can be used for anomaly detection in manufacturing systems. The event logs produced by the simulation in this thesis are structurally equivalent to such PLC event sequences, opening the possibility of applying process mining or anomaly detection methods to the simulated data a direction identified as future work in Chapter 7.

3
System Analysis

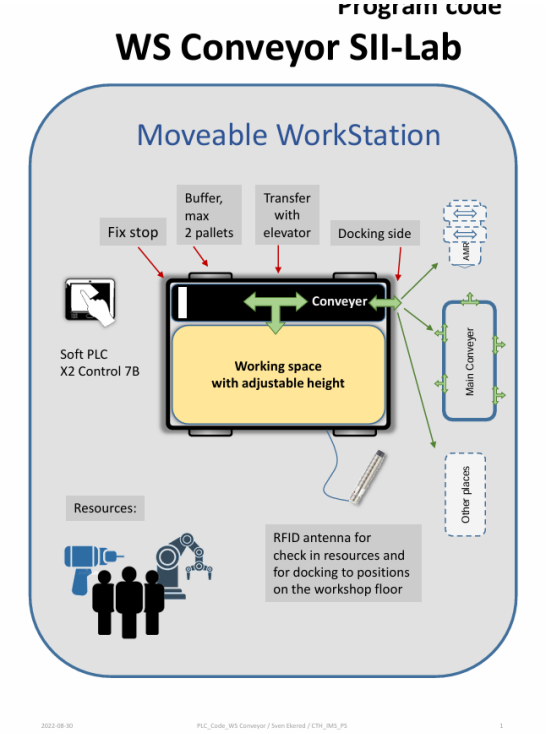
This chapter describes the WS Conveyor system and the process through which its behavior was analyzed prior to simulation development. The analysis drew on three sources: the PLC documentation, on site observation during factory visits, and a semi structured interview with the system architect. Together, these provided the specification against which the simulation was built and later verified.

3.1 The WS Conveyor System

The WS Conveyor is located at the Chalmers Smart Industry and Innovation Laboratory (SII Lab) and is used for the manual assembly of small drones. The system consists of an oval shaped conveyor belt with six docking stations (DS1 through DS6) positioned along its length. Pallets carrying products travel continuously around the belt, stopping at stations where workers perform assembly tasks before the pallet is released to continue its circuit.
The belt has a measured length of approximately 9.4 meters, with docking stations spaced at roughly 1.5 meter intervals. During normal operation, up to 10 pallets may circulate on the system, though the number is configurable. The belt operates at a default speed of 80% of its maximum capacity of 0.18 m/s, yielding an actual belt speed of 0.144 m/s.



(a) Photograph of the physical system.



(b) Schematic. Reproduced with permission from Ekered [3], page 1.

Figure 3.1: The WS Conveyor system at the Chalmers SII Lab. The photograph (left) shows the physical conveyor used for drone assembly, with the corresponding system schematic (right) identifying the six docking stations (DS1 DS6), the buffer queue, the elevator transfer, and the AMR connection at DS1.

Each docking station is equipped with two presence sensors — an entry sensor that detects an approaching pallet and a stop sensor that confirms the pallet is correctly positioned. Stations DS1 through DS6 share the same physical design, with two exceptions: DS1 is equipped with an Autonomous Mobile Robot (AMR) and cannot perform Mission 5, while DS6 also lacks Mission 5 capability due to a hardware limitation documented on page 23 of the PLC reference.

3.2 Hardware Architecture

The WS Conveyor system uses an industrial control architecture documented in detail in Ekered [3], page 3. The control logic runs on a Soft PLC X2 Control 7B executing the CoDeSys V3.5 SP13 Patch3 runtime. The PLC communicates with the field devices through an EtherCAT bus connected to a Crevis NA 9286 I/O module, which provides 16 digital outputs, 16 digital inputs, and 4 analog inputs. Additional sensors including IFM pneumatic sensors connect through an I/O Link Master that interfaces with the EtherCAT bus. An iX 2.40 SP5 HMI touchscreen provides the operator interface for control and monitoring. Two RFID antennas are mounted in the system: one movable antenna at the buffer queue position and one

fixed antenna at the elevator location, both used for pallet identification through 8 byte tag reading.

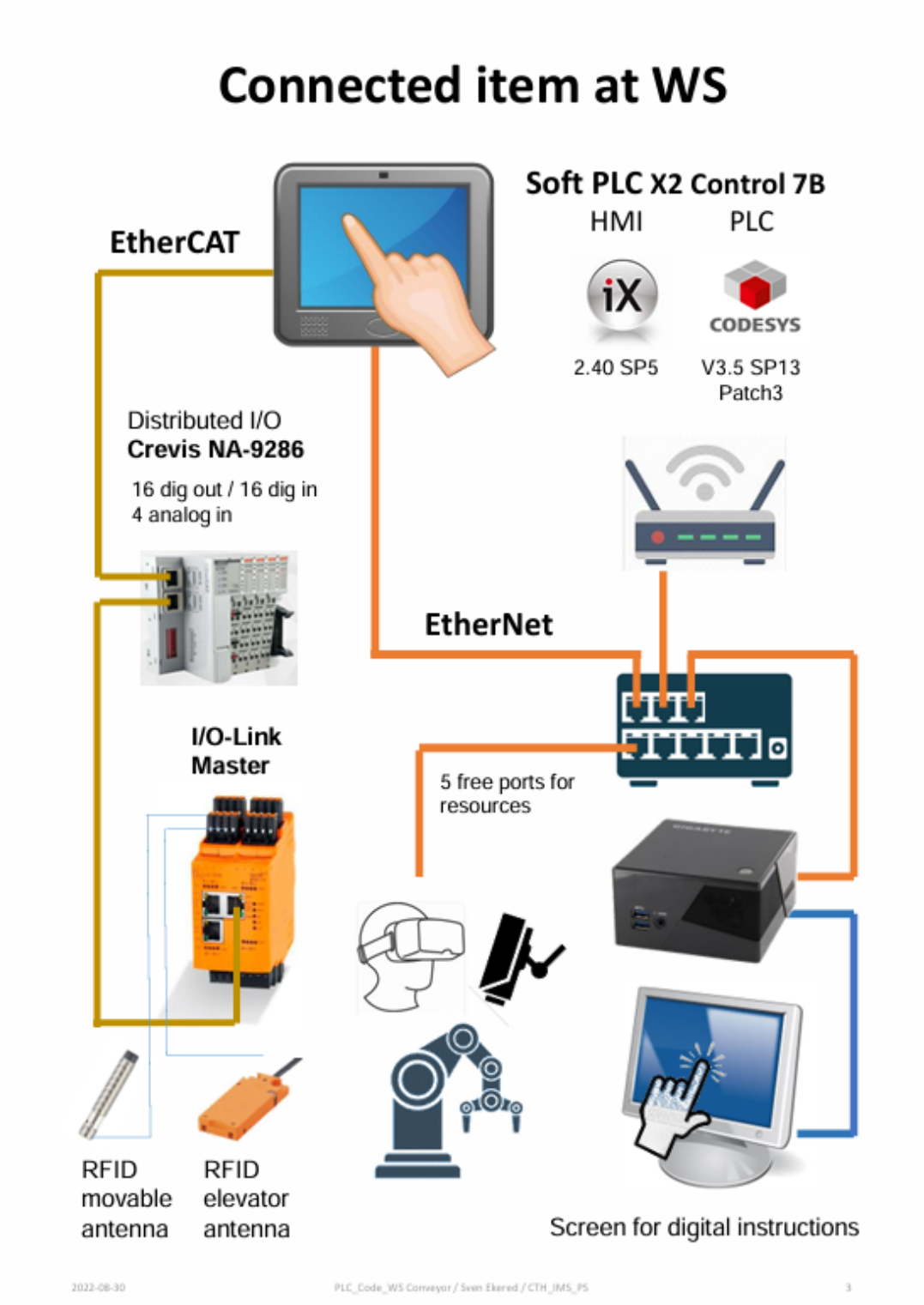


Figure 3.2: Hardware and network topology of the WS Conveyor system, showing the Soft PLC X2 Control 7B, the CoDeSys runtime environment, the Crevis NA 9286 I/O module connected over EtherCAT, the iX 2.40 SP5 HMI touchscreen, the RFID antennas, the I/O Link Master with IFM pneumatic sensors, and the AMR connection at DS1. Reproduced with permission from Ekered [3], page 3.

The real I/O signal names used by the PLC, documented on page 4 of [3], are listed in Table 3.1. These names are referenced directly in the Python simulation through the `DockingStation` class attributes, providing 1 to 1 traceability between the physical hardware variables and the simulation state.

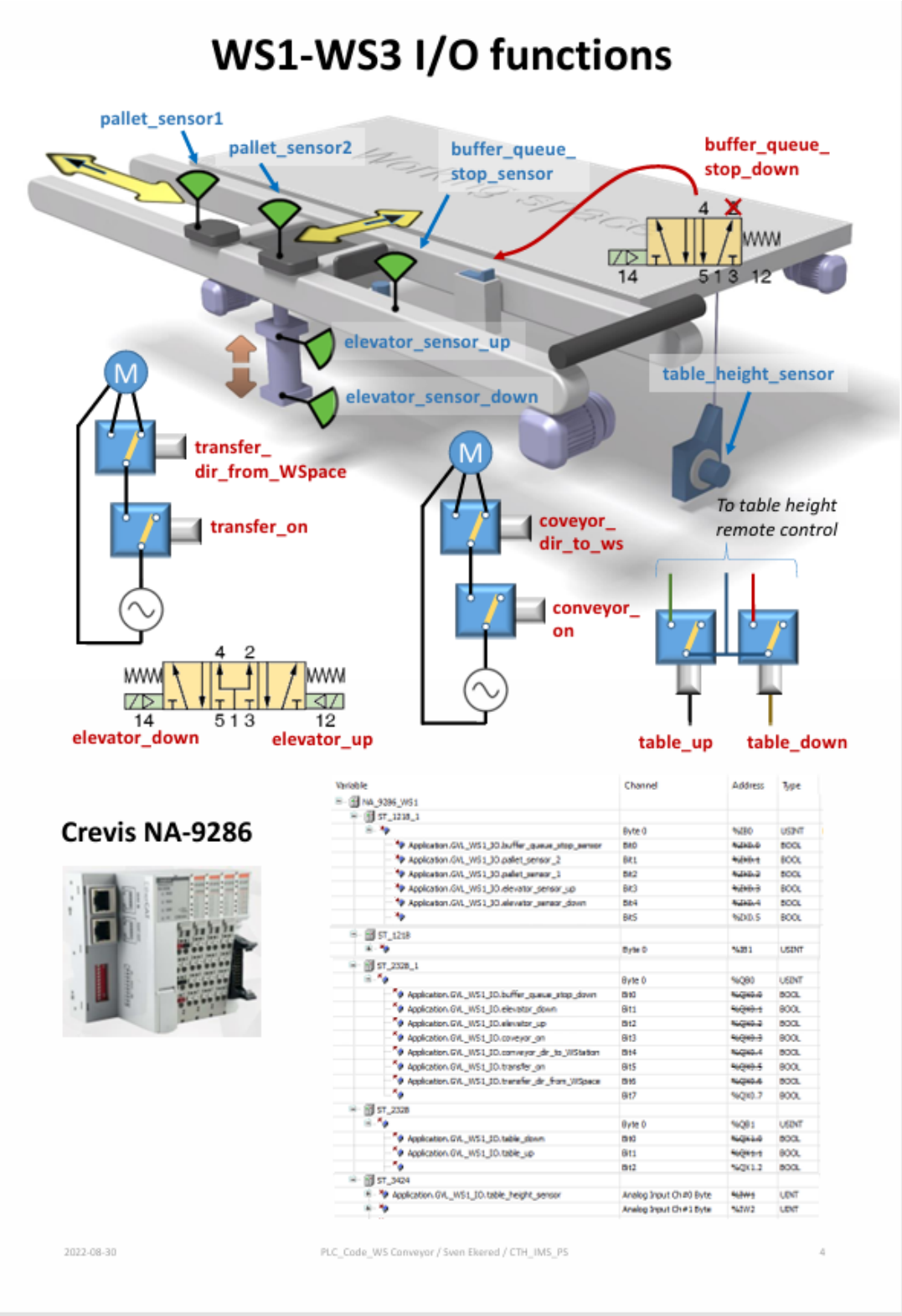


Figure 3.3: I/O functions diagram showing the real sensor and actuator signal names used by the PLC, including pallet detection sensors, elevator position sensors, conveyor motor controls, and table height sensors. Reproduced with permission from Ekered [3], page 4.

Table 3.1: Real I/O signal names from PLC documentation (Ekered, 2022, p. 4).

