DEGREE PROJECT FOR MASTER OF SCIENCE WITH SPECIALIZATION IN ROBOTICS
DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY WEST

Monitoring inventories for kitting solutions in the manufacturing industries

Prajwal Mahesh Palled

UNIVERSITY WEST

A THESIS SUBMITTED TO THE DEPARTMENT OF ENGINEERING SCIENCE

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE WITH SPECIALIZATION IN ROBOTICS

AT UNIVERSITY WEST

2026

Date:	August 31, 2026
Author:	Prajwal Mahesh Palled
Examiner:	Sudha Ramasamy
Advisor:	Shiva Tutunchi
Programme:	Master Program in Robotics and Automation
Main field of study:	Automation with a specialization in industrial robotics
Credits:	120 Higher Education credits (see the course syllabus)
Keywords:	Inventory Monitoring, Pick-to-light, Sensor fusion, Stock verification, Natural-language interaction, Multi-bin kitting.
Template:	University West, IV-Master 2.10
Publisher:	University West, Department of Engineering Science S-461 86 Trollhättan, SWEDEN Phone: + 46 520 22 30 00 Fax: + 46 520 22 32 99 Web: www.hv.se

Summary

Conventional pick-to-light and manual kitting systems guide operators to the correct part location but generally do not independently verify that the correct quantity has been picked. This thesis investigates how intelligent real-time inventory monitoring can improve inventory visibility and automate stock verification in manufacturing kitting operations through three progressively developed prototypes.

The first prototype combined weight and vision sensing in a laboratory setting. Although sensor fusion did not improve raw counting accuracy over individual sensing modalities under the tested conditions, it provided higher confidence in the reported counts and enabled hidden-part inference, correctly determining the number of parts hidden from the camera in 95.7% of tested frames. The system also detected and recovered from complete sensor failure without reporting a misleading count.

The second prototype integrated a weight sensor into an existing industrial pick-to-light installation at Binar Solutions AB and added natural-language operator interaction. All 16 baseline picks were independently verified as correctly tracked. The system also correctly detected intentional over-picks, while the order-queue path provided a defined recovery workflow for resolving such discrepancies. AI-assisted kit requests, order-queue shortage prediction, and bilingual English/Swedish operation were correctly handled across all tested trials, although AI-assisted interaction showed variability depending on the language-model access available during testing.

The third prototype extended the approach to a coordinated three-bin kitting station in simulation. Pick-tracking accuracy remained 100% across 29 combined trials, while multi-bin parsing, over-pick correction, priority-order resumption, and demand-planning shortage projection were correctly demonstrated.

Overall, the results show that intelligent inventory monitoring can combine verified sensing with automated discrepancy handling and, where appropriate, natural-language interaction to improve inventory visibility and stock verification across different development settings. The conclusions are bounded by the relatively small number of trials, the single industrial deployment site, and the simulation-based validation of the multi-bin prototype; the results therefore demonstrate correct behaviour under the tested conditions rather than generalized industrial-scale reliability.

Preface

The work was carried out with support from Vinnova (Sweden's Innovation Agency) within the Advanced Digitalization program under the project named AIKIDO (Dnr: 2025-01009). I would like to thank Binar Solutions AB for providing access to their pick-to-light installation and enabling the industrial phase carried out in this thesis. I would especially like to thank Jimmy Kleibrant for their support and assistance during the project.
I would also like to thank Examiner Sudha Ramasamy and Supervisor Shiva Tutunchi for their guidance, feedback and support throughout this thesis.

Affirmation

This master degree report, *Monitoring inventories for kitting solutions in the manufacturing industries*, was written as part of the master degree work needed to obtain a Master of Science with specialization in Robotics and Automation degree at University West. All material in this report, that is not my own, is clearly identified and used in an appropriate and correct way. The main part of the work included in this degree project has not previously been published or used for obtaining another degree.

Prajwal Mahesh Palled	31/08/2026
Signature by the author	Date

Contents

Preface

SUMMARY ..... **III**

PREFACE ..... **IV**

AFFIRMATION ..... **V**

CONTENTS ..... **VI**

SYMBOLS AND GLOSSARY ..... **IX**

Main Chapters

1 INTRODUCTION **1**

1.1 Purpose of the Study 3
1.2 Aim 3
1.3 Research questions 4
1.4 Limitations 4

2 RELATED WORK (BACKGROUND) **5**

2.1 Inventory Management Challenges 5
2.2 IoT-Based Inventory Monitoring Systems 5
2.3 Sensor Technologies for Inventory Monitoring 6
2.4 Computer Vision and Intelligent Monitoring Systems 7
2.5 Kitting Systems in Manufacturing 7
2.6 Hybrid and Intelligent Inventory Management Approaches 8
2.7 Multi-Sensor Fusion for Inventory Monitoring 8
2.8 Pick-to-Light and Operator-Guided Picking Systems 9
2.9 Large Language Model Agents for Industrial Decision Support 10
2.10 Research Gap 10

3 METHOD **12**

3.1 Methodological Approach 12
3.2 System Architecture 12
3.3 Phase 1: Sensor Fusion Prototype 14
3.3.1 Prototype Objective 14
3.3.2 Hardware Architecture 14
3.3.3 Sensor Fusion Methodology 15
3.3.4 Software Architecture 15
3.3.5 Experimental Configuration and Evaluation Metrics 15
3.4 Phase 2: Industrial Pilot - AI-Assisted Automated Pick-to-Light System 16
3.4.1 Prototype Objective 16
3.4.2 Hardware Architecture 16
3.4.3 System Methodology 17
3.4.4 Software Architecture 18
3.4.5 Experimental Configuration and Evaluation Metrics 18
3.5 Phase 3: Simulation-Based Multi-Bin Scalability Study 18

3.5.1 Prototype Objective *18*
3.5.2 Hardware Architecture *18*
3.5.3 System Methodology *19*
3.5.4 Software Architecture *19*
3.5.5 Experimental Configuration and Evaluation Metrics *19*
3.6 Data Collection and Experimental Procedure 20

4 SYSTEM DEVELOPMENT AND IMPLEMENTATION 21

4.1 Chapter Overview 21
4.2 Phase 1: Sensor Fusion Prototype 21
4.2.1 System Implementation Overview *21*
4.2.2 Weight Channel Implementation *22*
4.2.3 Vision Channel Implementation *23*
4.2.4 Sensor Health Evaluation *25*
4.2.5 Adaptive Sensor Fusion and Operating Modes *25*
4.2.6 Hidden-Part Inference *27*
4.2.7 Confidence and Agreement Metrics *28*
4.2.8 Operator Dashboard and Data Logging *28*
4.2.9 Implementation Challenges *29*
4.3 Phase 2 - Industrial Pilot - AI-Assisted Automated Pick-to-Light System 30
4.3.1 System Implementation Overview *30*
4.3.2 Weight-Based Pick Detection (ESP32 Firmware) *31*
4.3.3 Server-Side Deterministic Safety Layer *32*
4.3.4 AI Kitting Assistant *33*
4.3.5 Independent AI Safety Reviewer *34*
4.3.6 Order Queue and Demand-Planning Projection *35*
4.3.7 Tare and Unit-Weight Calibration *36*
4.3.8 Inventory Accuracy, Predictions, and System Performance Metrics *37*
4.3.9 Operator Dashboard and Bilingual Support *38*
4.3.10 Implementation Challenges *39*
4.4 Phase 3: Multi-Bin Simulation 40
4.4.1 System Implementation Overview *40*
4.4.2 Multi-Bin Weight Sensing and Calibration *41*
4.4.3 Pick Detection and Kit Sequencing *42*
4.4.4 Tamper and Anomaly Detection *44*
4.4.5 MQTT Command Handling and Manual Override Tools *44*
4.4.6 Order Queue and Priority Pick *45*
4.4.7 Inventory Accuracy Tracking *46*
4.4.8 Dashboard Implementation and Bilingual Natural-Language Input *46*
4.4.9 Implementation Challenges *47*
4.5 Chapter Summary 48

5 RESULTS AND DISCUSSION 49

5.1 Overview and Evaluation Approach 49
5.2 Phase 1 Results 49
5.2.1 Overview of Testing Conducted *49*
5.2.2 Count Accuracy: Matched Controlled Trial *49*
5.2.3 Mode Comparison Across a Continuous Session *50*
5.2.4 System Failure and Fusion Recovery *52*
5.2.5 Hidden-Part Inference *52*
5.3 Phase 1 Discussion 53
5.4 Phase 2 Results 55
5.4.1 Overview of Testing Conducted *55*
5.4.2 Pick Tracking Accuracy and Guardrail Behaviour *56*
5.4.3 AI-Assisted Kit Requests: Single- and Multi-Unit *56*

5.4.4 Order Queue Shortage Detection 57
5.4.5 Over-Pick Detection and Recovery 58
5.4.6 Informational Query Handling 59
5.4.7 Bilingual Operation 59
5.5 Phase 2 Discussion 60
5.6 Phase 3 Results 62
5.6.1 Overview of Testing Conducted 62
5.6.2 Multi-Bin Kit Sequencing and Manual Pick Tracking 62
5.6.3 Guardrail and Safety-Event Handling 63
5.6.4 Natural-Language Parsing, Bilingual Operation, and Order-Queue Resume 64
5.6.5 Demand-Planning Shortage Projection 66
5.7 Phase 3 Discussion 67
5.8 Cross-Phase Discussion 69
5.9 Chapter Summary 70
6 CONCLUSION 72
6.1 Future Work and Research 73
6.2 Critical Discussion 73
6.3 Generalization of the result 74
7 REFERENCES 75

Appendices

A. PHYSICAL PROTOTYPE - LABORATORY HARDWARE AND MULTI-BIN SIMULATION
B. PHYSICAL PROTOTYPE - INDUSTRIAL INSTALLATION AT BINAR SOLUTIONS AB

Symbols and glossary

AI	Artificial intelligence used in this thesis to interpret natural-language operator requests and interact with the kitting system.
API	An interface that allows software components to communicate with each other; used to connect the AI assistant with system functions.
BLE	A short-range wireless communication technology discussed as an inventory-tracking approach in the related work.
CSV	A text-based format used to store tabular data, including recorded test and sensor data.
ESP32	The microcontroller used in the prototypes to interface with sensors and communicate with the system.
HX711	A load-cell interface component that amplifies and converts the load-cell signal into digital weight measurements.
IoT	An approach in which physical devices and sensors collect and exchange data over a network; relevant to the inventory-monitoring architecture discussed in this thesis.
JSON	A structured text format used to exchange information between software components.
LED	A light-based indicator used in the pick-to-light system to show the bin or picking location requiring operator attention.
LLM	A language model used by the AI assistant to interpret natural-language requests and determine the appropriate system action.
MQTT	A lightweight messaging protocol used for communication between distributed system components.
OLED	A display technology used for showing status information in the prototype.
PTL	Pick-to-light: a system that uses visual indicators at picking locations to guide an operator to the required part.
RFID	A wireless technology that uses radio-frequency tags to identify and track physical items, discussed in the related work as an inventory-tracking approach.
ROI	A selected area of an image within which visual processing or object detection is performed.
Wokwi	An online electronics simulation platform used to develop and test the multi-bin prototype in Phase 3.
YOLO	An object-detection model architecture that detects and localises objects in an image in a single processing step, discussed in the related work.

1 Introduction

Manufacturing industries are increasingly transitioning towards smart and digitally connected production systems as part of the Industry 4.0 paradigm. Modern assembly environments rely heavily on real-time data visibility, meaning the ability to access accurate and up-to-date operational information about materials, inventory levels, and production status, in order to ensure operational stability, responsiveness, and efficiency [1]. Within such environments, material availability is a critical success factor, as production performance is directly influenced by whether the required components are available at the correct location and time [1], [2]. Recent research highlights that real-time monitoring technologies, including IoT-based tracking systems [1], [3] and edge or cloud-based monitoring platforms [2], [4], significantly improve inventory accuracy and reduce manual intervention.

One widely adopted material supply strategy in assembly environments is kitting. Kitting refers to the process of collecting a specific set of components and delivering them together in a container to support one or more assembly operations [5]. This approach is particularly beneficial in manufacturing environments with high product variety and limited workstation space, where efficient material organisation is required [5]. Compared to traditional line stocking, kitting can reduce congestion at assembly stations and support lean production principles by organising components according to specific assembly tasks [5]. For the purpose of this study, the effectiveness of kitting systems depends on accurate and timely inventory information, such as the current stock level of components in bins, the number of parts remaining, and whether all required parts are available to complete a kit. If even one component is missing from a kit, the entire assembly operation may be delayed.

Inventory inaccuracy is widely recognised as a significant challenge in production and inventory management systems [6]. Kang and Gershwin [6] demonstrate that even a small average stock loss of 1% can result in lost demand exceeding 10-20 times the actual loss due to disruption of automated replenishment systems. Lean manufacturing environments are particularly sensitive to such inaccuracies, since reduced safety stocks and short lead times offer limited protection against inventory record errors. In a kitting environment, where each kit requires a complete set of components, even minor discrepancies in component availability can cause significant operational interruptions.

Traditional inventory control systems in manufacturing are commonly based on manual counting [1], periodic auditing, or barcode scanning [1], [7]. Although these methods are simple and low-cost, they provide only snapshot-based visibility, meaning that stock levels are evaluated only at specific points in time rather than being tracked continuously, which can introduce latency between physical stock movement and system updates [1], [3], [7]. Studies on inventory management systems have shown that, in typical barcode-based operations, items are manually scanned either when operators pick components from storage bins or when material handlers refill bin locations [1], [3]. Because the inventory records depend on these scanning events and may be updated only periodically, the detection of stockouts and inventory discrepancies can be delayed [7]. Furthermore, manual processes increase the risk of transaction errors, misplaced

items, and incorrect identification of components due to human handling and manual data entry [3], [7].

To overcome these limitations, extensive research has explored IoT-based inventory monitoring solutions that enable automated and continuous tracking of stock levels [1], [3], [4]. Various sensing technologies, including RFID, computer vision, ultrasonic sensors, and weight-based sensing, have been investigated for automated inventory monitoring, each offering specific advantages and limitations depending on the application environment [3], [7]. RFID-based systems have demonstrated inventory accuracy improvements of up to 99% by enabling automated identification of items at entry and exit points [1], [4], [7], [8]. Vision-based systems using deep learning models such as YOLO have been implemented for automated stock recognition and counting in warehouse environments, achieving high detection accuracy and reduced processing latency [9], [10]. In addition, ultrasonic and weight-based sensing solutions have been proposed as affordable alternatives for monitoring stock levels on shelves, particularly in small and medium enterprises [11], [12], [13]. Recent studies further suggest that combining multiple sensing technologies through sensor fusion improves robustness and reduces the likelihood of undetected errors [3], [14], [15].

Despite these technological advancements, most existing research primarily focuses on inventory monitoring in warehouse management, retail environments, or large-scale logistics systems [1], [3]. These environments typically involve centralised storage areas where inventory is stored in bulk and accessed periodically for distribution or order fulfilment. In contrast, manufacturing kitting environments operate at the workstation level, where components are prepared and supplied directly to support specific assembly tasks. As a result, the operational requirements for inventory monitoring in kitting systems differ significantly from those in warehouse environments [5].

Kitting stations involve frequent handling of diverse components of varying sizes, limited local storage space, and direct interaction between material availability and assembly operations as detailed by Brynzér and Johansson [5]. Unlike warehouse systems, where buffer stock may reduce the immediate impact of inventory discrepancies [6], these workstation-level environments depend on the availability of every component required to complete a kit. Therefore, even small inaccuracies in component availability can immediately affect production continuity. Consequently, inventory monitoring systems intended for kitting applications must provide reliable stock estimation while remaining cost-effective, responsive, and easily integrated into existing manufacturing processes [1], [5].

Traditional inventory control methods including manual counting, periodic audits, and barcode scanning report inventory status only at fixed checkpoints rather than continuously, as discussed earlier in this section. This is sufficient for warehouse environments, where buffer stock absorbs the delay between an actual stock change and when it is next recorded. It is not sufficient for kitting operations, however, since a single missing component can halt an assembly task within the time of one pick cycle, a timescale of seconds rather than the hours or shifts between periodic checks. Therefore, there is a clear need for practical investigation into intelligent real-time inventory monitoring solutions specifically designed for manufacturing kitting operations, where intelligence is achieved through adaptive and sensor fusion approaches and, where applicable, AI-based decision support capabilities.

This thesis aims to address this need by designing, implementing, and evaluating inventory monitoring systems that progressively explore sensor fusion, edge compu-

ting, artificial intelligence, and automated pick-to-light technologies. The work progresses through three phases: an initial laboratory prototype investigating sensor fusion for inventory estimation, an industrial prototype integrating automated pick-to-light with AI-assisted operator interaction, allowing an operator to check stock status and issue kitting requests without learning a fixed command interface, and a simulation-based study examining the scalability of the proposed approach in a multi-bin kitting environment. Together, these phases provide a comprehensive investigation into how intelligent inventory monitoring technologies can improve inventory visibility, automate stock verification, and support more reliable kitting processes.

1.1 Purpose of the Study

The purpose of this degree project is to investigate how real-time inventory monitoring can support improved inventory visibility and automated stock verification in manufacturing kitting operations. Although several technologies for inventory control have been developed in recent years, many manufacturing companies still rely on partially manual or periodic update systems that provide limited real-time visibility.

This study investigates the design, implementation, and evaluation of intelligent inventory monitoring prototypes that progressively explore sensor fusion, AI-assisted pick-to-light operation, and scalable multi-bin inventory monitoring for kitting applications. Through these investigations, the thesis evaluates how intelligent sensing and edge-based processing can support continuous stock monitoring, automated pick verification, and improved inventory visibility while remaining technically feasible for manufacturing applications.

1.2 Aim

The aim of this thesis is to investigate how intelligent real-time inventory monitoring can improve inventory visibility and automate stock verification in manufacturing kitting operations through the design, implementation, and evaluation of three progressively developed prototypes: a sensor-fusion-based laboratory system, an AI-assisted automated pick-to-light system deployed in an industrial setting, and a simulation-based multi-bin system for investigating scalability.
Sub-goals:

- To develop a system architecture that integrates IoT-based sensing and edge computing for real-time inventory monitoring in kitting environments.
- To design, implement, and evaluate a sensor-fusion-based prototype capable of estimating component availability in a kitting bin by combining weight-based and vision-based sensing.
- To design, implement, and evaluate an industrial pilot by automating an existing manually confirmed pick-to-light system using weight-based sensing and an AI-based kitting assistant capable of interpreting operator requests in natural language.
- To investigate and evaluate the scalability of the proposed monitoring approach through a simulation-based multi-bin kitting system incorporating sequential pick guidance and automatic detection of faults and tampering events.

1.3 Research questions

- How can a weight- and vision-based sensor fusion approach improve the reliability of real-time inventory monitoring for manufacturing kitting operations?
- How can manually confirmed pick-to-light systems be enhanced through automated sensor-based stock verification and AI-assisted operator interaction?
- How can the proposed inventory monitoring concept be extended from a single-bin prototype to a scalable multi-bin kitting system, and what technical challenges emerge during this transition?

1.4 Limitations

The scope of this degree project is limited to the design, implementation, and evaluation of prototype inventory monitoring systems for manufacturing kitting operations. The following limitations apply:

- The project focuses on inventory monitoring at the kitting station level and does not include full warehouse management or enterprise resource planning (ERP) integration.
- The industrial implementation was limited to a single-bin pick-to-light system at the collaborating company, Binar Solutions AB. The scalability of the proposed approach to multi-bin kitting systems was therefore investigated through simulation rather than full industrial deployment.
- The load cells used across all three prototypes have a rated capacity of 5 kg, which limits the total weight of components that can be reliably monitored in a single bin.
- The proposed systems were evaluated using prototype hardware and controlled laboratory experiments. Consequently, the obtained results may differ from those obtained in large-scale industrial manufacturing environments.
- The AI-assisted kitting system depends on a cloud-based large language model (LLM) service. Therefore, response time and availability may be influenced by external network conditions and cloud service performance.

2 Related work (Background)

Inventory management plays a critical role in modern manufacturing and logistics systems. Efficient control of inventory ensures that materials are available when required, while avoiding excessive stock levels and operational delays. In manufacturing environments that rely on assembly processes, particularly kitting operations, accurate and timely inventory information is essential to maintain continuous production flow. Recent advances in Industry 4.0 technologies such as the Internet of Things (IoT), Radio Frequency Identification (RFID), computer vision, and intelligent monitoring systems have significantly improved the capability to track and manage inventory in real time.

This chapter presents an overview of previous research related to inventory monitoring, IoT-based inventory management systems, sensing technologies, and kitting operations. The aim of this review is to identify existing approaches, highlight their strengths and limitations, and establish the research gap addressed in this thesis.

2.1 Inventory Management Challenges

Traditional inventory management systems often rely on manual data entry, barcode scanning, or periodic stock audits [1], [7]. These approaches frequently result in delays in updating stock information, human errors, and limited visibility of inventory status. As a consequence, organisations may experience stockouts, overstocking, and production interruption [1], [6], [7].

Research by Kang and Gershwin [6] shows that even small inaccuracies in inventory records can lead to significant operational problems. Their study demonstrates that a stock loss of only 1% can result in up to 17% loss in customer demand due to stockouts, as inventory systems may incorrectly assume that items are still available when they are not physically present [6]. Such inaccuracies are particularly problematic in lean manufacturing systems where safety stock levels are minimal.

Similarly, traditional inventory control systems have been shown to suffer from latency issues because stock updates are only performed periodically [1]. This lack of real-time visibility reduces the ability of companies to respond quickly to shortages or operational disruptions [1]. These limitations have motivated the development of IoT-based inventory monitoring systems that enable continuous data collection and real-time inventory visibility. Such systems are discussed in the following section.

2.2 IoT-Based Inventory Monitoring Systems

Recent research has explored the use of Internet of Things technologies to improve inventory tracking and monitoring [1]. IoT systems enable continuous data collection from physical objects using sensors and communication networks, allowing real-time monitoring and automated decision-making [1], [4].

Akeem and Sonu [1] proposed an IoT-enabled inventory tracking system that utilises passive RFID tags, fixed RFID readers, and a Raspberry Pi gateway to collect and transmit inventory data to a cloud platform. Their experimental results showed that the

IoT-based system achieved an inventory accuracy of 99.8% compared to 85% in traditional barcode-based systems. Furthermore, the data update latency was reduced from several hours to less than five seconds [1].

Other research [7] has also highlighted the advantages of IoT-based inventory systems. Studies have shown that integrating RFID sensors with wireless communication technologies enables automatic identification and tracking of inventory items without manual intervention [7]. These systems typically follow a three-layer architecture consisting of a perception layer for data collection, a network layer for communication, and an application layer for data processing and visualisation.

IoT-based inventory management has also been shown to reduce labour requirements and improve operational efficiency by automating tasks such as stock counting and replenishment monitoring [4]. Despite these advantages, the performance of IoT-based inventory monitoring systems largely depends on the sensing technologies used to collect inventory information. Consequently, various sensing approaches have been investigated for different industrial applications, as discussed in the following section.

2.3 Sensor Technologies for Inventory Monitoring

Several sensing technologies have been investigated for inventory monitoring in industrial and warehouse environments. Among the most widely used technologies are RFID systems, ultrasonic sensors, weight-based sensors, and computer vision systems [7], [8], [11].

RFID technology enables automatic identification of objects using radio signals. RFID tags attached to products can be detected by RFID readers, allowing the system to track item movements and update inventory records automatically [7], [8]. RFID-based inventory monitoring systems have demonstrated high accuracy and reliability in many industrial applications, particularly in warehouse logistics and supply chain management [7], [8]. However, RFID systems also have several limitations. Each item must be equipped with an RFID tag, which increases implementation costs when dealing with large numbers of small components. In addition, signal interference, tag collisions, and limited reading ranges can affect detection reliability, particularly in dense industrial environments.

Ultrasonic sensors have also been used for inventory monitoring by measuring the distance between the sensor and the surface of stored items [11]. In such systems, the sensor is typically mounted above a storage bin or shelf and continuously measures the distance to the top layer of items [11]. Variations in the measured distance indicate variations in the volume or height of stored components, which can then be used to estimate the remaining stock level. For example, an IoT-based inventory monitoring system using ultrasonic sensors achieved an accuracy of approximately 98% when estimating stock levels on structured storage shelves [11]. However, ultrasonic sensing has limitations when items are stored irregularly or when the surface of the items is uneven. In such cases, distance measurements may fluctuate, reducing the accuracy of quantity estimation.

To improve reliability, several studies have proposed combining multiple sensing technologies in a single monitoring system [3], [12]. For instance, a dual-layer inventory monitoring system combining RFID identification with ultrasonic sensing for quantity estimation was proposed in [12]. In this approach, RFID technology is used to verify the identity of items, while ultrasonic sensors monitor the physical stock level within

containers. Although this combination improves monitoring accuracy, the system becomes more complex and costly due to the need for multiple sensing devices and additional integration.

Another research direction involves the use of weight-based sensing methods. Load cells or weight sensors can measure the total weight of items stored in a container and estimate the quantity of components based on known unit weights [13]. Such systems can detect picking actions by measuring small weight variations and can provide continuous monitoring of inventory levels [13]. However, weight-based systems require accurate calibration and may produce incorrect estimates when different components with varying weights are stored in the same container. Despite these limitations, weight-based sensing remains an attractive solution for many manufacturing applications because of its simplicity, low cost, and ability to provide continuous monitoring of stock changes.

2.4 Computer Vision and Intelligent Monitoring Systems

Computer vision technologies have also been widely investigated for automated inventory monitoring because they can provide visual information about the presence, location, and quantity of stored items. Vision-based systems can detect, classify, count, and track objects using camera images together with artificial intelligence algorithms.

Omoleye [14] proposed a system that integrates computer vision with IoT sensors to monitor manufacturing processes in real time [14]. The system combines visual data from cameras with physical sensor data to improve monitoring accuracy. Experimental results showed that the fusion of computer vision and sensor data achieved higher accuracy than using either technology alone.

Similarly, an AI-powered inventory monitoring system called SMARTSTOCK used the YOLO object detection algorithm to identify and count items from camera images in a warehouse environment [9]. The system achieved over 90% accuracy in stock tracking and enabled real-time monitoring through a centralised dashboard.

Computer vision systems are particularly useful for detecting stock shortages, verifying item placement, and identifying incorrect component selection. However, they may be affected by challenges such as occlusion or variations in lighting conditions, reflective surfaces and camera positioning.