Signal Name (PLC)	Type	Meaning
`pallet_sensor1`	BOOL (I/P)	Pallet present at transfer position 1
`pallet_sensor2`	BOOL (I/P)	Pallet present at transfer position 2
`buffer_queue_stop_sensor`	BOOL (I/P)	Pallet stopped at buffer queue
`elevator_sensor_up`	BOOL (I/P)	Elevator at upper position
`elevator_sensor_down`	BOOL (I/P)	Elevator at lower position
`table_height_sensor`	UINT (I/P)	Worktable height (analog)
`buffer_queue_stop_down`	BOOL (O/P)	Lowers queue stop pin
`elevator_up` / `elevator_down`	BOOL(O/P)	Elevator motor commands
`conveyor_on`	BOOL (O/P)	Belt motor on/off
`conveyor_dir_to_ws`	BOOL (O/P)	Belt direction control
`transfer_on`	BOOL (O/P)	Transfer mechanism activation

3.3 Mission Taxonomy

The PLC program implements two parallel mission numbering systems, documented across pages 15 17 and page 23 of [3]. The WS Program Sequence governs the physical pallet movements (between Docking Side, WorkSpace, and Buffer), while the DS Mission Sequence governs queue and routing decisions at each station. Both systems use the same range 0 6 but with different semantics, which is a recurring source of confusion in the source documentation. Table 3.2 reproduces the taxonomy with both interpretations.

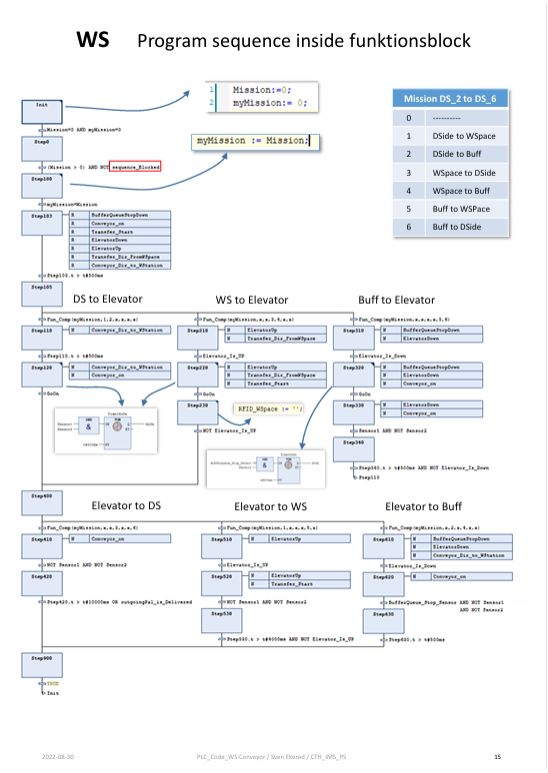
Table 3.2: Mission taxonomy from [3].

Mission	WS Program (physical)	DS Queue (routing)
0	No mission idle	No mission waiting
1	DSide → WSpace (elevator UP)	Pass DS (no stop)
2	DSide → Buffer (elevator DOWN)	MC to WS (DS2 DS6) / MC to Buffer
3	WSpace → DSide (elevator DOWN)	WS to MC (DS2 DS6) / AMR to MC
4	WSpace → Buffer (via conveyor)	(unused) / Buffer to AMR
5	Buffer → WSpace (elevator UP)	Stop at transfer (DS2 DS5; absent at DS6)
6	Buffer → DSide (via conveyor)	All stop down (all DS)

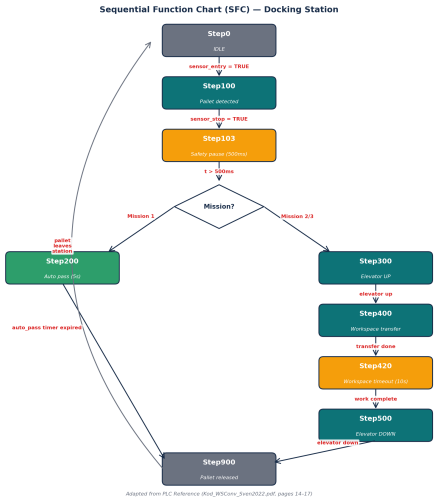
Two aspects of this taxonomy are reproduced as 1 to 1 design decisions in the simulation. First, the absence of Mission 5 at DS6 is enforced by the `get_valid_missions()` function in `logic.py`, which returns the list `[0, 1, 2, 3]` for DS6 and `[0, 1, 2, 3, 5]` for DS2 DS5. Second, the DS1 specific missions (involving the AMR robot) are recognised as a distinct configuration through a separate `has_amr` attribute on the `DockingStation` class. The traceability between the simulation parameters and the PLC source is summarised in 4.10.

3.4 Side by Side Comparison: SFC and Python State Machine

To demonstrate behavioural fidelity, Figure 3.4 pairs the original SFC mission sequence from Ekered [3], pages 15 17, against the Python state machine that implements the same logic. The SFC states (Step0, Step100, Step103, Step200/Step300, Step900) map directly to the Python execution flow (IDLE, DETECTED, WAIT, PASS/TRANSFER, RELEASED). Transition conditions on the PLC side (e.g., `Mission > 0`, `pallet_sensor1 = TRUE`, `t > 500ms`) correspond to function calls in the Python implementation (`station.mission > 0`, `check_sensor()`, `station.wait_timer_s > 0.5`). The PLC I/O signal names (italicised on the left) and the Python class methods (italicised on the right) make the 1 to 1 correspondence explicit.



(a) Original SFC mission sequence. Reproduced with permission from Ekered [3],



(b) Python state machine implementation showing the equivalent step sequence.

Figure 3.4: Side by side comparison of the original SFC state machine from the PLC documentation [3] (left) against the Python state machine implemented in this thesis (right). Step numbers, mission identifiers, transition conditions, and PLC I/O signal names are preserved 1 to 1 across both implementations, demonstrating behavioural fidelity.

3.5 PLC Documentation Analysis

The primary technical reference for this thesis is the CODESYS PLC program documentation, Kod_WSConv_Sven2022.pdf document containing the complete control logic for the WS Conveyor system. The analysis of this document produced the following key parameters, each traceable to a specific page:

Table 3.3: System parameters extracted from PLC documentation.

Parameter	Value	Unit	Source
Default belt speed	80	% of max	GVL_HMI
Maximum belt speed	0.18	m/s	HMI
Ramp time	1000	ms	GVL_HMI
Minimum speed	20	%	POU_HMI
Maximum speed	100	%	POU_HMI
Auto pass timer	5000	ms	
Safety pause	500	ms	Step103
Elevator timeout	10000	ms	Step420
Buffer capacity	2	pallets	Page 1
Number of stations	6	—	Page 6
DS6 Mission 5	No	—	Page 23

The PLC program uses Sequential Function Charts (SFC) to define the state machine at each docking station. The SFC progresses through a series of steps from idle (Step0) through pallet detection (Step100), safety pause (Step103), elevator operation (Step105/Step200), and workspace transfer (Step300/Step400) before returning to idle (Step900). Each transition is governed by sensor inputs and timer conditions specified in the documentation.

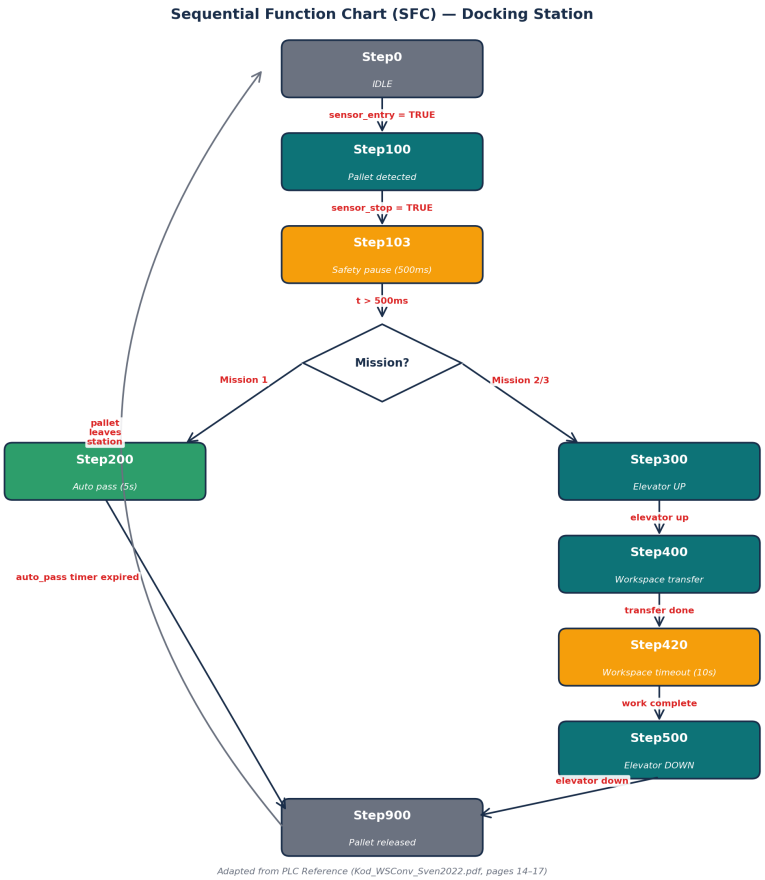


Figure 3.5: Sequential Function Chart (SFC) for one docking station, showing the progression from idle (Step0) through pallet detection, safety pause, and either auto pass (Step200) or workspace transfer (Steps 300 500). Adapted from the PLC reference documentation

The speed validation logic, documented on page 7, enforces a range of 20% to 100% of maximum speed. Values outside this range are rejected by the PLC. The auto pass mechanism, documented on pages 31 and 33, releases a pallet from a station after 5 seconds if no mission has been assigned, preventing pallets from blocking stations indefinitely.

3.6 Factory Visits

Two visits to the SII Lab were conducted during the project. The first visit focused on observing the physical system, photographing the docking stations, and confirming the general layout. The second visit included a semi structured interview with the system architect, Sven, which is described in the following section.
During the visits, the conveyor belt length was measured at approximately 9.4 meters, and the station spacing was observed to be roughly 2 meters between adjacent stations. These measurements were used to update the simulation parameters, which had previously been estimated at 9.0 meters and 1.2 meters respectively.

3.7 System Architect Interview

A semi structured interview was conducted with the system architect to confirm system behaviors not fully documented in the PLC reference and to collect empirical data on worker performance. The interview was audio recorded with permission and later transcribed for analysis. The key findings are summarized below.

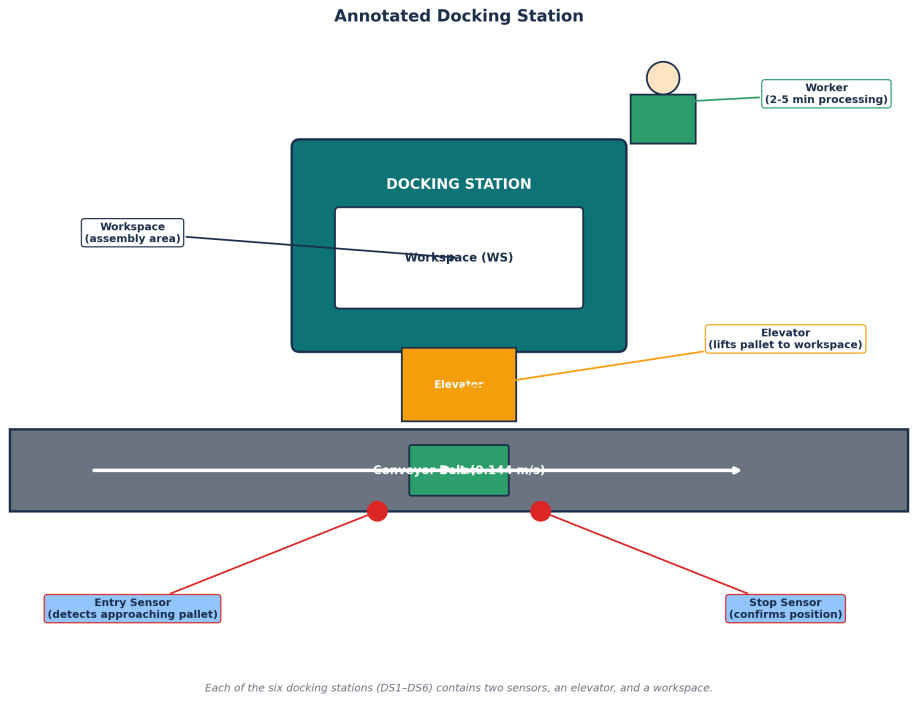


Figure 3.6: Annotated docking station showing the two sensor configuration (entry and stop sensors), the elevator mechanism, the workspace area, and the position of a pallet on the belt. Each of the six docking stations follows this layout. Adapted from PLC reference documentation.