2.5 Kitting Systems in Manufacturing

Kitting is a common material handling strategy used in manufacturing assembly systems. In a kitting process, all components required for a specific assembly operation are collected and placed together in a kit container before being delivered to the assembly station.

Brynzer and Johansson [5] conducted one of the early studies on the design and performance of kitting systems in manufacturing environments [5]. Their research identified several factors influencing kitting efficiency, including storage policies, batching strategies, and work organisation.

The study found that inefficient material handling and information systems can significantly increase the time required for picking operations [5]. In some cases, only 16% of the total time in a kitting process is spent on the actual picking activity, while the remaining time is consumed by walking, searching for components, or administrative tasks.

To improve picking accuracy and reduce human errors, several operator support technologies have been introduced in assembly logistics. These include verification methods such as check-weighing systems, barcode scanning, and pick-to-light systems. Such solutions assist operators by confirming that the correct components are picked during the kitting process. However, these technologies primarily assist the picking process itself rather than continuously monitoring component availability within storage containers or automatically verifying that the correct quantity has been removed. Consequently, inventory status in many kitting systems may still depend on manual confirmation or periodic inventory checks, delaying the detection of shortages and limiting the level of automation that can be achieved.

These limitations indicate that there is still a need for intelligent inventory monitoring approaches capable of continuously verifying component availability while integrating operator-assisted kitting systems.

2.6 Hybrid and Intelligent Inventory Management Approaches

Recent research has focused on combining multiple technologies to create more intelligent and reliable inventory management systems [3], [8], [10], [13].

For example, hybrid tracking systems using both RFID and Bluetooth Low Energy (BLE) have been developed to improve the ability to determine the physical location of items within manufacturing environments [8]. In such systems, RFID readers are typically installed at entry and exit points to identify items when they move between areas, while BLE beacons provide continuous zone-level tracking inside production spaces. By combining these technologies, the system can determine not only which item is present but also in which production zone, storage area, or workstation the item is located.

Other studies have proposed more advanced frameworks that combine robotics, IoT, and artificial intelligence [10], [13]. A smart robotic inventory management system integrated a robotic arm, computer vision, and RFID verification to automate pick-and-place operations and update inventory records in real time [10].

Similarly, the Smart IoT-Enabled Real-Time Inventory Management System (SIRIMS) integrates RFID sensors, smart shelves, AI cameras, and machine learning models to provide real-time inventory visibility and predictive analytics [3].

These systems demonstrate the potential of combining multiple sensing technologies and intelligent algorithms to create more robust inventory monitoring solutions.

2.7 Multi-Sensor Fusion for Inventory Monitoring

Recent research has investigated the application of multi-sensor fusion techniques to improve inventory monitoring accuracy and robustness [3], [14]. A comprehensive example is the framework proposed by Lin [15], which combines weight sensing, vision-based measurements, and probabilistic data fusion to estimate inventory levels more reliably than individual sensing technologies.

The study demonstrated that single-sensor approaches often suffer from limitations such as measurement noise, calibration sensitivity, occlusion, and incomplete observations, which introduce uncertainty into the estimated inventory state [15]. By integrating complementary sensing modalities and applying probabilistic fusion methods, the framework was shown experimentally to maintain accurate inventory estimates even

under challenging conditions such as sensor noise, partial visibility, and dynamic changes in stored items [15].

Although the proposed framework demonstrates the advantages of multi-sensor fusion, its implementation requires multiple sensing devices, probabilistic fusion algorithms, and increased computational resources [15]. Such added complexity may present practical challenges when deploying inventory monitoring systems in industrial environments where simplicity, cost-effectiveness, and ease of integration are often key design requirements.

2.8 Pick-to-Light and Operator-Guided Picking Systems

Pick-to-light (PTL) is a light-directed picking technology in which illuminated indicators mounted at storage locations guide an operator to the correct bin, while a display communicates the required pick quantity. Trojanowska et al. [16] developed a custom Poka-Yoke-based PTL device for a small-component assembly task and evaluated it in a case study with 30 workers, reporting a 7-35% reduction in assembly time and a reduction in the total number of assembly errors from 34 to 22, a 35% decrease, compared with the same task performed without the device [16].

In their implementation, pick confirmation is handled by an optical-reflective sensor that detects the operator's hand breaking a light beam inside the bin, after which the system automatically advances to the next component in the sequence [16]. This removes the need for a manual button press; however, the confirmation remains a presence check rather than a quantity verification. The sensor can detect that a hand entered the correct bin, but not how many units were removed. The authors also describe a broader PTL architecture that integrates barcode scanning with inventory management software for inventory updates [16]. However, this approach still depends on discrete scan events rather than continuous physical measurement of the bin contents.

Beyond improvements in picking efficiency, PTL research has also examined the impact of these systems on operator workload and situation awareness. Stockinger et al. [17] compared a pick-by-light system with a conventional pick-by-paper method in a controlled study involving 31 participants. The study found no significant difference in operators' situation awareness between the two methods. However, participants using the pick-by-light system exhibited higher physiological strain, measured through heart rate, whereas participants using the pick-by-paper method reported a higher subjective workload [17]. These findings indicate that the benefits of PTL systems extend beyond picking accuracy and efficiency, with different guidance methods influencing the operator experience in different ways.

An earlier lamp-and-display example, described in one of the case studies of Brynzér and Johansson [5], reports few picking errors but notes the trade-off of high equipment cost and increased idle time from picker queues at a shared station [5].

Across the reviewed literature, PTL systems primarily focus on guiding operators to the correct picking location and improving picking efficiency. Confirmation mechanisms are typically based on manual button presses, barcode scanning, or simple presence detection rather than continuous physical verification of the quantity of components removed from the bin. Consequently, inventory updates continue to depend primarily on operator actions rather than direct physical verification of stock changes. As a result, discrepancies between expected and actual inventory levels may remain unde-

tected until a later manual inventory check. These limitations highlight the need to integrate real-time sensing technologies with PTL systems to enable automatic stock verification and continuous inventory monitoring during kitting operations.

2.9 Large Language Model Agents for Industrial Decision Support

Recent research has begun to investigate the use of large language models (LLMs) as intelligent interaction and decision-support agents within industrial automation systems [18], [19], [20]. Rather than replacing conventional control systems, LLMs have been explored as interfaces capable of interpreting natural language instructions, reasoning about production information, and coordinating interactions between operators and existing automation software.

Xia et al. [18] proposed enhancing a flexible modular production system using LLM-based agents capable of interpreting operator requests expressed in natural language and translating them into appropriate control actions. Their work demonstrates that LLM agents can simplify system interaction and reduce the need for specialised reprogramming when adapting manufacturing systems to new production tasks. In a subsequent study, Xia et al. [19] presented a framework for controlling industrial automation systems using LLM agents integrated with structured information models. In this architecture, real-time production data are supplied to the language model through a structured representation of the current plant state, enabling the model to interpret system conditions and generate context-aware control decisions while maintaining compatibility with existing industrial control architectures.

An important capability supporting such industrial applications is function calling, in which an LLM invokes predefined external software functions or application programming interfaces (APIs) instead of relying solely on free-form text generation. Wang et al. [20] reviewed current industrial practices in LLM function calling and identified several challenges related to parameter accuracy, tool selection, execution validation, and reliable interaction with external systems. The survey concludes that successful industrial deployment depends not only on the language model itself but also on robust mechanisms for verifying and executing the functions selected by the model.

Across the reviewed literature, a consistent design principle emerges: the primary role of an LLM is to interpret operator intent and coordinate appropriate system actions, while the execution of control logic and validation of system state remain the responsibility of external software and automation systems. This separation improves system reliability by ensuring that language-based reasoning is supported by deterministic control mechanisms. However, most existing studies focus on production control, modular manufacturing systems, or general industrial automation, with relatively limited attention given to the application of LLMs for inventory monitoring and operator assistance within manufacturing kitting systems.

2.10 Research Gap

Although considerable progress has been made in IoT-based inventory monitoring, intelligent sensing technologies, and AI-supported manufacturing systems, the reviewed literature reveals several important research gaps. Most existing inventory monitoring solutions have been developed for warehouse management, logistics, or large-scale storage environments, where the primary objective is to identify, locate, and track inventory

throughout the supply chain [1], [3], [7]. In contrast, manufacturing kitting operations require continuous monitoring of component availability directly at the workstation, where the absence of a single component can interrupt an assembly process. Consequently, the operational requirements of workstation-level kitting differ significantly from those of warehouse inventory management, yet relatively few studies have specifically addressed this application.

Furthermore, although individual sensing technologies such as RFID, ultrasonic sensing, weight-based sensing, and computer vision have each demonstrated advantages for inventory monitoring, every approach also exhibits practical limitations related to cost, sensing reliability, environmental conditions, or deployment complexity [7], [8], [11], [13]. Multi-sensor fusion has been shown to improve estimation robustness by combining complementary sensing modalities [15]. However, existing sensor-fusion frameworks are generally designed for broad inventory monitoring applications and frequently rely on probabilistic fusion algorithms, multiple sensing devices, and comparatively complex system architectures. As a result, there remains a need to investigate how sensor-fusion principles can be applied to practical workstation-level kitting applications while maintaining a balance between monitoring performance and implementation complexity.

A further gap exists in the integration of inventory monitoring with operator-guided picking systems. Existing pick-to-light solutions primarily improve picking accuracy by guiding operators to the correct storage location, while confirmation is typically based on manual button presses, barcode scanning, or simple presence detection rather than direct verification of the quantity physically removed from the bin [5], [16], [17]. Consequently, inventory updates continue to depend largely on operator interaction instead of continuous physical measurement of stock changes, limiting the ability to detect discrepancies immediately during the picking process.

Recent studies have also demonstrated the potential of large language models to support industrial automation by interpreting natural language instructions and coordinating system actions [18], [19], [20]. However, these applications have focused primarily on modular production systems and industrial control architectures. The application of LLM-based operator assistance for manufacturing kitting, particularly when integrated with real-time inventory monitoring and automated pick confirmation, remains largely unexplored.

Therefore, this thesis addresses these research gaps by investigating intelligent inventory monitoring specifically for manufacturing kitting operations. The work examines the use of multi-sensor fusion to improve workstation-level inventory estimation, explores the integration of real-time inventory monitoring with automated pick-to-light operation and AI-assisted operator interaction, and investigates how these concepts can be extended towards scalable multi-bin kitting systems.

3 Method

This chapter describes the methodology used to design, develop, and evaluate the intelligent inventory monitoring system presented in this thesis. Section 3.1 introduces the overall research methodology adopted throughout the project. Section 3.2 presents the general system architecture that forms the foundation of the proposed solution and addresses the first sub-goal defined in Section 1.2. Sections 3.3 to 3.5 describe the design and implementation of the three development phases, while Sections 3.6 and 3.7 present the data collection procedure and the evaluation methodology used to assess the proposed system.

3.1 Methodological Approach

This degree project follows a design-oriented research methodology, in which knowledge is generated through the design, implementation, and evaluation of prototype systems developed to address a practical engineering problem. Rather than relying solely on theoretical analysis, this methodology emphasises the creation and experimental evaluation of functional artefacts to investigate research questions and evaluate proposed solutions [21].

This methodology was selected because the objective of the thesis is not only to investigate the challenges associated with real-time inventory monitoring in manufacturing kitting operations but also to develop and evaluate practical prototype systems capable of addressing those challenges. Prototype-based experimentation enables proposed solutions to be validated under controlled and industrially relevant operating conditions, providing practical evidence of their feasibility, strengths, and limitations.

The research was conducted through three sequential development phases, each investigating a different aspect of intelligent inventory monitoring while building upon the findings of the previous phase. The first phase investigates the feasibility of a multi-sensor approach for improving workstation-level inventory monitoring by combining complementary sensing technologies. Building on the knowledge gained from this investigation, the second phase focuses on integrating weight-based inventory monitoring into an existing industrial pick-to-light system to enable automated stock verification and AI-assisted operator interaction. Finally, the third phase investigates the scalability of the proposed monitoring concept by extending it from a single-bin implementation to a coordinated multi-bin kitting system through simulation.

Each development phase concluded with an experimental evaluation to assess the proposed solution within its intended scope before informing the subsequent stage of development. Together, these three phases form the methodological framework used to investigate the research questions presented in Section 1.3 and to evaluate the proposed intelligent inventory monitoring concept at increasing levels of practical complexity.

3.2 System Architecture

To address the first sub-goal defined in Section 1.2, a common system architecture was developed and used as the foundation for all three development phases of this research.

Although the hardware configuration evolved throughout the project, the overall architectural principles remained unchanged. This ensured that new capabilities could be investigated while maintaining a consistent sensing, communication, and data-processing framework.

Figure 1 illustrates the general architecture of the proposed intelligent inventory monitoring system. The architecture is organized into five logical layers: the sensing layer, the messaging layer, the application layer, the presentation layer, and the operator interaction layer. Together, these layers provide a modular framework for acquiring inventory data, processing sensor information, supporting intelligent decision-making, and presenting inventory status to the operator.

The sensing layer represents the physical interface with the kitting environment. Depending on the development phase, this layer consists of different sensing devices, including load cells, a depth camera, and pick-to-light indicators. These devices continuously acquire information related to component availability and operator interactions.

The messaging layer provides communication between the sensing hardware and higher-level software components. Throughout all three phases, MQTT was adopted as the communication protocol, enabling reliable publish-subscribe messaging between distributed sensing devices and the application layer while keeping sensing and processing components loosely coupled.

The application layer contains the core inventory monitoring logic. It receives sensor data through the messaging layer, processes inventory events, maintains the current inventory state, and executes the algorithms investigated during each development

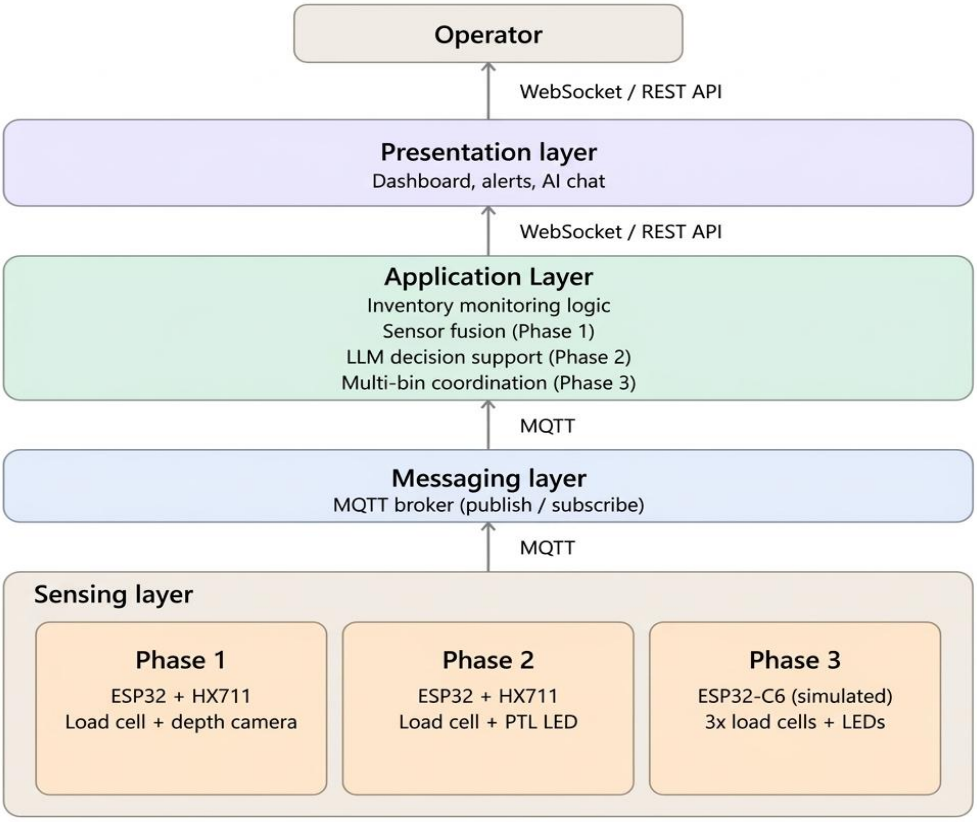


Figure 1. General architecture of the proposed intelligent inventory monitoring system.

phase. Depending on the objective of each phase, the application layer incorporates sensor fusion (Phase 1), LLM-based decision support (Phase 2), or multi-bin coordination mechanisms (Phase 3).

The presentation layer provides the user-facing interface through which inventory information is visualised. Depending on the prototype, this layer includes real-time dashboards, inventory alerts, and an AI-assisted conversational interface that communicates system status and supports operator interaction.

Finally, the operator interaction layer represents the human user who monitors inventory status, receives notifications, and interacts with the system during kitting operations. Separating the operator from the underlying system architecture emphasises the flow of information from physical sensing to human decision support.

Although the sensing hardware and application functionality differ between the three development phases, the layered architecture shown in Figure 1 remains consistent throughout the project. The specific implementation of each phase is presented in Sections 3.3-3.5.

3.3 Phase 1: Sensor Fusion Prototype

3.3.1 Prototype Objective

Phase 1 addresses the first research question by investigating whether combining two complementary sensing modalities can improve the reliability of real-time inventory monitoring compared with either sensing modality operating independently. The prototype was designed specifically to test this by combining a weight sensor, which can reliably measure quantity but cannot assess part identity or condition, with a vision sensor, which can assess part shape and condition but is limited by occlusion and cannot see items hidden beneath others. The objective was not only to investigate whether sensor fusion improves inventory monitoring reliability, but also to determine whether combining complementary sensing modalities enables inventory-related observations that cannot be achieved using either sensing technology independently.

3.3.2 Hardware Architecture

The Phase 1 prototype implements the general architecture introduced in Section 3.2 using three hardware components. A load cell of 5 kg, interfaced through an HX711 amplifier, measures bin weight. An Intel RealSense depth camera provides synchronised colour and depth image streams of the bin. An ESP32 microcontroller was selected as the edge device interfacing with the load cell because it offers integrated Wi-Fi connectivity, sufficient processing capability for sensor interfacing, and low power consumption, avoiding the need for a separate communication module at the bin itself. A Raspberry Pi was selected as the central processing platform because it provides a compact, energy-efficient embedded computing platform capable of performing continuous camera acquisition, image processing, and sensor-fusion operations in real time. Unlike a conventional desktop computer, the Raspberry Pi can be integrated directly into the kitting workstation while also providing sufficient interfaces to accommodate additional sensors, displays, and other peripheral devices as the system evolves. This makes it well suited for the development of a modular and extensible inventory monitoring system.

The hardware was integrated into a single-bin laboratory prototype representing a workstation-level kitting environment. While the hardware provides the sensing capability required for inventory monitoring, the research contribution of Phase 1 lies in the

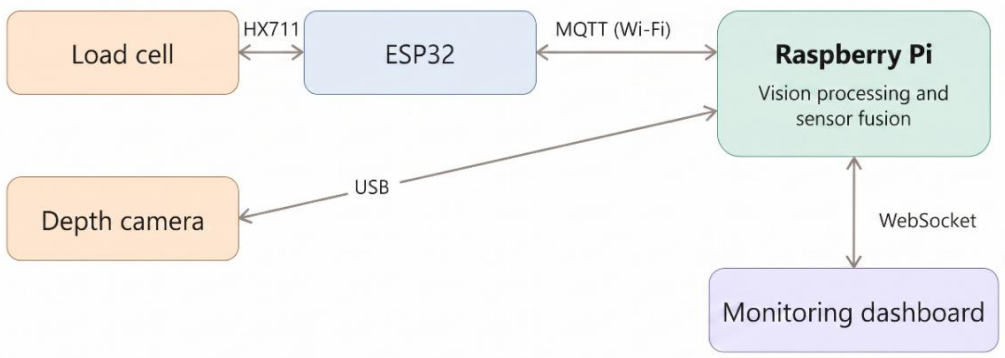


Figure 2. Hardware architecture of the Phase 1 sensor-fusion prototype.

methodology used to combine information from the different sensing modalities. This methodology is described in the following section.

3.3.3 Sensor Fusion Methodology

The sensor fusion methodology was designed around two principles. First, measurements from each sensing modality were evaluated independently before contributing to the final inventory estimate. Second, complementary information from weight sensing and vision sensing was combined to improve confidence in the estimated inventory state while reducing the limitations associated with individual sensors.

Weight sensing provided continuous quantitative measurements of component availability, whereas the depth camera observed the physical arrangement of components within the kitting bin. By combining these complementary sources of information, the prototype investigated whether sensor fusion could support inventory-related observations that would not be possible using either sensing modality independently.

To support this investigation, the prototype was designed to operate under different sensing conditions depending on the availability of each channel, allowing fused operation to be directly compared against each sensing modality operating alone. This design choice is what makes the single-sensor comparisons described in Section 3.3.5 possible.

A particular capability investigated during this phase was whether combining weight and depth information could identify situations in which components were physically present but partially occluded from the camera's field of view, making them difficult to detect using vision alone. This capability represents the primary research investigation of the Phase 1 prototype and is evaluated experimentally in Chapter 5.

3.3.4 Software Architecture

The software architecture separates sensor acquisition, communication, inventory processing, and user interaction into independent components. This modular design simplified prototype development and allowed individual software modules to be modified without affecting the overall system architecture. Detailed implementation of each software component is presented in Chapter 4.

3.3.5 Experimental Configuration and Evaluation Metrics

The prototype was evaluated using a structured protocol designed to investigate the influence of sensing modality on inventory monitoring reliability: a baseline condition using only correct parts, a mixed-part condition combining correct and incorrect items, isolated single-sensor conditions achieved by physically disabling one channel at a time,

and a matched controlled trial in which a single fixed set of parts was evaluated sequentially under all sensing conditions without altering bin contents between trials.

Inventory accuracy is measured by comparing the system's reported quantity against the manually verified ground truth established for each trial, as described in Section 3.6. Sensor agreement measures the consistency between the weight-based and vision-based inventory estimates, while confidence represents the system's current reliability based on sensor availability, detected incorrect parts, and disagreement between the sensing channels. Hidden-part detection is tested by deliberately placing one part underneath another so that it is not visible to the camera, and recording whether the hidden-part indicator activates within the expected detection window. Latency measures the time required for sensing and processing information to be reflected in the system output. The detailed calculation of these metrics is described in Chapter 4.

3.4 Phase 2: Industrial Pilot - AI-Assisted Automated Pick-to-Light System

3.4.1 Prototype Objective

Phase 2 addresses the second research question by investigating how a manually confirmed pick-to-light (PTL) system can be enhanced to provide reliable real-time stock verification while supporting natural-language interaction for operator assistance. This directly addresses the limitation identified in Section 2.8, where conventional PTL systems guide operators to the correct storage location but do not independently verify that the correct quantity of components has been removed.

Building upon the findings obtained from the Phase 1 sensor-fusion prototype, this phase investigates a practical industrial solution based on a single sensing modality. Rather than combining multiple sensors, the objective is to determine whether continuous weight sensing can provide reliable automated stock verification while remaining sufficiently simple for integration into an existing industrial pick-to-light workstation.

The prototype was developed in collaboration with **Binar Solutions AB** and extends an existing pick-to-light installation by integrating automatic stock verification, real-time inventory monitoring, and AI-assisted operator interaction. The objective is not only to guide operators during picking operations, but also to verify that the measured physical inventory change corresponds to the expected picking action without relying solely on manual confirmation.

3.4.2 Hardware Architecture

The Phase 2 prototype implements the general architecture introduced in Section 3.2 using a practical hardware configuration integrated into an existing industrial pick-to-light workstation at Binar Solutions AB. A load cell of 5 kg, interfaced through an HX711 amplifier continuously measures the weight of components stored within a single kitting bin. An ESP32 microcontroller was selected as the edge device because it provides integrated Wi-Fi connectivity, sufficient processing capability for continuous sensor acquisition, and straightforward integration with the existing pick-to-light hardware while publishing sensor measurements to the application layer through MQTT.

An existing pick-to-light LED indicator provides visual guidance by identifying the storage location associated with each picking operation. Weight sensing was selected as the stock verification mechanism because it provides direct physical evidence of inventory change, whereas conventional confirmation methods such as button presses or

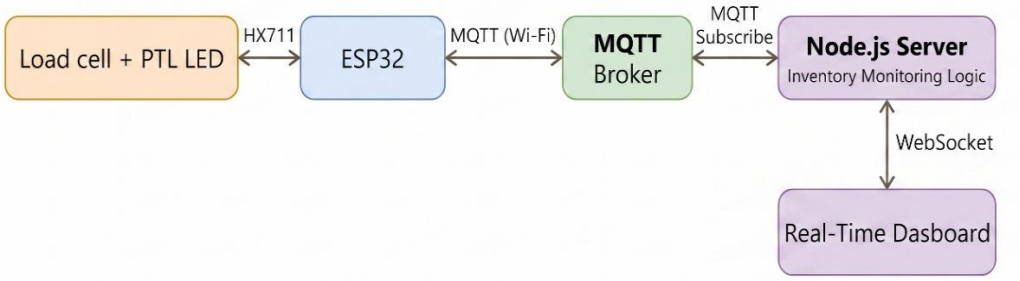


Figure 3. Hardware architecture of the Phase 2 industrial inventory monitoring prototype.

barcode scans only confirm that an operator has performed an action. Consequently, weight sensing directly addresses the confirmation gap identified in Chapter 2 while remaining compatible with an existing industrial pick-to-light installation.

While the hardware provides the sensing capability required for automated stock verification, the primary research contribution of Phase 2 lies in the methodology used to integrate automatic stock verification with operator-guided picking. This methodology is described in the following section.

3.4.3 System Methodology

The methodology investigated in Phase 2 extends a conventional pick-to-light workflow by introducing automated stock verification into the picking process. Unlike traditional pick-to-light systems, where inventory updates depend primarily on manual confirmation, the proposed methodology investigates whether continuous weight sensing can provide an independent physical verification of stock changes during kitting operations.

In addition to automated stock verification, the prototype investigates the use of a large language model as a natural-language interaction layer for supporting kitting operations. Operators may initiate picking operations through manual interaction, production orders, or natural-language requests. Regardless of how a picking operation is initiated, all requests follow the same stock verification methodology before the inventory state is updated.