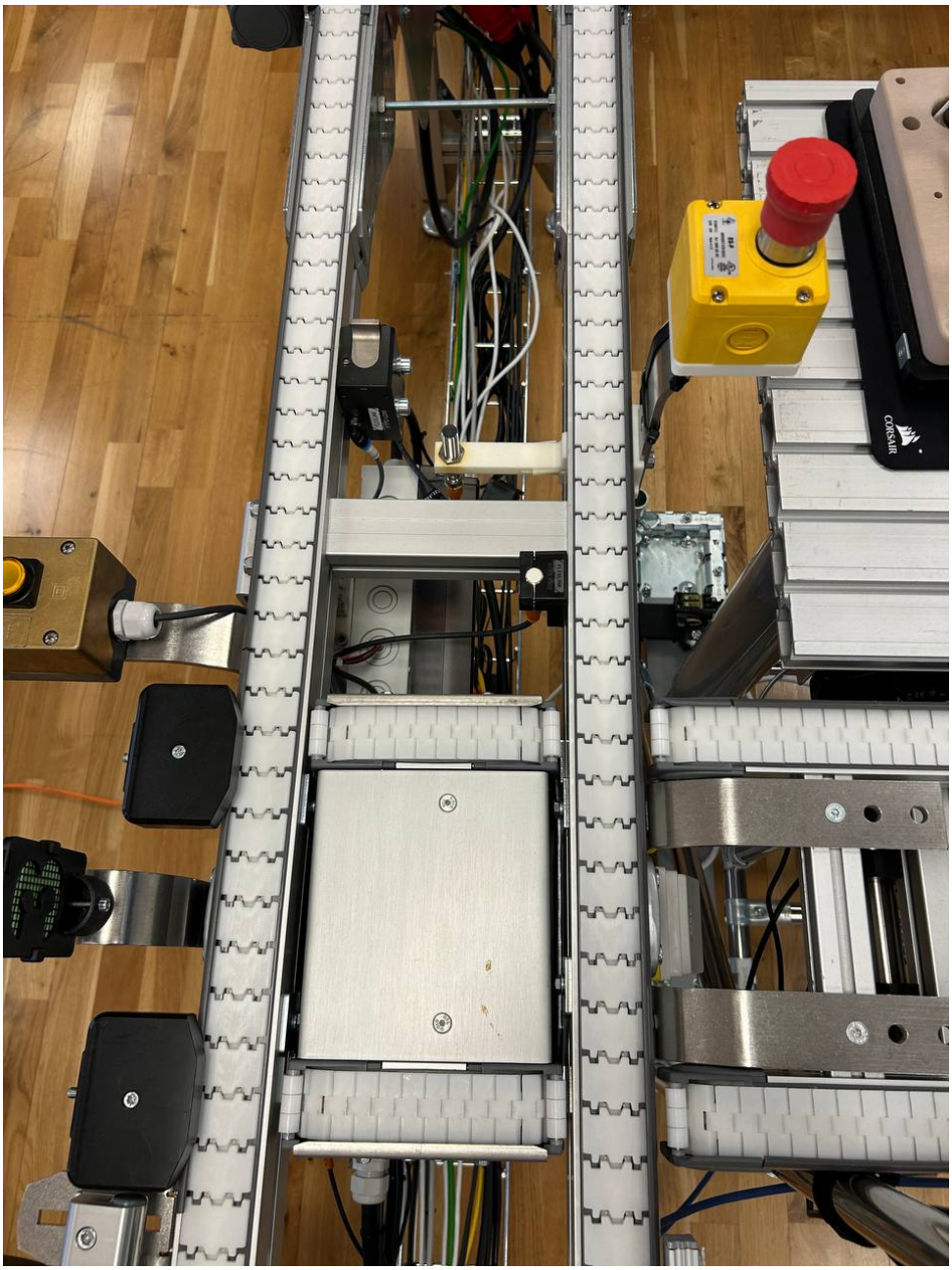


Figure 3.7: Photograph of one docking station on the WS Conveyor, showing the two pallet sensors, the elevator mechanism, and the workspace area above the belt. The same configuration is repeated for stations DS1 through DS6.

3.7.1 Sensor Configuration

The system architect confirmed that each docking station has two sensors: an entry sensor that detects an approaching pallet, and a stop sensor that confirms the pallet has reached the correct position. Both sensors must read true for the station to register a pallet as present. This two sensor configuration was not explicitly clear from the PLC documentation alone.

3.7.2 Pallet Routing

When a pallet arrives at a station, the system faces a binary decision: the pallet either passes through (Mission 1) or enters the station via the elevator mechanism (Mission 2 or 3). In the case of entry, the elevator lifts the pallet from the belt and a local conveyor moves it to the workspace. The mission assignment is handled automatically by the SCADA system, not by the worker.
Pallets do not necessarily visit stations in sequential order from DS1 to DS6. If a target station is occupied, the pallet completes another full loop of the belt and attempts again on the next pass.

3.7.3 Queue and Stuck Pallet Behavior

The system architect confirmed that pallets can queue behind occupied stations, with only one pallet passing at a time. He also confirmed that pallets can occasionally become stuck between stations. The expected travel time between adjacent stations is approximately 60 seconds, and exceeding this time indicates a fault. However, the current sensor based system cannot distinguish between a pallet waiting in a queue and a pallet that is stuck a known limitation that would require a camera system to resolve.

3.7.4 Worker Performance Data

Processing times for drone assembly were reported as follows: a typical experienced worker completes one pallet in approximately 2 to 3 minutes, with 2 minutes being the fastest observed time and 5 minutes being the slowest. New workers reach comparable skill levels after approximately half a day (4 hours) of practice, after which the processing time difference is roughly 20%. The system architect noted that more than 80% of workers on a typical shift are experienced.

3.7.5 Shift and Break Schedule

The system operates on a single 8 hour shift from 08:00 to 17:00. A total of six workers are assigned per shift five active on the belt and one backup worker who is trained on all stations and covers for short breaks such as toilet visits. The belt stops during three scheduled breaks: a 15 minute morning coffee break at 10:00, a 30 minute lunch break at 12:00, and a 15 minute afternoon coffee break at 14:30. The total scheduled break time is 60 minutes, leaving 7 productive hours per shift.

3.7.6 System Limitations

The system architect confirmed that the WS Conveyor currently uses only sensor based logic for pallet tracking. No camera or computer vision system is connected. He indicated that camera integration would be technically feasible and could address the inability to distinguish queued from stuck pallets, but this has not been implemented. The digital twin developed in this thesis replicates the sensor only logic, matching the real system's capabilities and limitations.

4

Implementation

This chapter describes the software implementation of the WS Conveyor digital twin. The simulation was developed in Python and structured as a modular codebase with clear separation between configuration, component modeling, decision logic, system state management, simulation execution, and event logging. The full source code is available in the project GitHub repository.

4.1 Software Architecture

The simulation consists of eight Python modules organized into three packages. The `mock_up` package contains the system model configuration, physical components, control logic, system state, and worker behavior. The `simulation` package contains the simulation engine. The `logging` package handles event recording.

Table 4.1: Python modules and their responsibilities.

Module	Package	Responsibility
`config.py`	mock_up	System parameters from PLC documentation
`components.py`	mock_up	DockingStation, ConveyorMotor, Pallet classes
`logic.py`	mock_up	Speed validation, sensor detection, timer rules
`state.py`	mock_up	SystemState — central state management
`worker.py`	mock_up	Worker behavior and experience modeling
`shift.py`	mock_up	Shift schedule and break management
`simulator.py`	simulation	Simulation engine — main loop
`logger.py`	logging	CSV event recording

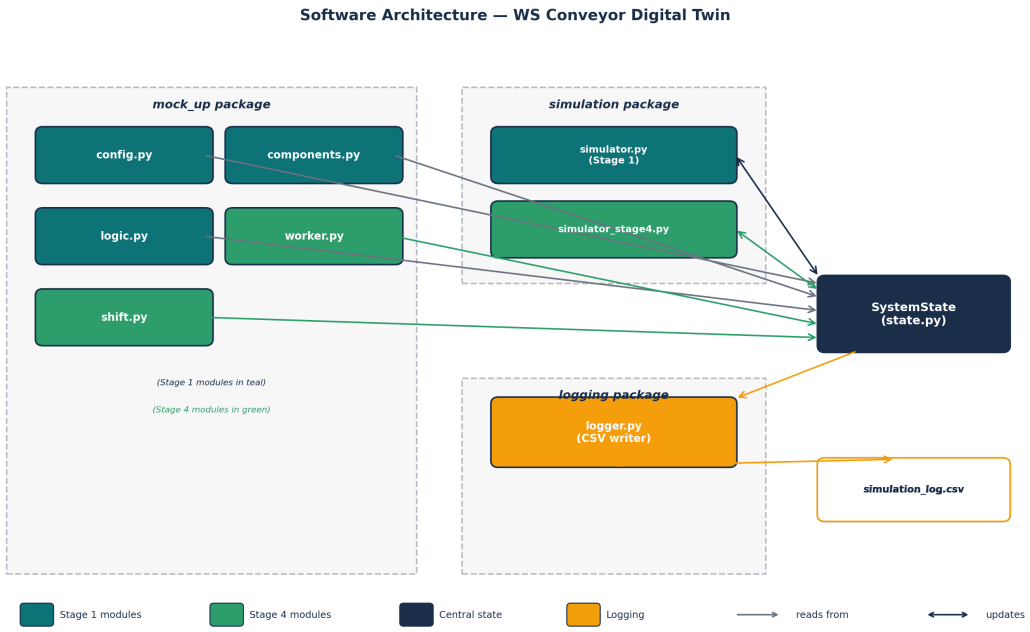


Figure 4.1: Software architecture of the digital twin showing the eight Python modules organized into three packages (mock_up, simulation, logging). All modules read from and write to the central SystemState object, which serves as the single source of truth for the simulation. Stage 1 modules are shown in teal; Stage 4 worker behavior modules are shown in green.

The design follows a principle of single responsibility each module handles one aspect of the system, and modules communicate through the shared `SystemState` object. This architecture was adopted on the explicit instruction of the supervisor, who recommended building the docking station class first and then mirroring it for all six stations rather than creating separate implementations.

4.2 Configuration Module

The `config.py` module stores all system parameters in a single Python dataclass. Every numerical value used in the simulation is defined here, and each value is annotated with its source in the PLC documentation. This centralized approach ensures that parameter changes propagate automatically to all modules that reference them. The configuration includes conveyor parameters (speed, ramp time, belt length), timing values (auto-pass timer, safety pause, elevator timeout), station topology (count, spacing), and worker behavior parameters collected from the system architect interview (processing times, experience boost, shift schedule, break durations). Physical dimensions were updated after the factory visit: the belt length was corrected from an initial estimate of 9.0 meters to the measured value of 9.4 meters, and the station spacing was adjusted based on on-site observation.

4.3 Component Modeling

Three classes in `components.py` represent the physical elements of the conveyor system.
`DockingStation` is a dataclass modeling one station. A single class definition is used for all six stations, with parameters such as `has_amr` and `has_mission5` distinguishing the behavior of DS1 and DS6 from the standard stations DS2 through DS5. Each station tracks its sensor states, pallet presence, wait timer, current mission, and SFC step.
`ConveyorMotor` represents the belt drive. It stores the current speed percentage and provides a method `get_actual_speed()` that calculates the absolute speed in meters per second based on the configured maximum speed.
`Pallet` tracks the position, movement state, and station association of each pallet on the belt. The `just_passed` attribute implements a cooldown mechanism that prevents a pallet from immediately retriggering the sensor of a station it has just left.

4.4 Decision Logic

The `logic.py` module contains four functions that implement the decision rules from the PLC documentation.
`validate_speed()` enforces the 20–100% speed range specified on page 7 of the PLC reference. Speeds outside this range are rejected by returning `None`.
`check_sensor()` determines whether a pallet is within detection range of a station sensor. The function calculates the absolute distance between the pallet position and the sensor position, returning `True` if the distance is within a configurable tolerance of 0.1 meters.
`check_auto_pass()` checks whether the auto-pass timer has expired. If the pallet has been waiting at a station for 5 seconds (5000 ms, from pages 31 and 33 of the PLC reference), the function returns `True` and the pallet is released.
`get_valid_missions()` returns the list of missions available at a given station. DS1 (AMR-equipped) and DS6 (hardware limitation) return missions [0, 1, 2, 3], while DS2 through DS5 return [0, 1, 2, 3, 5].