A key methodological principle investigated in this phase is the separation between natural-language interpretation and deterministic stock verification. Rather than allowing the language model to act directly on inventory data or hardware, AI-generated requests are independently verified against the current sensor-derived inventory state before any physical action is performed. Furthermore, the prototype investigates whether an additional, independently invoked AI review step can provide an additional safeguard beyond deterministic verification alone. This layered methodology allows natural-language interaction to support operator decision-making while maintaining deterministic stock verification as the authoritative basis for physical actions.

The prototype also investigates inventory monitoring during the execution of production-order queues, allowing stock availability to be evaluated across multiple scheduled orders rather than only the current picking request. In addition, the interaction layer was designed to support both Swedish and English to assess the practicality of natural-language interaction within an industrial working environment.

The proposed methodology forms the basis for the experimental evaluation presented in Chapter 5, where its ability to provide reliable automated stock verification and AI-assisted operator interaction is assessed.

3.4.4 Software Architecture

The software architecture separates sensing, communication, deterministic inventory management, AI-assisted interaction, and user presentation into independent software components, so that the deterministic inventory-verification logic and the AI interaction layer can each be developed, tested, and extended without affecting the other or the underlying communication infrastructure. Detailed implementation of each software component is presented in Chapter 4.

3.4.5 Experimental Configuration and Evaluation Metrics

The prototype was evaluated at the collaborating company across a series of live picking sessions, designed to assess both the physical reliability of automated pick confirmation and the correctness of the AI interaction layer independently, since a pick can be physically correct while the assistant's interpretation of a request is not, or vice versa.

Inventory accuracy is measured as the percentage of picks correctly confirmed by the weight-sensing system, accounting for both reliable sensing and correct fulfilment of the requested quantity. As a complementary check, a subset of trials additionally compares the system's confirmed quantity against a manually verified true quantity, established as described in Section 3.6, to assess whether this metric reflects genuine physical correctness. Tool-call correctness is measured as the proportion of AI-initiated pick requests that successfully pass all guardrails in the layered safety architecture described in Section 3.4.3. Two latency measures are evaluated separately: pick-confirmation latency, defined as the time between a physical weight-change event and the corresponding dashboard update, and assistant response latency, defined as the time between an operator's message and the assistant's reply. These latency measures represent different processing paths within the system and are therefore evaluated independently. The detailed classification criteria are described in Chapter 4.

3.5 Phase 3: Simulation-Based Multi-Bin Scalability Study

3.5.1 Prototype Objective

Phase 3 addresses the third research question (RQ3) by investigating how the single-bin inventory monitoring concept developed in Phase 2 can be extended to a coordinated multi-bin kitting station, and what additional challenges emerge when scaling the proposed approach beyond a single workstation. To investigate this, the proposed methodology is extended to a simulated multi-bin kitting environment implemented using the Wokwi electronics simulator. The simulation provides a virtual representation of the multi-bin prototype, allowing the scalability of the inventory monitoring approach and the coordination of multiple inventory locations to be investigated.

3.5.2 Hardware Architecture

The Phase 3 prototype implements the general architecture introduced in Section 3.2 using a simulated hardware configuration representing a multi-bin kitting workstation. The prototype consists of three independent 5 Kg load-cell channels, each associated with its own pick indicator, together with a shared operator interface comprising a push button and LCD display. All sensing channels are coordinated by a single ESP32-C6 microcontroller, allowing multiple inventory locations to be monitored simultaneously within a common control architecture.

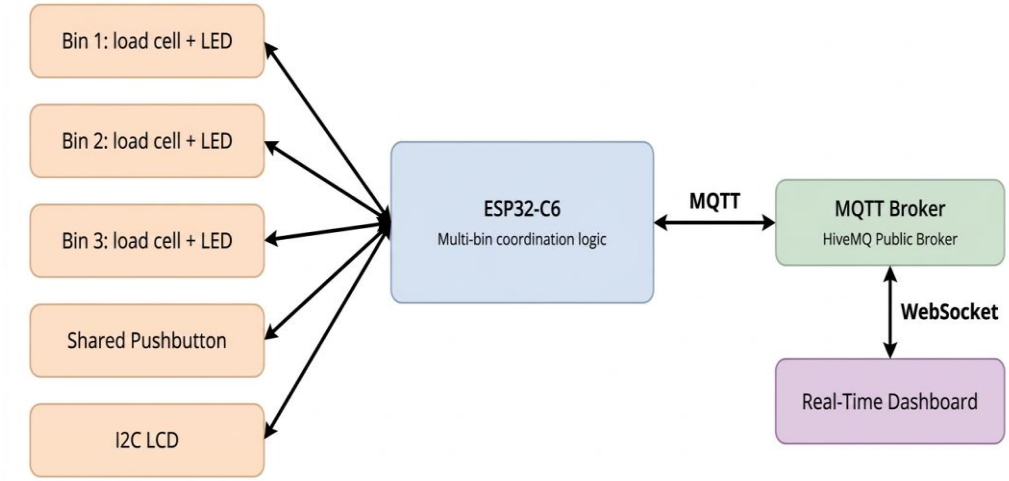


Figure 4. Hardware architecture of the Phase 3 multi-bin simulation prototype.

While the simulated hardware provides the sensing and communication framework required for the study, the primary research contribution of Phase 3 lies in the methodology used to coordinate multiple independently monitored bins. This methodology is described in the following section.

3.5.3 System Methodology

The methodology investigated in Phase 3 extends the single-bin monitoring approach developed in Phase 2 to a coordinated multi-bin kitting environment. Rather than monitoring only one storage location, the prototype investigates whether independent inventory states can be maintained simultaneously across multiple bins while preserving the same stock verification principles established in the previous phase.

In addition to maintaining independent inventory states, the prototype investigates the coordination of multiple bins during complete kitting operations. This includes sequencing picking tasks across several storage locations while ensuring that inventory monitoring remains independent for each bin.

The methodology also investigates inventory events occurring outside the active picking process, including unexpected inventory changes at inactive bins and the influence of independent calibration for multiple sensing channels. These aspects were included to investigate whether the proposed monitoring concept remains reliable when extended beyond a single sensing location.

3.5.4 Software Architecture

Unlike Phase 1 and Phase 2, where inventory logic runs on a separate processing platform, the coordination and monitoring logic in Phase 3 runs directly within the ESP32-C6 firmware, with the browser dashboard used only for remote monitoring and user interaction. This configuration was selected to investigate a lightweight implementation suitable for coordinating multiple sensing channels within a single embedded platform. Detailed implementation is presented in Chapter 4.

3.5.5 Experimental Configuration and Evaluation Metrics

The prototype will be evaluated through repeated simulated multi-bin trials focusing on inventory accuracy and the detection of unexpected inventory changes during multi-bin

operation. The evaluation will consider four performance indicators: Inventory Accuracy, Bins Completed, Corrections Needed, and Anomalies Detected. Inventory Accuracy will be calculated as the percentage of accurately completed bin picks relative to the total number of completed bin picks. Bins Completed represents the total number of completed bin picks, while Corrections Needed represents the number of over-pick corrections recorded during the picking sessions. Anomalies Detected represents the number of unexpected inventory changes detected during operation. These measures will be used to assess the reliability of the proposed multi-bin monitoring approach.

3.6 Data Collection and Experimental Procedure

Data collection follows a consistent experimental procedure throughout the evaluation of all three research phases. Although each prototype investigates a different research question, the experiments are conducted under controlled conditions and relevant system data are recorded to support repeatable and objective evaluation.

For each experimental scenario, the prototype is configured according to the procedure described in Sections 3.3.5-3.5.5 before a series of repeated experimental trials is performed. During each trial, sensor measurements, system responses, communication events, and inventory updates are recorded automatically using the corresponding software platform. Automatic data logging is used where available to provide timestamped records of system behaviour and to reduce reliance on manual observation during the experiments.

Where quantitative evaluation requires comparison with the actual inventory state, an independent ground-truth reference is established through manual verification of the physical inventory before and after each experimental trial. This reference is used only for performance evaluation and is not available to the inventory monitoring system during operation, ensuring that the reported results reflect the system's own estimation capability.

To improve the reliability of the evaluation, each experimental scenario is repeated under consistent operating conditions. Repeating the experiments under the same conditions reduces the influence of random variation and enables the performance of the proposed methodologies to be assessed consistently across the three research phases.

The experimental data collected through this procedure form the basis for the evaluation metrics described in Sections 3.3.5, 3.4.5, and 3.5.5. The corresponding results and discussion are presented in Chapter 5.

4 System Development and Implementation

4.1 Chapter Overview

This chapter presents the implementation of the three development phases introduced in Chapter 3. While Chapter 3 described the system architecture, design methodology, and evaluation approach adopted for each phase, this chapter focuses on their implementation, including the software and firmware developed, the algorithms employed, the flow of data through the system, and the engineering decisions made during development. Where appropriate, the implementation is related to the research questions presented in Section 1.3 and the literature reviewed in Chapter 2.

The chapter is organised in the same order as the three development phases. Section 4.2 presents the implementation of the Phase 1 sensor-fusion prototype. Section 4.3 presents the implementation of the Phase 2 industrial pick-to-light prototype developed in collaboration with Binar Solutions AB. Section 4.4 presents the implementation of the Phase 3 simulation-based multi-bin prototype. The results obtained from testing each of these implementations are presented and discussed in Chapter 5.

4.2 Phase 1: Sensor Fusion Prototype

4.2.1 System Implementation Overview

The Phase 1 prototype was implemented as a collection of five software components, each responsible for a specific function within the layered architecture introduced in Section 3.2. The implementation separates sensor acquisition, data processing, communication, user interaction, and offline analysis into independent modules, allowing each component to be developed and tested without affecting the overall system.

The vision processing and sensor-fusion logic were implemented as a single Python application running on the Raspberry Pi. This application acquires image data from the camera, receives weight measurements through MQTT, executes the fusion algorithm, records system data, and publishes the resulting inventory state. A second Python application, also running on the Raspberry Pi, provides the communication bridge between MQTT and the browser-based operator interface using Flask and Socket.IO, enabling real-time data visualization and operator interaction.

The weight sensing functionality was implemented separately on the ESP32 microcontroller, where firmware was developed to acquire load-cell measurements, perform calibration, and publish weight data through MQTT. Following experimental evaluation, an offline analysis script processes the recorded log files to generate the statistical summaries, tables, and figures used in Chapter 5.

This modular design simplified development by allowing each component to be developed, tested, and refined independently while maintaining a consistent communication framework across the prototype.

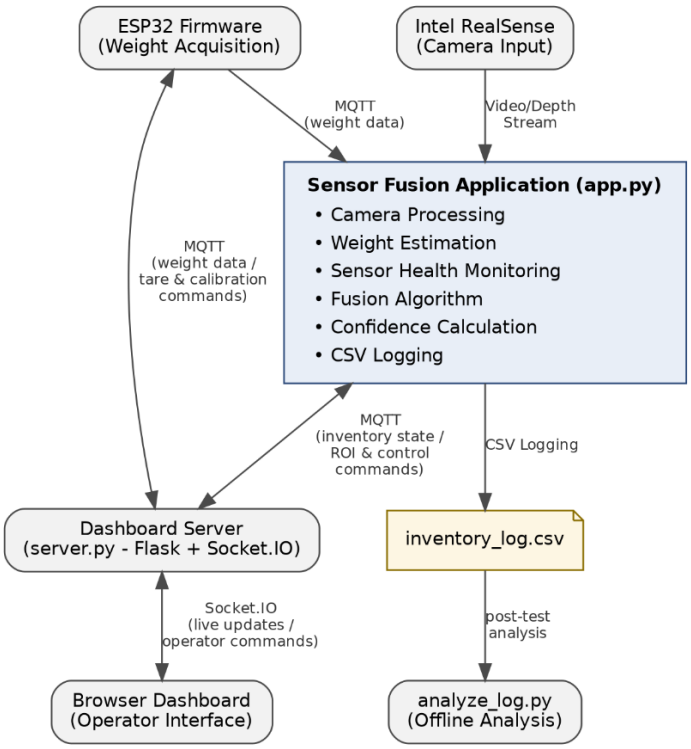


Figure 5. Software implementation of the Phase 1 sensor-fusion prototype.

4.2.2 Weight Channel Implementation

The weight sensing channel was implemented entirely on the ESP32 microcontroller. Since the load cell produces only a small analogue signal, it was interfaced through an HX711 load-cell amplifier, which converts the analogue output into digital measurements that can be processed by the ESP32. During normal operation, the firmware continuously acquires weight measurements from the HX711 by averaging three consecutive samples. This averaged measurement is then passed through an additional threshold-based smoothing step before being used to estimate the number of components in the bin. If the new reading differs from the previously smoothed weight by more than 5 g (0.005 kg), the firmware treats the change as significant and immediately adopts the new value. If the difference is 5 g or smaller, the change is treated as sensor variation and the smoothed value is updated gradually using 70% of the previous value and 30% of the new measurement. In addition, if the smoothed weight falls below 2 g (0.002 kg), it is set to exactly zero to prevent a genuinely empty bin from displaying a small non-zero value due to sensor noise. The resulting filtered weight is used for component-count estimation and is published to the application layer through MQTT.