4.5 System State

The `state.py` module defines the `SystemState` class, which serves as the single source of truth for the entire simulation. It holds references to one `ConveyorMotor`, six `DockingStation` objects, and a configurable number of `Pallet` objects, along with the simulation clock.
At initialization, pallets are distributed evenly along the belt using the station spacing parameter, ensuring they begin at different positions rather than clustering at the origin. The simulation clock starts at zero for Stage 1 analysis or at 28800 seconds (08:00) for Stage 4 shift-based simulation.

Every other module reads from and writes to this single state object. This avoids the consistency issues that would arise if different modules maintained independent copies of pallet positions or station states.

4.6 Simulation Engine

The simulation engine in `simulator.py` executes a fixed-timestep loop that advances the system state every 0.1 seconds. Each iteration performs three operations in sequence.
First, `move_pallets()` advances the position of every non-stopped pallet by the product of belt speed and time step. For the default configuration, this is $0.144 \times 0.1 = 0.0144$ meters per step. When a pallet's position exceeds the belt length, it resets to zero, modeling the oval loop.
Second, `check_stations()` evaluates two categories of pallets. Stopped pallets have their wait timers incremented; if the timer reaches the auto-pass threshold, the pallet is released. Moving pallets are checked against each station's sensor position; if a pallet is within detection range of a free station, it is stopped and the timer begins.
Third, the simulation clock is incremented by the time step.

The choice of 0.1 seconds as the simulation timestep was driven by the physical constraints of the WS Conveyor system rather than chosen arbitrarily.
At the default belt speed of 0.144 m/s, each 0.1-second step advances a pallet by approximately 1.44 cm, which is well below the sensor detection tolerance of 10 cm. This ensures that no pallet can pass through a sensor zone undetected within a single timestep.
A coarser timestep for example, 0.5 seconds would advance each pallet by approximately 7.2 cm per step, which remains below the detection tolerance but introduces three concerns. First, the 5 second auto-pass timer would have only 10 time steps of resolution, leaving the exact moment of timer expiry vulnerable to a one-step rounding error and producing systematic bias in throughput measurements. Second, events that occur within the same 0.5-second window would be lumped together, losing information about their true ordering, which is relevant when pallets approach multiple stations simultaneously.
Third, the simulation would lose the close correspondence with the PLC scan cycle, which typically operates at 10 to 100 ms. Conversely, a finer timestep such as 0.01 seconds would not improve fidelity, because sensors and PLC inputs cannot resolve changes at that scale; it would simply multiply the computational cost tenfold without adding accuracy. The 0.1-second timestep therefore represents the largest value that preserves both sensor and timer fidelity, while keeping the 24-hour simulation computationally tractable.
For a 24-hour simulation at 0.1-second resolution, the loop executes 864,000 iterations. A one-hour simulation is completed in approximately 10 seconds on a standard laptop.

4.7 Worker Behavior Module

The `worker.py` module, added in Stage 4, models individual workers at each docking station. The `Worker` class tracks the worker's experience level, current task progress, total pallets processed, and break status.
Processing times are drawn from a uniform distribution bounded by the values confirmed by the system architect: experienced workers process pallets in 120 to 150 seconds (2 to 2.5 minutes), while beginners take 150 to 300 seconds (2.5 to 5 minutes). A worker transitions from beginner to experienced after 4 hours of accumulated work, reflecting the system architect's observation that learning takes approximately half a day.
The `create_shift_workers()` factory function generates six workers for a shift five assigned to stations DS1 through DS5 and one backup worker. Following the system architect's report that more than 80% of workers are typically experienced, five workers are initialized as experienced and one as a beginner.

4.8 Shift and Break System

The `shift.py` module models the shift schedule and break structure. The `Shift` class defines the working hours (08:00 to 17:00) and three scheduled breaks: a 15-minute morning coffee break, a 30-minute lunch break, and a 15-minute afternoon coffee break. During scheduled breaks, the conveyor belt is stopped and all workers are placed on break, consistent with the system architect's description of real operations.
The `should_belt_stop()` method is called every simulation step to determine whether the belt should be running or stopped at the current time. This integrates the shift and break logic directly into the simulation loop without modifying the core pallet movement or sensor detection code.

4.9 Event Logging

The `logger.py` module records simulation events to a CSV file with five columns: timestamp in seconds, event type, station ID, pallet ID, and a free-text details field. Four event types are logged in Stage 1 (SIMULATION_START, PALLET_ARRIVE, AUTO_PASS, SIMULATION_END), with three additional types in Stage 4 (WORKER_START, WORKER_DONE, BREAK_START/END, SHIFT_END). The CSV format was chosen for its simplicity and compatibility with analysis tools. The 24-hour Stage 1 simulation produces 55,686 event records, which are subsequently read by the analysis modules for throughput calculation and visualization.

4.10 Traceability to PLC Source

A key strength of the simulation is its 1-to-1 traceability to the PLC documentation in Ekered [3]. Every numerical parameter, signal name, and decision rule used in the

Python implementation maps to a specific page in the source document. Table 4.2 lists the most important correspondences. This explicit mapping is what allows the verification testing described in Chapter 5 to be meaningful: each test confirms that the Python behaviour matches the documented PLC behaviour, not an idealised model of a conveyor.

Table 4.2: Traceability between simulation parameters and PLC documentation (Ekered, 2022).

Simulation parameter	PLC source variable	Document page
HMI delay parameters		
`wait_at_queue_ms = 5000`	`hmi_TimeForWaitAtQueue`	p. 33
`wait_after_passing_ms = 1000`	`hmi_TimeForWaitAfterPassingDS`	p. 33
`wait_after_ws_ms = 4000`	`hmi_TimeForWaitAfterTransferdToWS`	p. 33
`wait_after_amr_ms = 4000`	`hmi_TimeForWaitAfterTransferdToAMR`	p. 33
Conveyor bounds		
`conveyor_speed_pct = 80`	`hmi_conveyor_speed` (default)	p. 33
`speed_min_pct = 20`	validation in POU_HMI_Control	p. 7
`speed_max_pct = 100`	validation in POU_HMI_Control	p. 7
`conveyor_ramp_ms = 1000`	`hmi_conveyor_ramp` (default)	p. 33
`ramp_min_ms = 100`	validation in POU_HMI_Control	p. 7
`ramp_max_ms = 5000`	validation in POU_HMI_Control	p. 7
Mission taxonomy		
`get_valid_missions(DS1)`	DS1 row of Table 10	p. 23
`get_valid_missions(DS2..DS5)`	DS2–DS5 rows of Table 10	p. 23
`get_valid_missions(DS6)` [0,1,2,3]	DS6 has no Mission 5 (hardware)	p. 23
SFC step timing		
`sfc_safety_pause_ms = 500`	Step103 reset duration	p. 15
`elevator_timeout_ms = 10000`	Step420 workspace timeout	p. 15
Buffer		
`buffer_max_capacity = 2`	Buffer max 2 pallets	p. 1

The four HMI delay parameters deserve particular emphasis: the simulation uses the exact default values from the PLC's GVL_HMI , and the speed/ramp validation bounds in `logic.py` replicate the validation logic of POU_HMI_Control exactly. The mission taxonomy 0 to 6 is reproduced in full, with the DS6 hardware limitation enforced as a deliberate design decision rather than an oversight.

4.11 Version Control

The project was managed using Git with a feature-branch workflow. Each module was developed in a separate branch named with the module name and creation date (e.g., `feature/config-260305`).The complete source code, test suite, and simulation logs are publicly available in the project GitHub repository:

`https://github.com/Saleenafiroz/ChalmersThesis`

5

Results and Analysis

This chapter presents the results from verification testing, baseline simulation, multi-scenario analysis, and worker behavior simulation. The parameter-sensitivity analysis is framed using the active-period bottleneck concept from Subramaniyan and Skoogh [13], which provides a structured vocabulary for identifying the system resource that most constrains throughput. The analysis progressed through four stages: establishing correctness through testing, characterizing baseline behavior, exploring parameter sensitivity, and modeling realistic worker-driven operation.

5.1 Verification Results

Verification was carried out through 95 automated tests using the pytest framework, organized into three categories: unit tests, integration tests, and simulation tests. Following Sargent's [4] distinction between verification and validation, the testing described in this section addresses verification confirming that the simulation code correctly implements the intended model. Validation confirming that the model adequately represents the real system is addressed separately through the face-validity review described in 5.1.4.

5.1.1 Unit Tests

Seventy-five unit tests verify individual functions across four modules. Table 5.1 summarizes the distribution and purpose of each test group.

Table 5.1: Unit test summary by module.

Module	Tests	Purpose
config.py	10	Verify all PLC-documented values match specification
logic.py	25	Validate speed range (8), sensor detection (6), auto-pass timer (5), mission rules (6)
components.py	27	Verify station configuration (14), motor behavior (8), pallet initialization (5)
state.py	13	Verify SystemState creation, component independence, initial conditions
Total	75	

The logic tests include boundary condition testing for example, verifying that speed values of 19% and 101% are rejected while 20% and 100% are accepted. The sensor detection tests confirm that a pallet at distance 0.1 meters triggers the sensor (boundary) while a pallet at 0.3 meters does not.

5.1.2 Integration and Simulation Tests

Ten integration tests verify that components work together correctly. These tests run short simulations (100–300 steps) and check composite behaviors: a pallet arriving at a station and triggering the sensor, the timer counting up and triggering autopass, the pallet being released and resuming movement, and the CSV log file being created with correct entries.
Ten simulation tests verify the complete `run_simulation()` function, confirming that a short simulation completes without error, returns a valid state object, produces a CSV file, and logs the expected event types. The 26 scenario configurations described later in this chapter were each run as complete simulations, constituting a form of scenario-based virtual testing as described by Dahmen et al. [21].

5.1.3 Test Coverage

The pytest-cov plugin was used to measure code coverage. All six simulation modules achieved 100% statement coverage, as shown in Table 5.2. This level of coverage aligns with the NASA Systems Engineering Handbook's recommendation for unit-test-based verification of simulation software, where statement coverage provides a minimum confidence level that all code paths have been exercised.

Table 5.2: Test coverage by module.

Module	Statements	Missed	Coverage
config.py	25	0	100%
components.py	41	0	100%
logic.py	21	0	100%
state.py	10	0	100%
simulator.py	68	0	100%
logger.py	15	0	100%
Total	180	0	100%

5.1.4 Face-Validity Review

Following Sargent's [4] face-validity procedure, a structured review session was conducted with the system architect (Sven Ekered), who examined the simulation outputs and assessed whether they were consistent with his operational knowledge of the real system. The architect confirmed that the balanced station distribution, the throughput levels, and the general pallet-flow behavior were plausible. He also confirmed specific parameter values processing times, shift structure, break schedule that were subsequently used to calibrate the Stage 4 worker behavior model. This

review constitutes the primary validation evidence for the simulation; full operational validation against live PLC event traces was not performed and is identified as future work.

5.2 Baseline Simulation 24-Hour Run

A 24-hour simulation was run with the default configuration: 5 pallets, 80% belt speed (0.144 m/s), and a 5-second auto-pass timer. The simulation executed 864,000 time steps and produced 55,686 logged events.
The choice of a 24-hour simulation horizon was deliberate and supported by three considerations. First, 24 hours corresponds to a meaningful operational period in industrial practice one full daily cycle making the results directly interpretable for production planning. Second, this duration is long enough to confirm steady state behavior: shorter runs risk capturing only warm up dynamics, whereas a full day allows the simulation to settle and demonstrates whether any drift develops over time. Drift is a known failure mode of discrete-time simulations when timers, counters, or floating-point accumulators are not implemented correctly, and 864,000 successive time steps provide a strong empirical check against such errors. Third, 24 hours fits within practical computational limits each run completes in approximately four minutes on a standard laptop which makes the 26 scenario sensitivity sweep tractable. Longer simulations were considered, but were judged to add limited insight relative to the additional compute time. The 24-hour horizon therefore balances industrial relevance, statistical stability, and computational efficiency.

5.2.1 Event Summary

The simulation logged 27,844 pallet arrivals and 27,840 auto-pass events. The difference of 4 corresponds to four pallets that were stopped at stations when the simulation clock reached exactly 86,400 seconds. Two additional events (simulation start and end) account for the total of 55,686 records. The event distribution is split almost exactly 50/50 between arrivals and auto-passes, which is expected since every arrival in Stage 1 is followed by an auto-pass.