In addition to continuous sensor acquisition, the ESP32 firmware implements the tare and calibration procedures required for reliable inventory monitoring. Both operations are initiated from the operator dashboard through dedicated MQTT commands. The tare procedure resets the measurement baseline when the container is empty, while the calibration procedure determines the average weight of an individual component using a known quantity of parts. The calculated unit weight is subsequently used to convert measured weight into an estimated component count during normal operation. For local operator feedback, the current weight and estimated part count are also displayed on a small OLED display mounted at the bin, allowing an operator standing at

the station to check the reading without needing to view the dashboard. Both the resulting tare offset and the calibrated unit weight are stored in the ESP32's non-volatile flash memory, ensuring that the calibration is retained across power cycles and firmware updates without requiring the calibration procedure to be repeated each time the system starts.

An important characteristic of the weight channel is its measurement latency. During experimental testing, consecutive weight updates were observed at intervals of approximately 270-280 ms(millisecond). This latency is primarily determined by the HX711 hardware rather than by the firmware, wireless communication, or MQTT transmission. The HX711 operates at a fixed sampling rate of approximately ten samples per second, and since the firmware averages three consecutive readings before producing an output, the effective update interval increases accordingly. Consequently, the latency of the weight channel represents an inherent characteristic of the sensing hardware rather than a limitation of the software implementation. This distinction is important because, unlike the vision channel, whose response time depends primarily on image-processing complexity, the performance of the weight channel is fundamentally constrained by the characteristics of the load-cell acquisition hardware.

4.2.3 Vision Channel Implementation

The vision sensing channel was implemented on the Raspberry Pi using an Intel RealSense depth camera, which provides synchronized colour and depth images for every frame. During operation, the application continuously processes each incoming frame to identify valid inventory components, detect incorrect parts, and provide the vision-based inventory estimate used by the sensor-fusion algorithm.

Before object detection, the raw depth measurements are stabilised using an exponential moving average (EMA) with a smoothing factor of 0.25. Since individual depth frames contain small frame-to-frame variations, this temporal smoothing reduces measurement noise while preserving the overall geometry of the observed components. Where a monitoring area has been selected through the operator dashboard, both the colour and depth images are restricted to the corresponding region of interest (ROI), ensuring that only objects located within the selected inventory area are analysed. If no monitoring area is defined, the complete camera frame is processed.

Object detection follows a contour-based image-processing pipeline. The colour image is first converted to greyscale, lightly blurred to reduce image noise, and threshold using Otsu's automatic thresholding method. A morphological closing operation is then applied to remove small gaps before object contours are extracted. Contours with an area smaller than 1500 pixels or larger than 5000 pixels are discarded to eliminate noise and unrelated objects before further analysis.

Each remaining contour is classified using a two-stage procedure. In this prototype, a correct component is explicitly defined as a square-shaped object, reflecting the square kitting parts used during development and testing. The geometric tests are therefore specifically tuned to recognise this geometry rather than arbitrary part types. The first stage evaluates its geometric characteristics, including the number of vertices in its polygon approximation, aspect ratio, bounding-box occupancy, circularity, and similarity to a rotated bounding rectangle, to determine whether the detected object matches the expected shape of a valid component. Contours satisfying these criteria are treated as candidate parts. The second stage evaluates the depth consistency of each candidate by analysing the variation of depth values across its surface. A component exhibiting a sufficiently uniform depth profile is classified as a correct part, whereas a candidate with

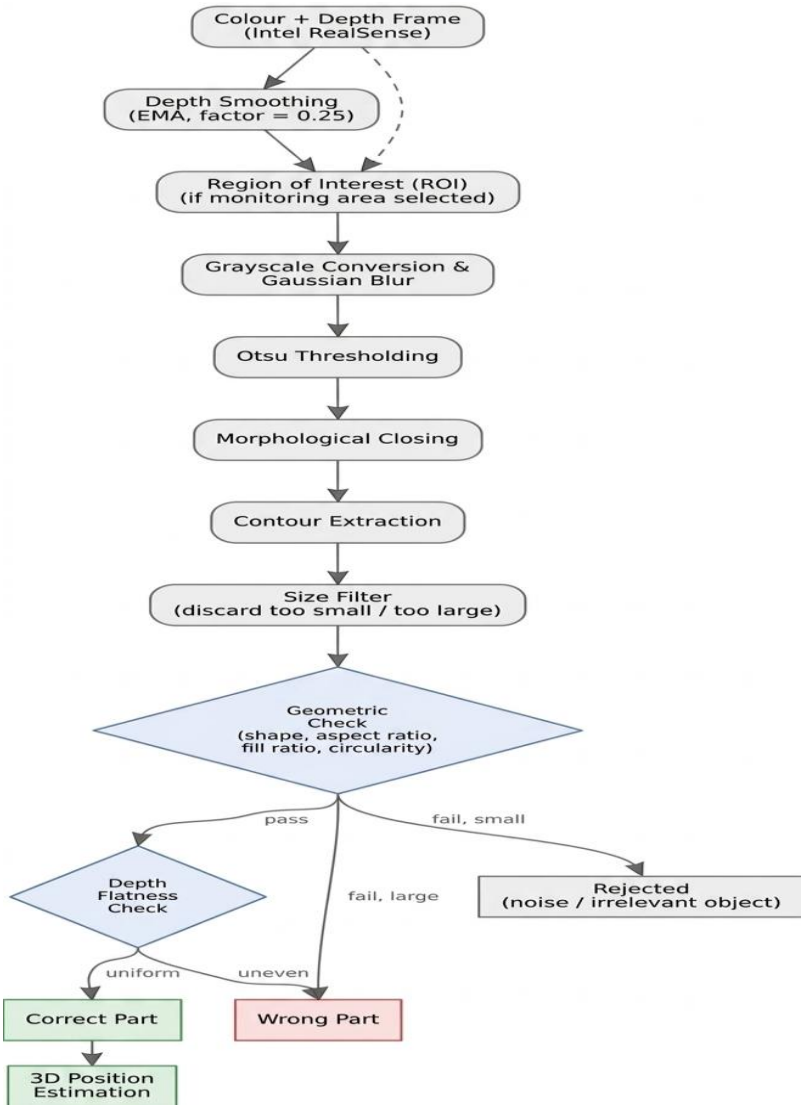


Figure 6. Vision processing pipeline implemented in the Phase 1 sensor-fusion prototype.

excessive depth variation is classified as an incorrect part, indicating that the object may be damaged, deformed, tilted, or otherwise improperly positioned. Similarly, sufficiently large contours that fail the geometric classification stage are also recorded as incorrect parts.

For each correctly classified component, the corresponding depth information is also used to compute its real-world three-dimensional position using the Intel RealSense SDK. When a monitoring area is active, the calculated coordinates are expressed relative to the selected region within the bin. Although these coordinates are not used directly by the inventory monitoring system developed in this work, they provide the foundation for future integration with robotic manipulation systems requiring three-dimensional component localisation.

The vision processing pipeline produces the estimated number of correct components, the number of detected incorrect parts, and the associated component position information for each processed frame. These outputs are forwarded to the adaptive sensor-fusion algorithm described in Section 4.2.5, where they are combined with the corresponding weight measurements to produce the final inventory estimate.

In addition to providing the vision-based inventory estimate used by the adaptive sensor-fusion algorithm described in Section 4.2.5, the vision channel also supports an automatic calibration mode for the weight channel. This procedure is initiated from the operator dashboard and executed as a background task on the dashboard server. The server first issues a tare command to the ESP32 and then prompts the operator to place only valid components in the bin for calibration. This requirement is necessary because the calibration procedure uses the number of correctly detected components produced by the vision channel as the known calibration quantity. The dashboard continuously monitors both the vision-based component count and the corresponding weight measurement until stable readings are obtained, after which the detected component count is automatically transmitted to the ESP32 to calculate the average weight of a single component. This extends the manual calibration procedure presented in Section 4.2.2 by automatically determining the calibration quantity instead of requiring the operator to count and enter it manually. Because this procedure relies on both sensing channels operating simultaneously, automatic calibration is available only when both the vision and weight channels are functioning correctly. By eliminating the need for the operator to manually count and enter the calibration quantity, the procedure simplifies calibration while reducing the possibility of operator input error.

4.2.4 Sensor Health Evaluation

Rather than assuming that both sensing channels are always operational, the implementation continuously evaluates the health of the vision and weight channels independently before performing sensor fusion. This allows sensor failures to be identified separately from normal operating conditions.

For the vision channel, sensor health is determined from the successful acquisition of camera frames rather than by the number of detected parts. This distinction is important because an empty bin, a temporarily occluded camera, and a disconnected camera could all result in zero detected objects. By monitoring frame availability instead of detection results, the implementation distinguishes between these situations. To avoid reacting to isolated communication delays, the camera is considered unavailable only after 15 consecutive failed frame acquisitions.

The weight channel is monitored using a similar principle. Instead of checking the measured weight value, the implementation monitors the arrival of new MQTT messages from the ESP32. If no new weight measurement is received within three seconds, the weight channel is classified as unavailable. This approach avoids interpreting communication failures as valid inventory readings.

The health status produced by these independent checks is used by the adaptive fusion algorithm described in the following section to determine which sensing channels are available for inventory estimation.

4.2.5 Adaptive Sensor Fusion and Operating Modes

The final inventory estimate is produced by combining the outputs of the weight and vision channels using an adaptive sensor-fusion algorithm. Before performing sensor fusion, the implementation evaluates the health of both sensing channels as described in Section 4.2.4 and selects the appropriate operating mode according to the available sensor information. This enables the system to continue operating during partial sensor failures while taking advantage of both sensing modalities whenever they are available.

When both sensing channels are operational, the prototype operates in Fusion Mode. Under normal conditions, the fused inventory estimate is obtained by selecting the smaller of the weight-based and vision-based part counts. This conservative decision rule was adopted because it reduces the risk of overestimating the available inventory, which could lead to stock shortages remaining undetected during subsequent picking operations.

To improve robustness, the implementation also incorporates a stability override. When one sensing channel produces stable measurements while the other exhibits temporary fluctuations, the fusion algorithm temporarily favours the stable estimate instead of applying the default minimum rule. This prevents short-term disturbances, such as transient vision misdetections or unstable measurements, from introducing unnecessary fluctuations into the reported inventory state while allowing normal fusion to resume once both sensing channels become stable.

Based on the evaluated sensor health, the prototype operates in one of four operating modes:

- **Fusion Mode** - both sensing channels are available and the fused inventory estimate is produced.
- **Weight-Only Mode** - the vision channel is unavailable, and inventory estimation relies solely on the load-cell measurements.
- **Vision-Only Mode** - the weight channel is unavailable, and inventory estimation relies entirely on image processing.
- **System Failure Mode** - neither sensing channel is available, preventing reliable inventory estimation.

This operating strategy ensures that every possible combination of sensor availability is handled explicitly, allowing the prototype to continue operating whenever sufficient sensing information is available while clearly identifying situations in which reliable inventory monitoring cannot be performed.

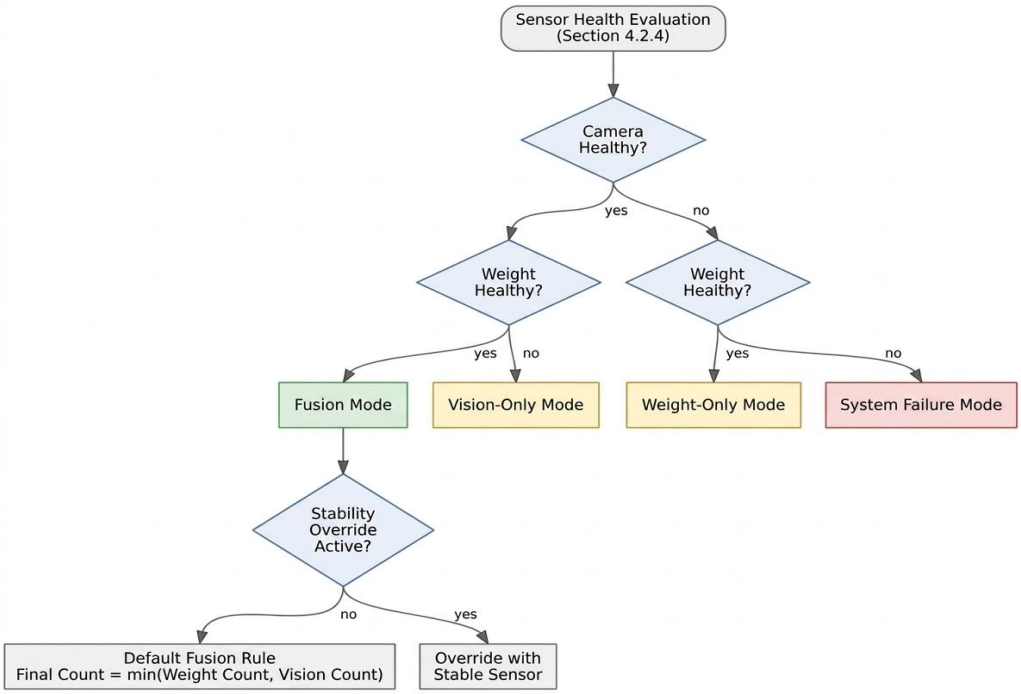


Figure 7. illustrates the operating-mode selection and adaptive sensor-fusion process implemented in the Phase 1 prototype.

During implementation, an important refinement was made to the fusion algorithm. An earlier version treated a vision count of zero as a special case and automatically trusted the weight-based estimate. Practical testing showed that this approach could produce incorrect inventory estimates when the camera correctly detected that all visible parts were damaged or otherwise classified as incorrect, resulting in zero valid parts despite material still being present in the bin. Consequently, the special-case rule was removed, and a zero vision count is now processed through the normal adaptive fusion algorithm like any other measurement. This modification improved the robustness of the fusion strategy by ensuring that fusion decisions are based on sensor health and validated measurements rather than assumptions about individual sensor outputs.