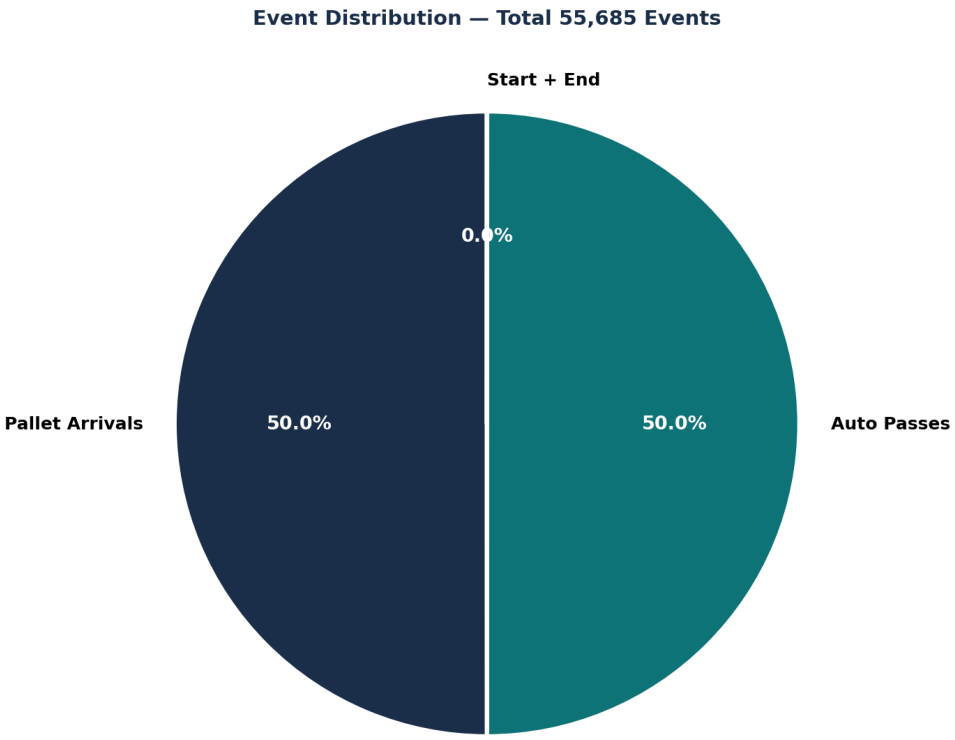


Figure 5.1: Event distribution for the 24-hour baseline simulation. Pallet arrivals and auto-pass events each account for approximately 50% of all 55,686 logged events.

5.2.2 Station Balance

Table 5.3 shows the arrival count per station. In the vocabulary of Subramaniyan and Skoogh [13], the station utilization of 26.86% represents the active period of each station the fraction of simulation time during which the station is occupied by a pallet.

Table 5.3: Pallet arrivals per station 24-hour baseline simulation.

Station	Arrivals	Utilization (%)
DS1	4,641	26.86
DS2	4,641	26.86
DS3	4,641	26.86
DS4	4,641	26.86
DS5	4,640	26.85
DS6	4,640	26.85
Total	27,844	—

The maximum difference between any two stations is 1 arrival. This near-perfect balance confirms that the conveyor distributes pallets evenly across all stations in the baseline configuration.

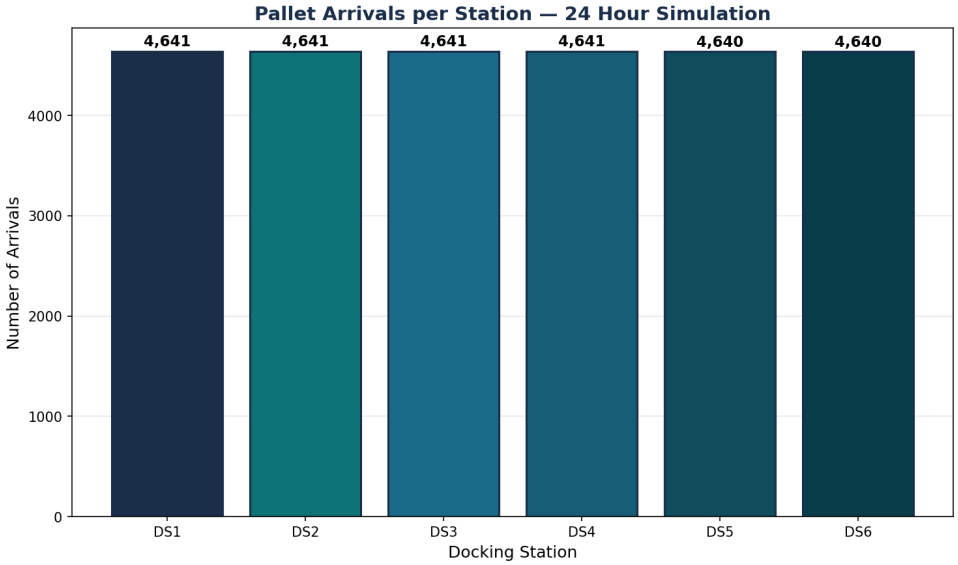


Figure 5.2: Pallet arrivals per docking station over 24 hours. All six stations received between 4,640 and 4,641 arrivals, confirming balanced distribution.

5.2.3 Throughput Stability

Hourly throughput remained between 1,159 and 1,163 arrivals per hour across all 24 hours, with an average of 1,160 arrivals per hour. This stability is noteworthy: it indicates that the simulation reaches a steady state almost immediately after the first few cycles and does not exhibit the drift behavior that can occur in poorly configured discrete-time simulations.

The flat appearance of the throughput curve is itself an important result, not merely a consequence of the chosen plot scale. Three factors explain why the curve is essentially a straight line.

1. the system is deterministic. Every parameter belt speed, auto-pass timer, station configuration, pallet count is fixed for the entire run, and no stochastic elements were included in the baseline configuration. With identical conditions at every moment, the system produces identical output rates in every hour, which manifests itself as a flat line.

2. the system reaches steady state very early. Because the five pallets are evenly initialized around the loop, there is no extended warm-up period during which pallets need to disperse from a starting cluster. The system enters its equilibrium operating regime within the first cycle (under two minutes of simulated time), and remains there for the remaining 24 hours.

3. the absence of variation is a verification result. Discrete time simulations are known to develop slow drift when floating-point accumulation, off by one timer logic, or counter rounding errors are present. Across 864,000 successive time steps, the maximum deviation in hourly throughput is only 4 pallets out of 1,163 less than 0.4the implementation is numerically sound and free of accumulating errors. In this sense, the straight line is not a limitation of the visualization but a direct demonstration that the simulation behaves consistently over long horizons.

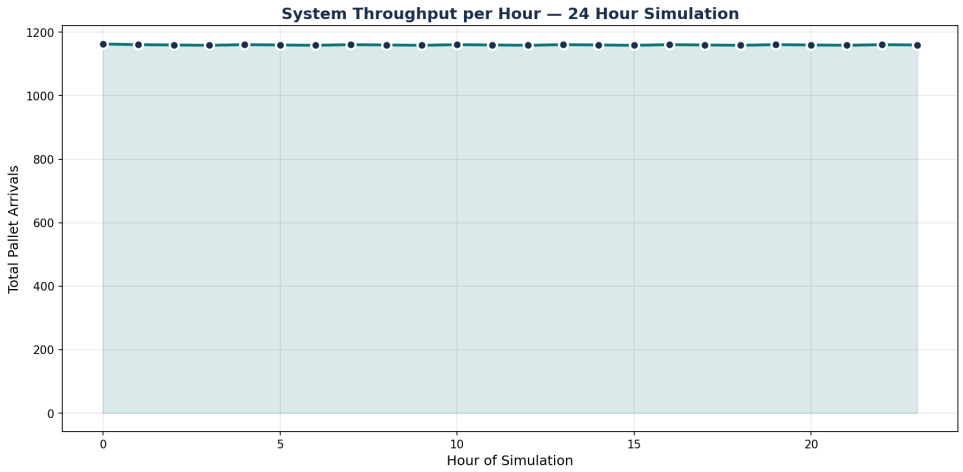


Figure 5.3: System throughput per hour over the 24-hour baseline simulation. Hourly arrivals remain stable between 1,159 and 1,163 throughout the run.

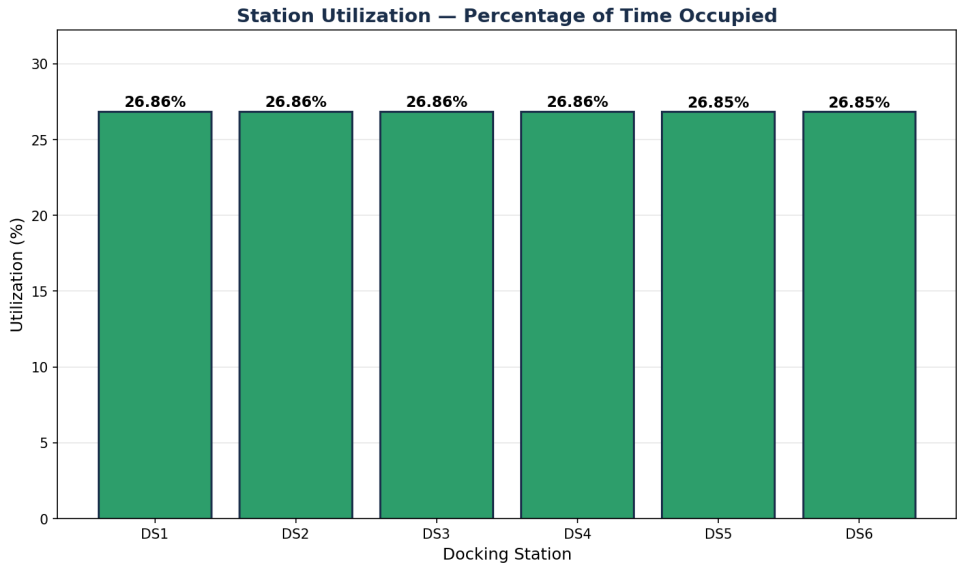


Figure 5.4: Station utilization expressed as the active-period fraction of total simulation time. All six stations show approximately 26.86% active period.

5.2.4 Cycle Time

The calculated pallet cycle time the time for one pallet to complete a full loop - is 92.5 seconds. This consists of 62.5 seconds of travel time (9.0 meters at 0.144 m/s) and 30 seconds of cumulative waiting time (5 seconds at each of 6 stations). This decomposition is important for understanding the parameter-sensitivity results that follow: the 30-second waiting component represents 32% of the cycle time, and it is this component not the 62.5-second transit component that dominates throughput behavior.

5.3 Parameter Sensitivity Scenario Analysis

Following the supervisor's recommendation to vary system parameters and observe the effects, three sets of scenarios were conducted. Each scenario ran for 1 hour of simulated time. The results are interpreted through the lens of active-period bottleneck analysis [13]: identifying which system component most constrains throughput under each configuration.

5.3.1 Pallet Count Variation

Table 5.4 shows the effect of varying the number of pallets from 3 to 8 while holding speed and timer constant.

Table 5.4: Throughput as a function of pallet count (1-hour simulation).

Pallets	Arrivals per Hour
3	699
4	931
5	1,163
6	1,395
7	1,627
8	1,859

The relationship is linear: each additional pallet adds approximately 232 arrivals per hour. This linearity holds because in the baseline configuration there is no contention between pallets the auto-pass timer dominates the cycle time, and pallets do not queue behind each other.

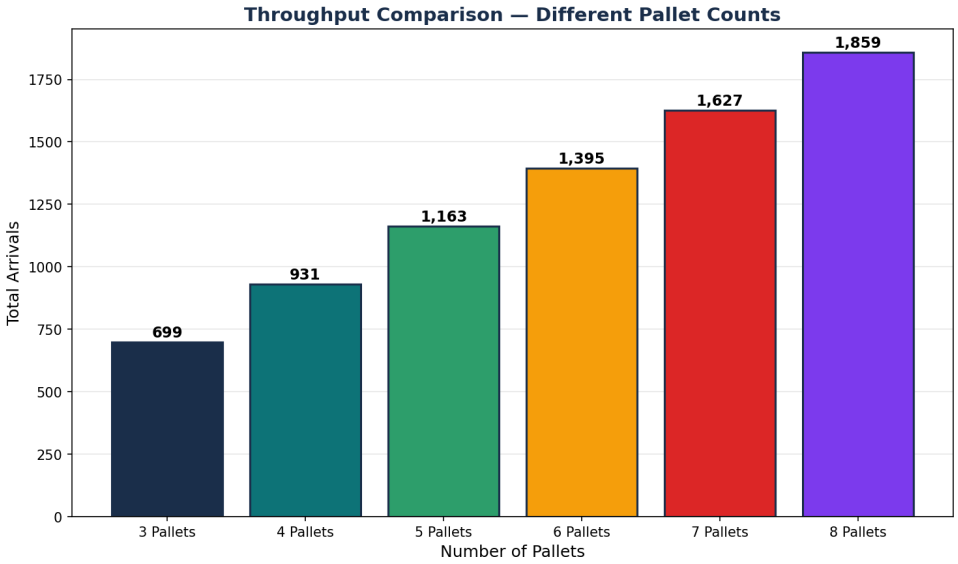


Figure 5.5: Throughput comparison for different pallet counts (1-hour simulation). Each additional pallet adds approximately 232 arrivals per hour.

5.3.2 Belt Speed Variation

Varying belt speed from 40% to 100% produced no change in throughput all scenarios yielded 1,163 arrivals per hour. This result, while initially counterintuitive, is explained by the cycle-time decomposition: at 80% speed, a pallet takes approximately 10.4 seconds to travel between adjacent stations (1.5 meters at 0.144 m/s), but then waits 5 seconds at each station. The total cycle of 92.5 seconds consists of 62.5 seconds of transit and 30 seconds of cumulative station dwell. Increasing belt speed from 80% to 100% reduces the transit component from 62.5 to 50 seconds but leaves the 30-second dwell component unchanged a net reduction of only 13% in cycle time. Moreover, because the transit segments are distributed across 6 inter-station gaps, each individual gap shrinks by only about 2 seconds, which is insufficient to change the number of station visits per hour.
This finding has practical implications: investing in faster belt speeds would yield minimal production benefit if station dwell times are not simultaneously reduced. In bottleneck-analysis terms [13], the belt is not a constraining resource; the docking stations are.

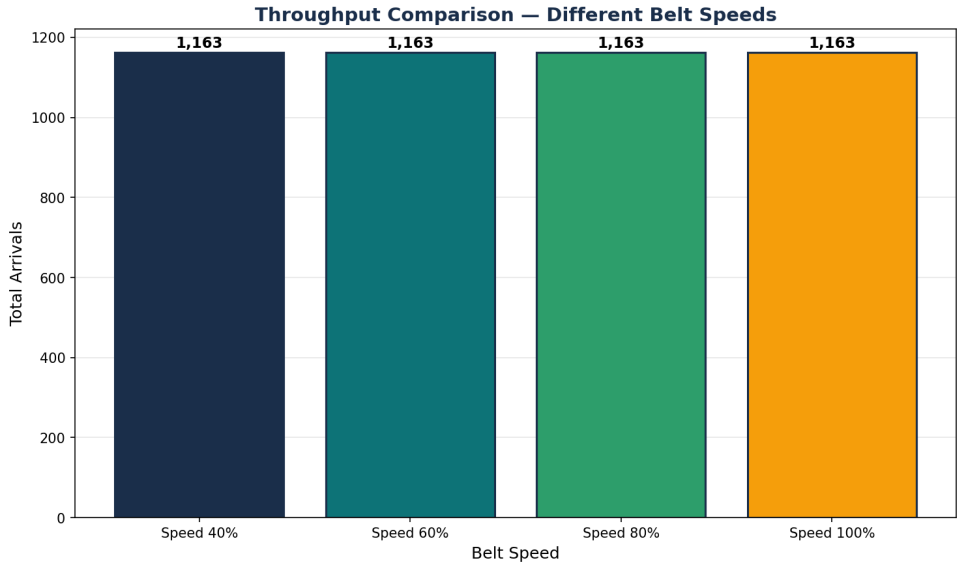


Figure 5.6: Throughput comparison for different belt speeds. All four configurations produce identical throughput of 1,163 arrivals per hour, demonstrating that belt speed is not a constraining resource.

5.3.3 Auto-Pass Timer Variation

The auto-pass timer showed the strongest effect on throughput among the three parameters tested, confirming the stations as the active-period bottleneck.

Table 5.5: Throughput as a function of auto-pass timer duration (1-hour simulation).

Timer (ms)	Arrivals per Hour
2,000	1,454
3,000	1,346
5,000	1,163
8,000	975
10,000	752

Reducing the timer from 10 seconds to 2 seconds nearly doubles throughput from 752 to 1,454 arrivals per hour. This confirms that station dwell time is the primary throughput bottleneck in the current system configuration.

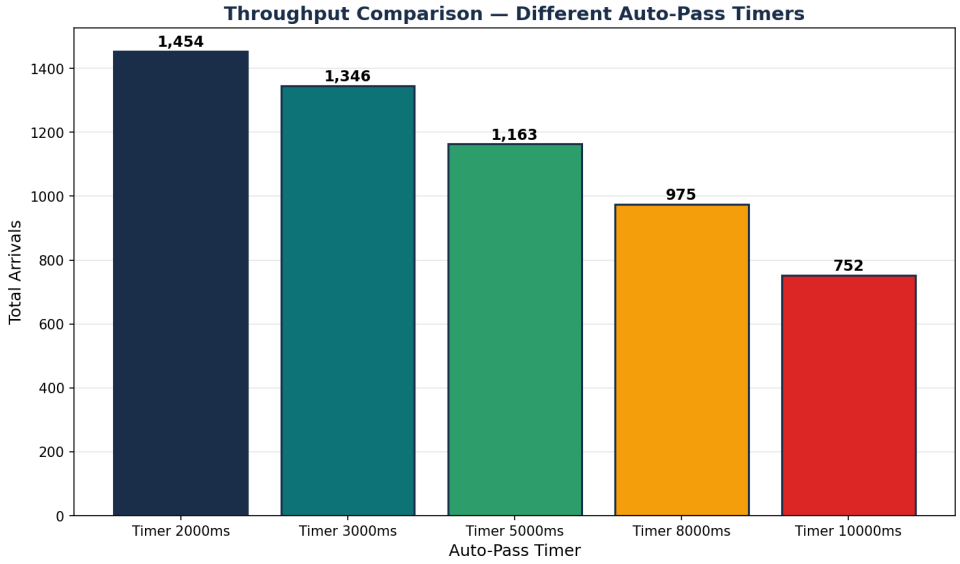


Figure 5.7: Throughput comparison for different auto-pass timer durations. Reducing the timer from 10 seconds to 2 seconds nearly doubles throughput.

5.4 Worker Behavior Simulation

The Stage 4 simulation replaced the fixed auto pass timer with worker driven processing, using timing data from the system architect interview. Each simulation covered a full 8-hour shift from 08:00 to 17:00, including three scheduled breaks totaling 60 minutes.

5.4.1 Baseline Shift Real Configuration

With 5 pallets and the mixed experience workforce reported by the system architect, the simulation produced 773 pallets during one 8 hour shift. Experienced workers at DS1, DS2, DS3, and DS5 each processed approximately 160 pallets, while the

beginner worker at DS4 processed 133 a difference of approximately 17%, consistent with the 20% experience gap reported by the system architect.
The worker at DS4 started as a beginner but transitioned to experienced status after 4 hours, at which point processing times decreased. Despite this transition, the cumulative effect of slower initial performance is visible in the final count. This result should be interpreted as a calibration-based simulation output rather than a measured population effect, since the processing-time parameters are derived from a single informant interview.

5.4.2 Pallet Count with Workers

Table 5.6: Shift production as a function of pallet count (with workers).

Pallets	Processed in Shift
3	531
5	798
8	918
10	945

Unlike the linear relationship observed in the Stage 1 auto pass simulation, the worker-driven scenario shows diminishing returns. Going from 3 to 5 pallets adds 267 units (a 50% increase), but going from 8 to 10 pallets adds only 27 units (a 3% increase). Beyond 8 pallets, the workers become the active-period bottleneck [13] additional pallets circulate on the belt without being processed because all stations are occupied.

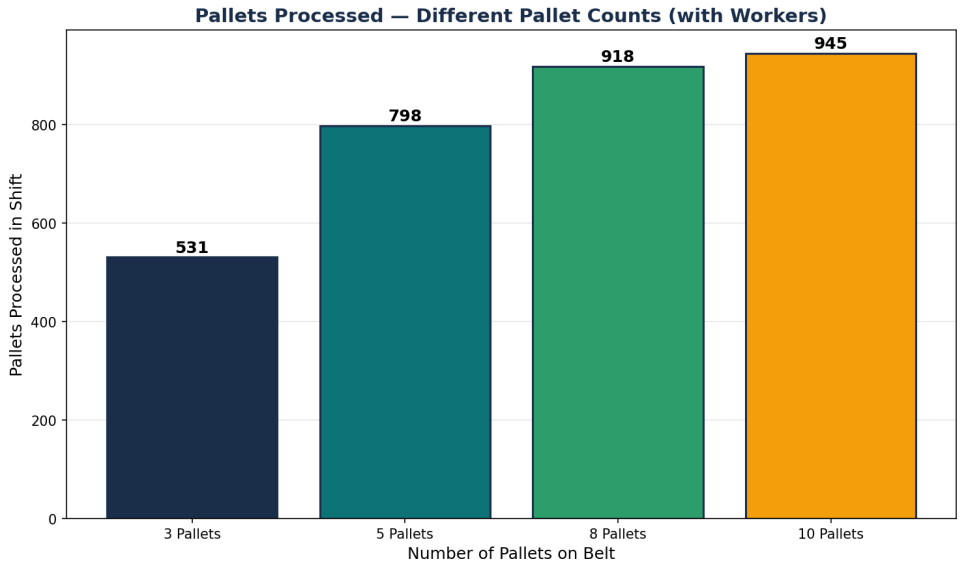


Figure 5.8: Shift production as a function of pallet count with worker behavior enabled. Diminishing returns appear beyond 8 pallets as workers become the active-period bottleneck.

5.4.3 Experience Level Impact

Table 5.7: Shift production by workforce experience composition.

Workforce	Processed in Shift
All Experienced (2–2.5 min)	858
Mixed Real (2–5 min)	794
All Beginners (3–5 min)	543

An all-experienced workforce produces 58% more output than an all-beginner workforce. The real mixed configuration falls 7.5% below the all-experienced scenario, suggesting that the presence of even one beginner measurably reduces overall throughput. This 58% gap is the most actionable result in the worker analysis: it quantifies the production cost of an untrained workforce and provides a basis for evaluating the return on training investment.

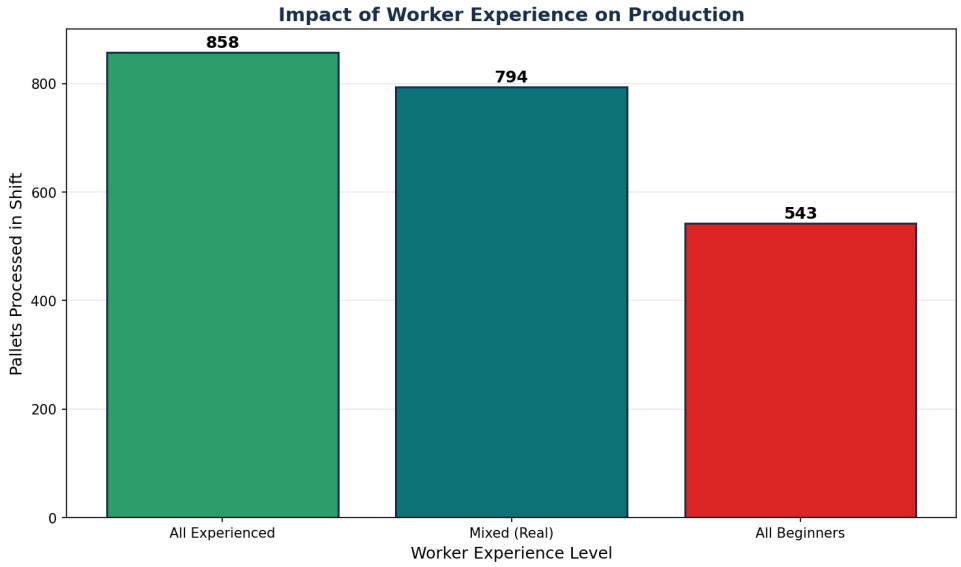


Figure 5.9: Impact of worker experience composition on shift production. An all-experienced workforce produces 58% more output than an all-beginner workforce.

5.4.4 Processing Speed Impact

Table 5.8: Shift production by worker processing speed.

Processing Speed	Processed in Shift
Very Fast (1–2 min)	1,302
Normal (2–3 min)	815
Slow (3–5 min)	549
Very Slow (5–10 min)	324

Worker processing speed has the most pronounced effect on production. Halving the average processing time from the slow range (3-5 minutes) to the very fast range (1–2 minutes) more than doubles output from 549 to 1,302 pallets per shift. This result reinforces the conclusion from the auto-pass timer analysis: station dwell time whether determined by a timer or a worker is the dominant factor governing system throughput. In the active-period bottleneck framework [13], the workers at the docking stations constitute the system's binding constraint, and any improvement effort should be directed at reducing their processing time rather than at increasing belt speed or adding pallets beyond the saturation point.