4.2.6 Hidden-Part Inference

One capability investigated specifically through sensor fusion is the detection of potentially hidden components within the monitored bin. This functionality is based on the observation that the load cell measures the total quantity of material present regardless of visibility, whereas the vision channel can identify only the components visible within the camera's field of view. By comparing these two independent sources of information, the prototype can infer the presence of components that are physically present but not directly visible.

The implementation compares the quantity estimated from the weight measurements with the total number of components already accounted for by the vision channel, including both correctly and incorrectly classified parts. When the weight measurement consistently indicates that more material is present than can be explained by the

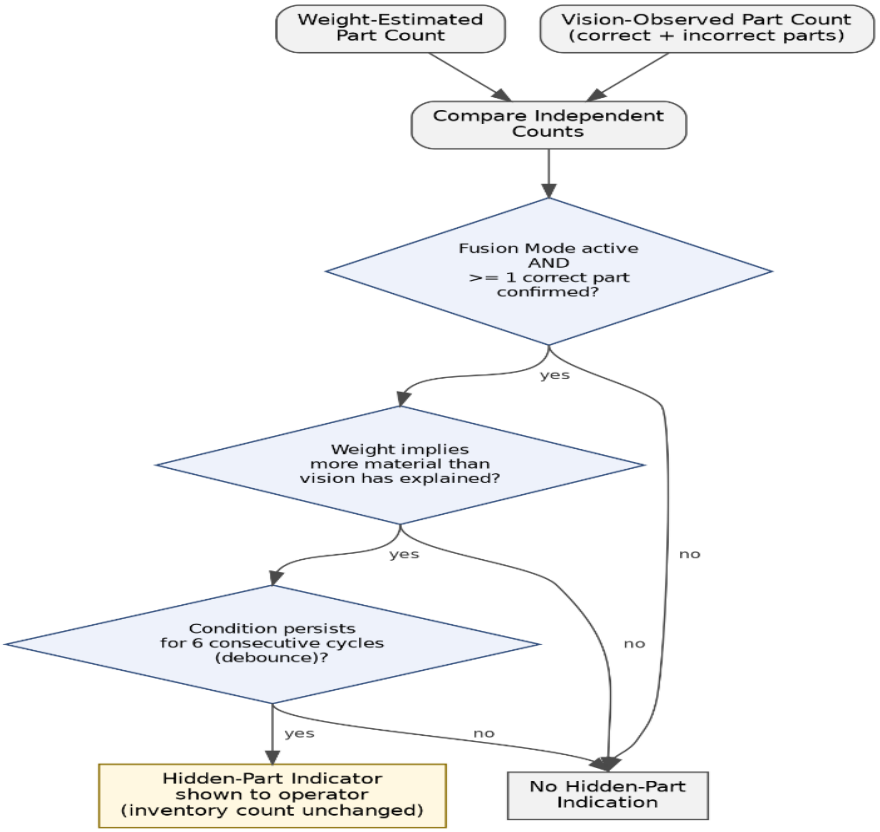


Figure 8. illustrates the hidden-part inference algorithm implemented in the Phase 1 prototype.

vision observations, the remaining quantity is interpreted as a potential hidden component. This evaluation is performed only when both sensing channels are operating in Fusion Mode and at least one correct component has been identified by the vision system. To improve robustness, the hidden-part indication is generated only after the condition persists for six consecutive processing cycles, reducing the likelihood of false detections caused by temporary sensor fluctuations.

Rather than modifying the reported inventory estimate, the inferred hidden-part information is presented as an additional status indicator for the operator. Since this capability depends on comparing independent measurements from both sensing modalities, it cannot be achieved using either the vision or weight sensor alone. Consequently, it represents a fusion-specific capability that directly supports the first research question by demonstrating that sensor fusion can provide information unavailable from either sensing modality independently.

4.2.7 Confidence and Agreement Metrics

To provide additional information about the quality of the reported inventory estimate, the prototype calculates two summary indicators during every processing cycle: a confidence score and a sensor agreement score. Both values are presented on the operator dashboard and recorded together with the inventory data for later analysis.

The confidence score represents the system's internal assessment of the reliability of the current inventory estimate. The calculation begins at 100% and predefined penalties are applied when conditions that may reduce the reliability of the estimate are detected. A penalty of 50% is applied if either the camera or the weight sensing channel becomes unavailable, while the detection of an incorrect part reduces the confidence by 20%. When both sensing channels are operating, an additional penalty is applied if the reported part counts differ, reducing the confidence by 5% for each part of disagreement up to a maximum penalty of 20%. This disagreement penalty is applied only when both sensing channels are available, preventing a sensor failure from being penalised twice. The final confidence score is limited to the range of 0-100%.

The sensor agreement score measures the consistency between the inventory estimates produced by the weight and vision channels. It is calculated from the difference between the two reported part counts and expressed as a percentage, with 100% indicating complete agreement and progressively lower values representing increasing disagreement between the sensing modalities. Since this metric compares only the outputs of the two sensing channels, it should not be interpreted as an accuracy measure against an independently verified ground truth. Throughout this thesis, it is therefore referred to as a sensor agreement score rather than an accuracy metric.

Together, these indicators provide additional information about the reliability and consistency of the reported inventory estimate while supporting the experimental evaluation presented in Chapter 5.

4.2.8 Operator Dashboard and Data Logging

The operator dashboard provides a real-time interface for monitoring the inventory state and interacting with the prototype. A separate dashboard server implemented using Flask and Socket.IO acts as the communication bridge between the MQTT-based sensing system and the web browser. The server maintains an in-memory representation of the latest system state, which is updated continuously as new MQTT messages are received from the sensing and fusion components.

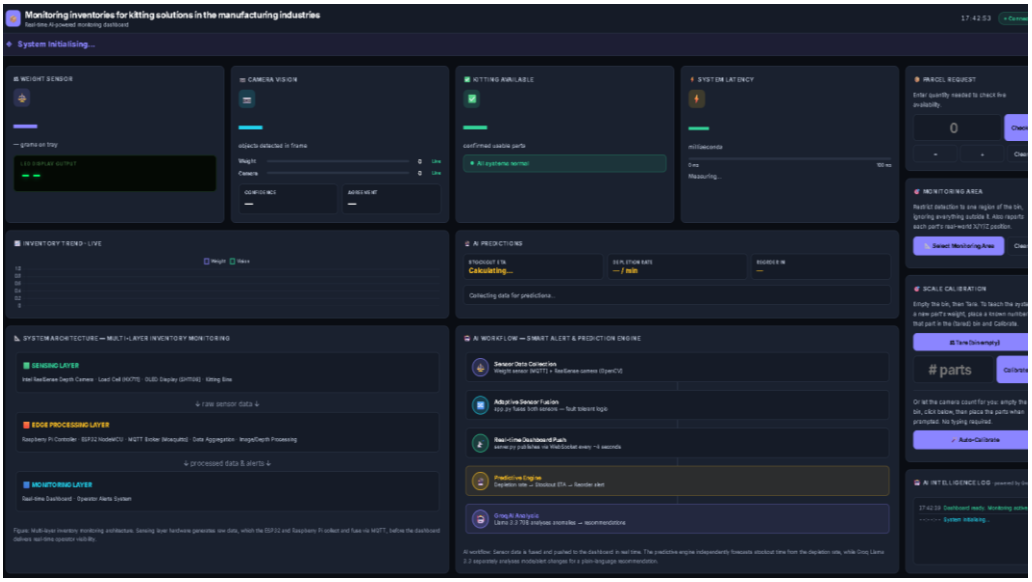


Figure 9. Operator dashboard implemented for the Phase 1 prototype.

To improve usability, the browser is not updated every time an MQTT message arrives. Instead, the dashboard server broadcasts the current system state at regular intervals of approximately four seconds. Earlier versions of the implementation transmitted updates for every incoming MQTT message, resulting in refresh rates that followed the camera processing cycle and produced unnecessary interface activity. Separating data acquisition from dashboard updates provided a more stable and readable operator interface while preserving the latest system state internally.

Figure 9 shows the operator dashboard implemented for the Phase 1 prototype. The dashboard presents the current inventory estimates from both sensing channels, the fused inventory estimate, confidence and sensor agreement indicators, operating mode, system alerts, latency information, inventory trend visualisation, calibration controls, monitoring-area selection, and additional operator support functions. These values are updated automatically throughout system operation, allowing the operator to monitor the inventory state without interacting directly with the sensing hardware.

Every processing cycle is also recorded automatically in a CSV log file. Each record includes the timestamp, weight-based and vision-based inventory estimates, fused inventory estimate, operating mode, confidence score, sensor agreement score, channel latencies, system alerts, incorrect-part detections, and hidden-part indications. This log provides a complete record of system behaviour throughout each experiment and forms the basis for the offline evaluation presented in Chapter 5.

Following each experiment, the recorded CSV data are processed using a separate offline analysis script, which computes summary statistics and generates the comparison tables and figures used for evaluating the proposed methodology. This separation between online monitoring and offline analysis allows the real-time inventory monitoring system to operate independently of the experimental evaluation process.

4.2.9 Implementation Challenges

During the implementation of the Phase 1 prototype, several practical engineering challenges were encountered that required modifications to the original design. Addressing these challenges improved both the robustness of the implementation and the reliability of the proposed inventory monitoring methodology.

One significant challenge involved handling camera disconnections while the system was running. If the Intel RealSense camera was unplugged or temporarily lost connection during operation, attempting to reconnect to the camera directly within the main processing loop caused that loop to block for several seconds while the reconnection was in progress. Since the same processing loop is responsible for detecting sensor failures, this behaviour delayed the system's own fault detection at exactly the moment when a rapid response was required. To overcome this problem, the reconnection procedure was implemented in a separate background thread, allowing the main processing loop to continue operating normally while reconnection attempts were performed independently. Furthermore, each reconnection establishes a completely new camera connection rather than attempting to reuse the previous one, since a physical USB disconnection may leave the original connection in an unrecoverable state. This approach enables the system to recover automatically from temporary camera failures without requiring manual intervention or restarting the application.

In addition to this improvement, several engineering refinements were introduced throughout the implementation process. These included the development of a robust sensor-health evaluation strategy capable of distinguishing genuine sensor failures from normal operating conditions, the refinement of the adaptive fusion algorithm following practical testing, the introduction of a debounced hidden-part detection mechanism, and the separation of continuous data acquisition from periodic dashboard updates to improve operator usability. These refinements are described in detail in Sections 4.2.4 - 4.2.8 and collectively contributed to improving the robustness, stability, and practical usability of the proposed inventory monitoring system.

The implementation process followed an iterative engineering approach in which practical observations during development were used to refine the system beyond its original design. Rather than simply integrating existing technologies, the final prototype evolved through continuous testing, analysis, and implementation improvements, resulting in a more reliable sensor-fusion inventory monitoring system for manufacturing kitting applications.

4.3 Phase 2 - Industrial Pilot - AI-Assisted Automated Pick-to-Light System

4.3.1 System Implementation Overview

The Phase 2 prototype was implemented as three main software components and deployed as a working pilot system at Binar Solutions AB rather than being tested only in a laboratory environment. The implementation follows the layered architecture introduced in Section 3.2, with sensing hardware communicating through MQTT with a central Node.js application layer, while the operator interface communicates with the application layer through Socket.IO. Unlike the Phase 1 prototype, where sensor processing, fusion logic, and dashboard communication were distributed across several Python components, the main decision-making logic in Phase 2 is concentrated within a single Node.js server application.

In this architecture, the AI assistant functions as a natural-language interface that interprets operator intent, not as the component responsible for inventory integrity: every inventory action it proposes is passed through a deterministic verification layer that checks current stock state and validates the requested quantity before any change is committed. This separation is discussed further in Section 5.5.

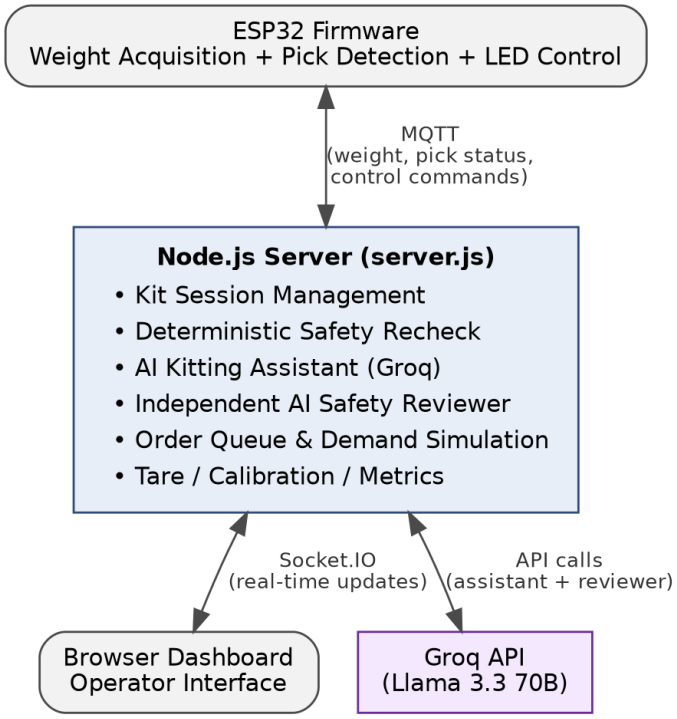


Figure 10. Software implementation of the Phase 2 Industrial Pick-to-Light prototype.

The Node.js server forms the central component of the implementation. It maintains the kit-session state described in Section 4.3.3, performs deterministic safety checks for every pick request regardless of how the request was initiated, manages the AI assistant and independent safety-review functions, handles the Order Queue, maintains tare and calibration state, calculates the performance metrics presented to the operator, and communicates the resulting system state to the browser-based dashboard. A separate firmware application runs on the ESP32 microcontroller and is responsible for the physical sensing functions, including load-cell measurement through the HX711 amplifier, pick detection, LED control, and MQTT communication with the Node.js server. The operator dashboard is implemented as a browser-based single-page application and does not communicate directly with the ESP32 or MQTT broker; instead, it receives system updates from the Node.js server through Socket.IO.

This separation keeps the sensing firmware focused on physical measurement and pick detection while concentrating the decision-making, safety logic, and AI-related functions within the application layer. As a result, additional functions such as the Order Queue and independent safety reviewer could be integrated without requiring changes to the underlying sensing and pick-detection firmware.

4.3.2 Weight-Based Pick Detection (ESP32 Firmware)

The physical pick-detection algorithm runs entirely on the ESP32 and can be triggered either by a start command sent from the server over MQTT or by pressing a physical button connected directly to the ESP32. When a pick begins, the firmware records the current weight reading as a baseline and turns on the bin's LED, providing the operator with a clear visual indication of which bin should be picked from. From this point onward, the firmware continuously monitors the load-cell measurement and publishes a weight update over MQTT every 300 ms(millisecond), allowing the dashboard to display the current bin weight during operation.

Both the baseline weight and the subsequent weight readings used for pick detection are obtained from the filtered weight signal rather than directly from the raw load-cell output. If a new reading differs from the previously smoothed value by more than 5 g (0.005 kg), the firmware treats the change as significant and adopts the new value immediately. If the difference is 5 g or smaller, the change is treated as sensor variation and the smoothed value is updated gradually using 70% of the previous value and 30% of the new measurement. This is the same threshold-based smoothing approach used in the Phase 1 prototype and described in Section 4.2.2. Applying the filtering before pick detection allows the algorithm to respond promptly to significant weight changes while reducing the influence of minor measurement fluctuations.

When the measured weight decreases by at least 5 g (0.005 kg) relative to the recorded baseline, the firmware does not immediately confirm the pick. Instead, it waits for one second before taking the confirmation measurement, allowing the weight reading to settle after the operator's interaction with the bin. This confirmation reading is checked against the same 5 g threshold; if the drop no longer meets it once the reading has settled, the event is treated as a false alarm and the firmware silently resumes monitoring rather than reporting a pick. The measured weight reduction is then divided by the calibrated unit weight of a single component to estimate the number of picked components as a whole-number quantity.

To determine whether the measured reduction is consistent with a valid pick, the firmware also evaluates the deviation between the measured weight reduction and the nearest expected multiple of the calibrated unit weight. If this deviation is within the implemented tolerance of 30%, the pick is classified as confirmed. If the deviation exceeds this tolerance, the pick is classified as uncertain, and the operator is expected to inspect the bin manually. In both cases, the LED is switched off and the resulting pick status and quantity are published to the Node.js server through MQTT.

The detection process is intentionally based on the physical weight change rather than on a button press or barcode scan, because these alternative inputs cannot independently verify how much material was physically removed from the bin. The load-cell measurement therefore provides the physical verification mechanism on which the subsequent deterministic safety checks in the Node.js application can operate.

4.3.3 Server-Side Deterministic Safety Layer

A central design decision in this phase is that all three ways of triggering a pick - the AI assistant, the manual dashboard button, and the Order Queue - write into the same shared internal object on the server, referred to as a kit session. This object tracks the active bin, the required quantity, the quantity already picked, and, where applicable, the queued order that initiated the operation. Because all three trigger paths use the same underlying session state, the logic for determining whether a kit still requires additional picks, has been completed, or has been over-picked is implemented only once and applied consistently regardless of how the pick was initiated.

Before a pick request is allowed to reach the ESP32, the server independently verifies the requested operation using the kit definition and the latest settled inventory value received from the sensing system. The server does not rely solely on the quantity proposed by the AI assistant. Rather than comparing the AI's proposed quantity against this independently calculated value and rejecting on a mismatch, the server recalculates the correct quantity itself and uses that value directly, overriding whatever the AI proposed; the AI's original figure is only used as a fallback if the server's own calculation is not usable. If the target bin is unavailable, if insufficient stock is available, or if the

requested kit has already been completed, the server rejects the operation outright rather than sending the pick command to the hardware.

This deterministic verification layer provides a separation between request generation and action authorization. The AI assistant, dashboard, and Order Queue can propose or initiate a pick, but the server-side safety logic remains responsible for deciding whether that action is permitted. This is particularly important for the AI-assisted interaction, since a language model may occasionally produce an incorrect quantity or interpretation even when its response appears plausible. By independently validating and, where necessary, correcting each request against the system state before allowing hardware execution, the implementation prevents such requests from directly controlling the physical picking process.

4.3.4 AI Kitting Assistant

The AI Kitting Assistant provides a natural-language interface through which an operator can request a kit, check the current stock level, or cancel an ongoing request. The assistant is implemented using Groq's Llama 3.3 70B model and is integrated with the Node.js server through tool calling. A trigger_pick function is made available to the model with four structured parameters: the target bin, a quantity_expected value representing how many units should be removed in that specific pick call, a total_quantity_for_kit value representing the full size of the overall kit request, and a kit_label. Separating the per-call quantity from the total kit size allows a single kit request to be fulfilled across more than one physical pick when necessary, while the server-side safety layer described in Section 4.3.3 uses the total quantity to determine when the kit is actually complete.

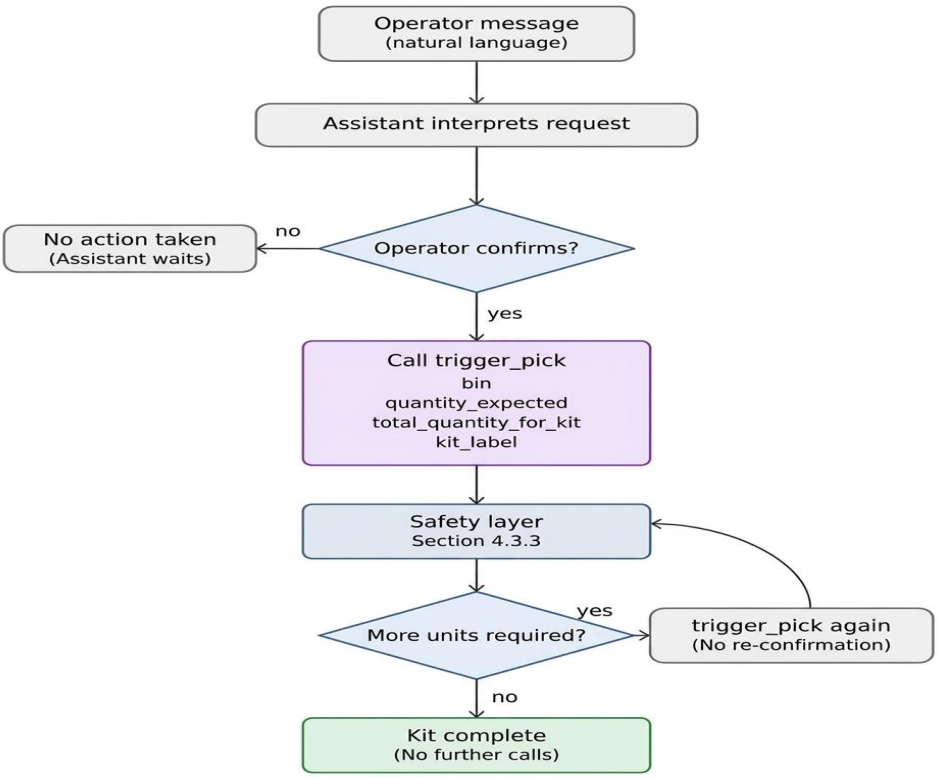


Figure 11. Workflow of the AI-assisted kitting request and multi-unit pick process.

The assistant uses a server-side kit catalogue containing the available kit types and the number of components required for each kit. The same catalogue is also used by the Order Queue described in Section 4.3.6, ensuring that kit definitions are maintained in one location rather than being duplicated across different parts of the application. Consequently, adding or modifying a kit requires updating the corresponding catalogue entry rather than changing the logic of the individual system components.

A two-step confirmation process is implemented to prevent an unintentional natural-language request from immediately initiating a physical pick. When a new kit is requested, the assistant first presents its interpretation of the request and asks the operator for confirmation without calling trigger_pick. The pick operation is initiated only after the operator confirms the request. If the requested kit contains multiple units, subsequent units are handled through the same kit session and can be triggered automatically as the previous picks are successfully confirmed by the sensing system, without requiring the operator to confirm each individual unit again.

During development, an issue was identified in which the assistant could refer to an outdated stock value mentioned earlier in the conversation rather than the latest value reported by the sensing system. Since the language model does not inherently distinguish which previously mentioned value represents the current inventory state, the implementation was modified to insert the current stock value into the assistant's context immediately before each response. The accompanying instruction explicitly directs the model to use this latest value and disregard older stock values from the conversation. This ensures that stock-related responses are based on the current system state rather than potentially outdated conversational information.

4.3.5 Independent AI Safety Reviewer

In addition to the AI Kitting Assistant and the deterministic server-side safety layer described in Section 4.3.3, the implementation includes a separate AI-based review step for AI-initiated pick requests. After the assistant proposes a pick, a second separate invocation of the language model is used as a safety reviewer rather than as a conversational assistant. The reviewer is given the operator's original message together with the proposed pick and is explicitly instructed to evaluate the request critically. The pick is allowed to proceed only if the reviewer explicitly approves the proposed action.

The reviewer was introduced to provide a second reasoning step rather than relying solely on the assistant that originally interpreted the operator's request. In particular, providing the original operator message to the reviewer allows it to judge intent directly: it is explicitly instructed to reject a proposed pick if the operator's message was actually a plain question about stock rather than a genuine request to prepare a kit, even if the proposed quantity would otherwise look reasonable, and to separately confirm that the proposed quantity is a valid, positive whole number that does not exceed current stock. The reviewer therefore operates with a different objective and prompt context from the assistant that generated the proposed action. This separation provides an additional opportunity to identify an incorrect interpretation before the request reaches the physical system.

If the call to the reviewer model itself fails, for example due to a network or API error, the implementation does not silently block the pick. Instead, it defaults to approving the request while logging the failure, on the reasoning that a failure in the review step itself should not be allowed to halt the whole system. This failure is still recorded so that it can be noticed and investigated separately from a normal rejection.

The AI review does not replace the deterministic safety checks described in Section 4.3.3. Instead, the two mechanisms serve different purposes: the deterministic layer verifies the proposed operation against the known system state, kit definition, and available stock, while the AI reviewer provides an additional semantic check of the operator's request and the proposed pick. A request must therefore pass the applicable safety checks before the hardware is permitted to execute the operation.

For operations initiated directly through the Order Queue's Start Kit control, this additional language-model review is not performed because the operator has explicitly selected the order rather than relying on an AI interpretation of a natural-language request. Nevertheless, the deterministic safety layer remains active for these manually initiated operations, ensuring that the physical pick is still checked against the current system state before execution.

4.3.6 Order Queue and Demand-Planning Projection

The Order Queue provides a mechanism for an operator to define and manage multiple kit orders that are intended to be processed sequentially. In addition to storing the orders, the implementation performs a demand-planning projection to evaluate whether the complete queue can be fulfilled using the currently available inventory. The projection starts from the latest measured stock value and processes the queued orders in sequence. For each order, the required quantity is subtracted from a running stock balance before the next order is evaluated. Consequently, an order is assessed against the stock remaining after all preceding orders rather than against the original stock value. This allows shortages that would only occur later in the production sequence to be identified when the queue is created, before the corresponding order is physically started.

When a shortage is identified, the system reports the affected order together with the quantity missing. Where email notifications are enabled, the notification also includes a queue-level summary showing the total quantity required to fulfil all queued orders and the total projected shortfall. This provides an early indication of material requirements and can support procurement or replenishment decisions before production reaches the affected order.

Each queued order progresses through explicit states during execution. An order initially remains pending and becomes active when its Start Kit control is used and physical picking begins. Following completion of the required picks, the order is marked as completed. If the physical picking result indicates that the order has been overpicked, the order instead enters a needs_review state, requiring operator attention before the queue can treat the order as normally completed. This state-based structure provides a clear representation of the current status of each order and prevents an overpicked order from being treated as successfully completed.

The queue also provides an optional auto-continue function, which is disabled by default. When enabled, the system can automatically start the next eligible queued order after the current order has been completed, supporting continuous operation without requiring the operator to manually start every order. An order is only started automatically if the demand-planning projection indicates that sufficient stock is available for it; an order that would result in a projected shortage is left in the pending state rather than being started automatically, requiring operator attention. This design limits automatic chaining of physical actions and ensures that unattended sequential operation is enabled only when explicitly selected by the operator.