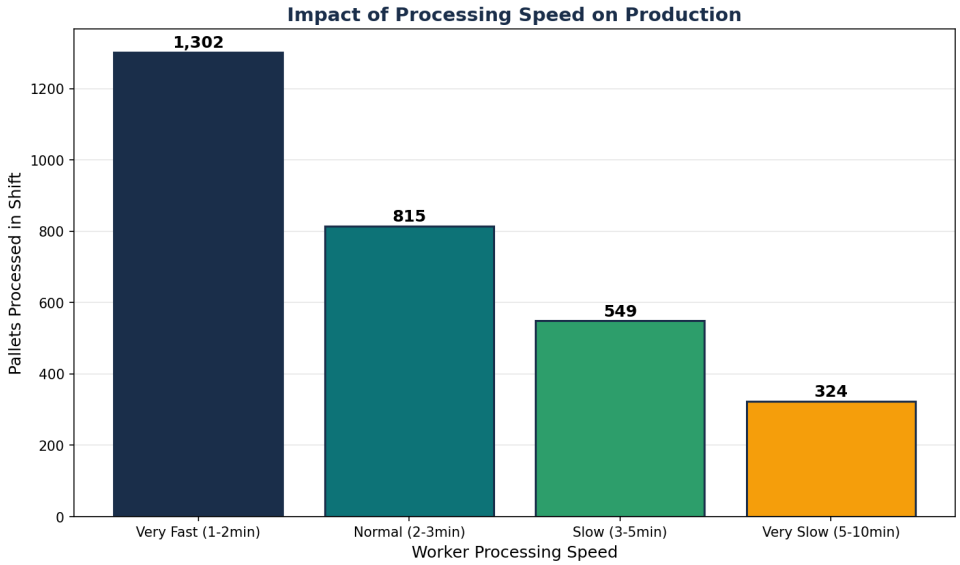


Figure 5.10: Impact of worker processing speed on shift production. Halving processing time more than doubles output, confirming that worker speed is the dominant throughput factor.

6
Discussion

This chapter reflects on the results presented in Chapter 5, interprets the findings in relation to the research questions, and addresses the limitations of the work. It also situates the digital twin within the broader framework of digital model maturity and discusses opportunities for future extension.

6.1 Addressing the Research Questions

6.1.1 RQ1: Simulation Accuracy

The first research question asked how accurately the WS Conveyor behavior can be replicated using a discrete-time simulation based on PLC documentation. The answer requires careful distinction between verification and validation, following the framework established by Sargent [4].

The simulation is verified against the PLC documentation: all 95 automated tests pass, 100% statement coverage confirms that every line of simulation code has been exercised, and every numerical parameter is traceable to a specific page in the 39-page PLC reference (Table 3.1). This verification evidence is strong and demonstrates that the code faithfully implements the intended model.

Validation confirming that the model represents the real system rests on two pieces of evidence. First, every configurable parameter was compared against the PLC documentation, with discrepancies resolved through on-site measurement (belt length corrected from 9.0 to 9.4 meters) or through the system architect interview (processing times calibrated to 2–5 minutes). Second, face validity [4] was established through a structured review session with the system architect, who confirmed the simulation outputs as consistent with his operational experience.

Full operational validation comparing simulated event traces against real PLC log data in real time was not performed, as no automatic data connection exists between the physical system and the simulation. This remains the most significant open validation question and is the first priority for future work.

6.1.2 RQ2: Parameter Sensitivity

The second research question concerned the effect of system parameters on throughput. The scenario analysis, interpreted through the active-period bottleneck framework of Subramaniyan and Skoogh [13], revealed a clear hierarchy of influence.

Station dwell time whether governed by the auto-pass timer or by worker processing speed is the active-period bottleneck. Reducing the auto-pass timer from 10 seconds

to 2 seconds nearly doubled the hourly arrival rate from 752 to 1,454. Similarly, worker processing speed dominated the Stage 4 results, with very fast workers (1–2 minutes) producing four times the output of very slow workers (5–10 minutes). In Subramaniyan and Skoogh's terminology, the docking stations' active period is the binding constraint on system throughput.
Pallet count has a linear effect in the absence of worker constraints. Each additional pallet adds roughly 232 arrivals per hour in the auto-pass configuration. However, when workers are present, this relationship saturates beyond 8 pallets, workers cannot process pallets quickly enough, and additional pallets circulate without being handled. This saturation point marks the transition from a pallet-limited regime to a worker-limited regime.
Belt speed has negligible effect on throughput. The cycle-time decomposition (62.5 seconds transit + 30 seconds cumulative dwell = 92.5 seconds) explains why: even at maximum speed, the transit component shrinks by only 12.5 seconds, while the 30-second dwell component which dominates remains unchanged. The belt is not a constraining resource in this system.

6.1.3 RQ3: Worker Impact

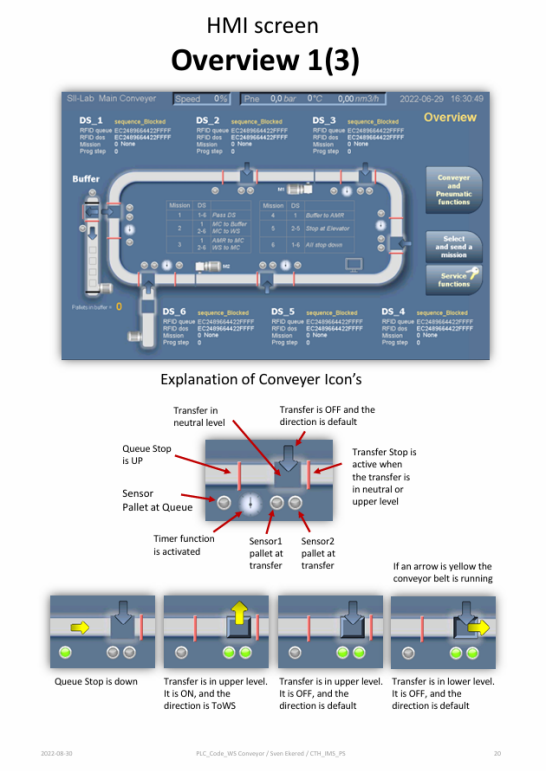
The third research question focused on the impact of worker performance on production output. The results show that worker speed is the single most influential factor in determining shift output. An all-experienced workforce produces 858 pallets per shift, compared to 543 for an all-beginner workforce a 58% difference. This finding is the most directly actionable result in the thesis: it quantifies the production cost of workforce inexperience and provides a basis for evaluating training investment.
The system architect's observation that workers become proficient after approximately half a day is reflected in the simulation through the experience transition mechanism. Worker 4, who starts as a beginner, processes 17% fewer pallets than experienced colleagues over the full shift despite transitioning to experienced status after 4 hours. This persistent gap is explained by the slower initial performance during the first half of the shift. It should be noted that this result is derived from a calibration-based simulation using parameters from a single-informant interview, and should therefore be interpreted as an indication of the effect's direction and approximate magnitude rather than a precise measurement of worker performance differences.
The break schedule reduces productive time by 12.5% (60 minutes of breaks in an 8-hour shift), but this is a fixed overhead that affects all scenarios equally. The system architect confirmed that belt stoppage during breaks is standard practice, consistent with arrangements at larger manufacturing facilities.

6.2 Digital Twin Maturity

As noted in Chapter 2, the classification by Kritzinger et al. [2] distinguishes three levels of integration: digital model, digital shadow, and digital twin. The system developed in this thesis is a digital model data was transferred manually from the

PLC documentation and from the system architect, and no automatic data flow exists between the physical conveyor and the simulation.
Mapping the current implementation onto the three-element architecture proposed by Subramaniyan and Skoogh [14] clarifies what has been achieved and what remains. Their framework identifies three elements: (1) real-time data acquisition from the physical system, (2) a continuously updated simulation model, and (3) a decision-support system. The present thesis covers element (2) the simulation model, verified against documentation and validated through expert review. Elements (1) and (3) constitute the path toward a digital shadow and a full digital twin, respectively.
Moving toward a digital shadow would require establishing a one-way data connection from the PLC to the simulation, enabling the model to reflect real-time sensor states and pallet positions. A full digital twin would additionally require the simulation to send control recommendations back to the PLC for example, suggesting optimal mission assignments or identifying when to redistribute workers between stations.

(a) Photograph of the HMI touchscreen at the SII-Lab.



(b) Reference HMI screen layout. Reproduced from Ekered [3].

Figure 6.1: The HMI operator interface of the WS Conveyor system. The photograph (left) shows the physical iX 2.40 SP5 touchscreen installed at the SII-Lab, and the reference layout (right) shows the operator view as documented in the PLC reference. A digital shadow extension would automate the data feed from this interface into the simulation, removing the manual data transfer that currently characterises the work as a digital model.

The modular software architecture developed in this thesis supports this progression. The separation between the state module (which tracks system status) and the simulation module (which advances time) means that the state could be populated from PLC sensor data rather than from calculated positions, without modifying the analysis or visualization layers.

6.3 Findings from the System Architect

The semi-structured interview with the system architect yielded several findings that would not have been obtainable from the PLC documentation alone. The two-sensor configuration per station, the queue behavior between stations, the 60-second travel time expectation, and the inability to distinguish queued from stuck pallets were all confirmed through this interaction.

These findings served two purposes. First, they informed the simulation design for example, the processing time distribution was calibrated to the 2–5 minute range reported by the system architect rather than the initially assumed 2–50 minute range from early supervisor discussions. Second, they provided face-validity evidence [4] for the simulation, as the system architect reviewed the output charts and confirmed that the balanced station distribution and throughput levels were consistent with his experience of the real system.

The interview method semi-structured rather than a formal questionnaire was chosen on the supervisor's recommendation given that there was only one informant. This approach allowed follow-up questions and yielded richer detail than a fixed-format instrument would have. The interaction with the system architect as the primary stakeholder was a key methodological element in ensuring the simulation's fidelity to the real system.

6.4 Limitations

Several limitations affect the generalizability and accuracy of the simulation, and acknowledging them honestly is part of situating the work within its proper scope.

The simulation models pallets as non-interacting entities. In the real system, pallets occupy physical space on the belt, and a slower pallet can block a faster one. The current model does not implement physical collision or queuing on the belt itself a pallet either stops at a station or continues freely.

The processing time distribution is modeled as uniform between minimum and maximum bounds. Real worker performance may follow a different distribution, potentially skewed or bimodal depending on task complexity and worker familiarity with specific product variants.

The simulation does not model the elevator mechanism, the pneumatic subsystem, or the RFID identification system. These were deliberately excluded to maintain focus on the core conveyor and worker dynamics, but their omission means that the simulation cannot capture faults or delays originating in these subsystems.

Worker fatigue is not explicitly modeled. The system architect suggested that processing times might increase toward the end of a shift, but this has not been em-

pirically measured and was therefore not incorporated into the current model. The ergonomics and operations-research literature addresses fatigue curves in shift work, but engaging with that body of work was beyond the scope of this thesis.
The thesis does not engage with the formal-verification literature for IEC 61131-3 programs, such as model checking of SFC or Ladder logic. The PLC reference document was treated as a specification, and verification was carried out at the Python simulation level rather than at the PLC program level. This is a legitimate scope decision but means that errors in the PLC program itself would propagate into the simulation.
The event logs produced by the simulation are analyzed using descriptive statistics and visualization. Process-mining methods, as described by Van der Aalst and others, would provide a more systematic approach to extracting behavioral patterns from the event data but were not applied in this work.
Finally, the simulation has not been validated against real-time production data. The system architect confirmed that the outputs are plausible through face validity [4], but quantitative comparison with actual PLC event logs would provide stronger evidence of fidelity. This is the most important open validation question.

6.5 Practical Implications

The scenario analysis suggests several practical considerations for operating the WS Conveyor.
Investing in faster belt speeds is unlikely to yield significant throughput improvements. The dominant factor is station dwell time, and any operational improvement effort would be more productively directed toward reducing worker processing times through training, tool design, or task simplification.
The system operates well below capacity in the current configuration. With station utilization at approximately 27% and 73% idle time, there is substantial room for increased throughput if more pallets are added or if workers process pallets more quickly. However, the saturation effect observed beyond 8 pallets indicates that worker capacity, not belt capacity, is the binding constraint.
The training investment for new workers is modest. The system architect reported that half a day of practice is sufficient to reach experienced performance levels, with only a 20% speed difference between beginners and experienced workers. This suggests that workforce turnover has a limited impact on production if incoming workers receive even basic training before their first shift.

7
Conclusion and Future Work

7.1 Conclusion

This thesis developed a digital model of the WS Conveyor system at the Chalmers SII Lab, progressing from PLC documentation analysis through implementation, verification, scenario analysis, and worker behavior modeling.
The simulation replicates the sensor based control logic of the real system using a discrete time approach with 0.1 second resolution, chosen to mirror the PLC's fixed scan cycle execution model. All system parameters were traced to specific pages in the PLC documentation (Table 3.1), and physical measurements were confirmed through on site visits. The implementation was verified through 95 automated tests achieving 100% statement coverage. Face validity [4] was established through a structured review session with the system architect, who confirmed the simulation outputs as consistent with his operational experience. The simulation is verified against documentation; full operational validation against live PLC event traces remains future work.
The scenario analysis, interpreted through the active period bottleneck framework of Subramaniyan and Skoogh [13], demonstrated that station dwell time whether governed by an auto pass timer or by worker processing speed is the dominant factor determining system throughput. Belt speed has negligible impact due to the relative brevity of travel time compared to station waiting time. Pallet count has a linear effect in the absence of worker constraints but saturates when workers become the bottleneck, with diminishing returns observed beyond 8 pallets.
The worker behavior simulation, calibrated with empirically collected data from the system architect, showed that an 8 hour shift with five workers and three scheduled breaks produces approximately 773 to 798 pallets under real operating conditions. Worker processing speed has the strongest effect on production output, with very fast workers producing four times the output of very slow workers. An all experienced workforce produces 58% more output than an all beginner workforce. The experience gap between beginners and experienced workers is approximately 20%, and new workers reach proficiency within half a day.
The system currently operates as a digital model with manual data transfer, covering element (2) of the three element digital twin architecture described by Subramaniyan and Skoogh [14]. The modular software architecture supports future extension toward a digital shadow or full digital twin through the addition of automatic PLC data feeds.

7.2 Future Work

Several directions for future work emerge from this thesis, organized along the ISA 95 automation pyramid levels from shop floor integration through enterprise level decision support [11].

7.2.1 Automatic Data Connection

The most direct extension would be establishing a data connection between the real PLC and the simulation. Following the ISA 95 hierarchy, this begins at Level 1 (shop floor sensor data) and progresses through Level 2 (supervisory control). A one way connection (digital shadow) would allow the simulation to mirror real time sensor states and pallet positions. A bidirectional connection (digital twin) would additionally enable the simulation to suggest operational adjustments for example, recommending mission assignments or identifying optimal pallet counts based on current worker performance. Lundius [19] discusses MES as the bridge between shop floor and enterprise systems, providing a concrete architectural reference for this integration.

7.2.2 Camera Based Pallet Tracking

The system architect confirmed that the current sensor only logic cannot distinguish between a pallet waiting in a queue and a pallet stuck between stations. A camera system could resolve this ambiguity by providing continuous position tracking of all pallets. The digital twin could integrate camera data as an additional input source, improving the accuracy of the simulation state and enabling real time anomaly detection.

7.2.3 Predictive Maintenance and Process Mining

The event logs generated by the simulation and potentially by a live connected digital twin represent structured time series data that could be used to train machine learning models for predictive maintenance. Patterns in pallet travel times, sensor activation delays, or worker processing time trends could signal emerging equipment faults or performance degradation before they cause production stoppages.
The CSV event logs are structurally equivalent to PLC event relationship tables as described by Laftchiev and Sun [20]. Applying process mining methods to these logs either from the simulation or from a live PLC connection is a natural next step that would enable systematic behavioral pattern extraction beyond the descriptive statistics used in this thesis.

7.2.4 Queue and Collision Modeling

The current simulation treats pallets as non interacting entities. Implementing physical pallet interaction including queuing on the belt, blocked station rerouting, and collision avoidance would improve the fidelity of the simulation, particularly at higher pallet counts where contention effects become significant.

7.2.5 Extended Worker Modeling

The worker model could be extended to include fatigue effects (increasing processing times toward the end of a shift), individual skill profiles (rather than binary beginner/experienced classification), and task specific processing times that vary by station and product type. Multi shift simulation with handover dynamics would also add realism for production planning applications.

7.2.6 SCADA Integration

The system architect confirmed that mission assignment is handled by the SCADA system. Integrating SCADA logic into the simulation or connecting the simulation to the real SCADA planner would enable more realistic pallet routing and mission scheduling, moving beyond the uniform auto pass behavior currently implemented. In the ISA 95 framework [11], this represents the progression from Level 2 (supervisory control) to Level 3 (manufacturing operations management), positioning the digital twin as a component of a broader manufacturing execution system.

Bibliography

[1] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," White Paper, Florida Institute of Technology, 2014.

[2] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[3] S. Ekered, "Program code, WS Conveyor SII-Lab," Internal documentation, Chalmers University of Technology, Department of Industrial and Materials Science, SII-Lab, Gothenburg, Sweden, 30 August 2022. 39 pages.

[4] R. G. Sargent, "Verification and validation of simulation models," *Journal of Simulation*, vol. 7, no. 1, pp. 12–24, 2013.

[5] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0," Final Report of the Industrie 4.0 Working Group, acatech – National Academy of Science and Engineering, Frankfurt, Germany, 2013.

[6] CODESYS Group, "CODESYS – the IEC 61131-3 automation software," [Online]. Available: https://www.codesys.com. [Accessed: May 2026].

[7] S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests*. Boston, MA, USA: Addison-Wesley, 2009.

[8] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017.

[9] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and industrial applications," *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, 2021.

[10] T. S. Baines, R. Kay, S. Adesola, and M. Higson, "Strategic positioning: An integrated decision process for manufacturers," *International Journal of Operations & Production Management*, vol. 25, no. 2, pp. 180–201, 2005.

[11] G. S. Martinez, S. Sierla, T. Karhela, and V. Vyatkin, "Automatic generation of a simulation-based digital twin of an industrial process plant," *Sensors*, vol. 21, no. 14, p. 4736, 2021.

[12] O. Salunkhe, O. Stensjö, and Å. Fasth-Berglund, "Assembly 4.0: Wheel hub nut assembly using a cobot," *Procedia Manufacturing*, vol. 17, pp. 222–231, 2018.

[13] M. Subramaniyan, A. Skoogh, H. Salomonsson, P. Bangalore, and J. Bokrantz, "A data-driven algorithm to predict throughput bottlenecks in a production system based on active periods of the machines," *Computers & Industrial Engineering*, vol. 125, pp. 533–544, 2018.

[14] M. Subramaniyan, A. Skoogh, M. A. Syed, J. Bokrantz, and B. Johansson, "A generic hierarchical clustering approach for detecting bottlenecks in manufacturing," *Journal of Manufacturing Systems*, vol. 55, pp. 143–158, 2020.

[15] J. Bokrantz, A. Skoogh, C. Berlin, and J. Stahre, "Maintenance in digitalised manufacturing: Delphi-based scenarios for 2030," *International Journal of Production Economics*, vol. 191, pp. 154–169, 2017.

[16] M. R. Lopes, C. Costigliola, R. Pinto, S. Vieira, and J. M. C. Sousa, "Pharmaceutical 4.0 – A review of digital transformation in the pharmaceutical industry," *International Journal of Computer Integrated Manufacturing*, vol. 37, no. 1–2, pp. 7–32, 2024.

[17] J. Lee, E. Lapira, B. Bagheri, and H. Kao, "Recent advances and trends in predictive manufacturing systems in big data environment," *Manufacturing Letters*, vol. 1, no. 1, pp. 38–41, 2013.

[18] F. J. Maseda, I. Martínez de Alegría, and E. Bilbao, "Application of GRAFCET methodology in a programmable logic controller," *Sensors*, vol. 21, no. 8, p. 2762, 2021.

[19] S. Lundius, "Development of a manufacturing execution system for a production testbed," M.S. thesis, Dept. of Industrial and Materials Science, Chalmers University of Technology, Gothenburg, Sweden, 2019.

[20] E. Laftchiev and D. Sun, "Discovering the topology of PLC-based manufacturing systems from event logs," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 408–413, 2018.

[21] U. Dahmen, K. Behr, M. Botta, and J. Rossmann, "Scenario-based virtual testing and digital twin models for automated driving," *Applied Intelligence*, vol. 53, pp. 29755–29774, 2023.

[22] M. Achouch, M. Dimitrova, K. Ziane, S. Sattarpanah Karganroudi, R. Dhouib, H. Ibrahim, and M. Adda, "On predictive maintenance in Industry 4.0: Overview, models, and challenges," *Applied Sciences*, vol. 12, no. 16, p. 8081, 2022.

A

Interview Questions

The following questions were prepared for the semi-structured interview with the system architect at the SII-Lab. The interview was conducted on-site, and subsequently transcribed for analysis. Follow-up questions were asked based on the responses received.

Worker Processing Times

1. How long does a typical worker take to process one pallet at a station?
2. What is the fastest processing time you have observed?
3. What is the slowest processing time you have observed?
4. Are the tasks different at each station (DS1 to DS6)?
5. Does the processing time vary during the day?
6. Are there different types of products that take different processing times?

Shift and Schedule

7. How many workers are on duty during one shift?
8. How long is one shift?
9. What time does the first shift start and end?
10. Is there a second shift?
11. What happens during shift handover? Does the belt stop?
12. How long does the shift handover take?

Breaks and Pauses

13. Do workers take breaks during a shift?
14. How long are the breaks and when do they occur?
15. Does the belt continue running when a worker takes a break?
16. Do all workers take breaks at the same time or separately?

Worker Experience

17. Are there different skill levels among workers?

18. Approximately how much faster is an experienced worker compared to a new worker?
19. How long does it take for a new worker to become experienced?
20. What proportion of a typical shift consists of experienced workers?

Mission Assignment

21. Who decides which mission a pallet receives the worker or the system?
22. What percentage of pallets pass through versus enter the workspace?
23. Does a pallet always visit stations in order from DS1 to DS6?
24. What happens during Mission 5?

Physical Measurements

25. What is the exact length of the conveyor belt?
26. What is the distance between docking stations?
27. How many pallets are normally on the system?
28. Is the belt speed always set to 80%?

Validation

29. Does the PLC store event log files?
30. How many pallets pass through the system in a real 8-hour shift?
31. (Showed simulation charts) Do these results seem realistic?
32. Are there behaviors in the real system that the simulation might be missing?

B

Scenario Results

This appendix provides the complete results from all 26 simulation scenarios conducted during the analysis phase.

Stage 3 Scenarios Parameter Sensitivity (Auto-Pass)

Table B.1: Pallet count scenarios (1-hour simulation, auto-pass mode).

Pallets	Speed (%)	Arrivals per Hour
3	80	699
4	80	931
5	80	1,163
6	80	1,395
7	80	1,627
8	80	1,859

Table B.2: Belt speed scenarios (1-hour simulation, 5 pallets).

Speed (%)	Actual (m/s)	Arrivals per Hour
40	0.072	1,163
60	0.108	1,163
80	0.144	1,163
100	0.180	1,163

Table B.3: Auto-pass timer scenarios (1-hour simulation, 5 pallets, 80% speed).

Timer (ms)	Arrivals per Hour
2,000	1,454
3,000	1,346
5,000	1,163
8,000	975
10,000	752

Stage 4 Scenarios Worker Behavior(8-Hour Shift)

Table B.4: Pallet count scenarios with workers (8-hour shift).

Pallets	Processed in Shift
3	531
5	798
8	918
10	945

Table B.5: Experience level scenarios (8-hour shift, 5 pallets).

Workforce Composition	Processed in Shift
All Experienced (2–2.5 min)	858
Mixed — Real (2–5 min)	794
All Beginners (3–5 min)	543

Table B.6: Processing speed scenarios (8-hour shift, 5 pallets).

Processing Speed	Processed in Shift
Very Fast (1–2 min)	1,302
Normal (2–3 min)	815
Slow (3–5 min)	549
Very Slow (5–10 min)	324

Source Code Repository

The complete simulation source code, test suite, analysis modules, and simulation logs are available in the project GitHub repository:

`https://github.com/Saleenafiroz/ChalmersThesis`

The repository is organized as follows:

Table B.7: Project repository structure.

Path	Contents
`src/mock_up/`	Config, components, logic, state, worker, shift
`src/simulation/`	Simulation engine (Stage 1 and Stage 4)
`src/logging/`	CSV event logger
`src/analysis/`	Data loader, analyzer, visualizer, scenarios
`tests/unit/`	75 unit tests
`tests/integration/`	10 integration tests
`tests/simulation/`	10 simulation tests
`data/logs/`	Simulation event logs (CSV)
`data/results/`	Charts and analysis outputs

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY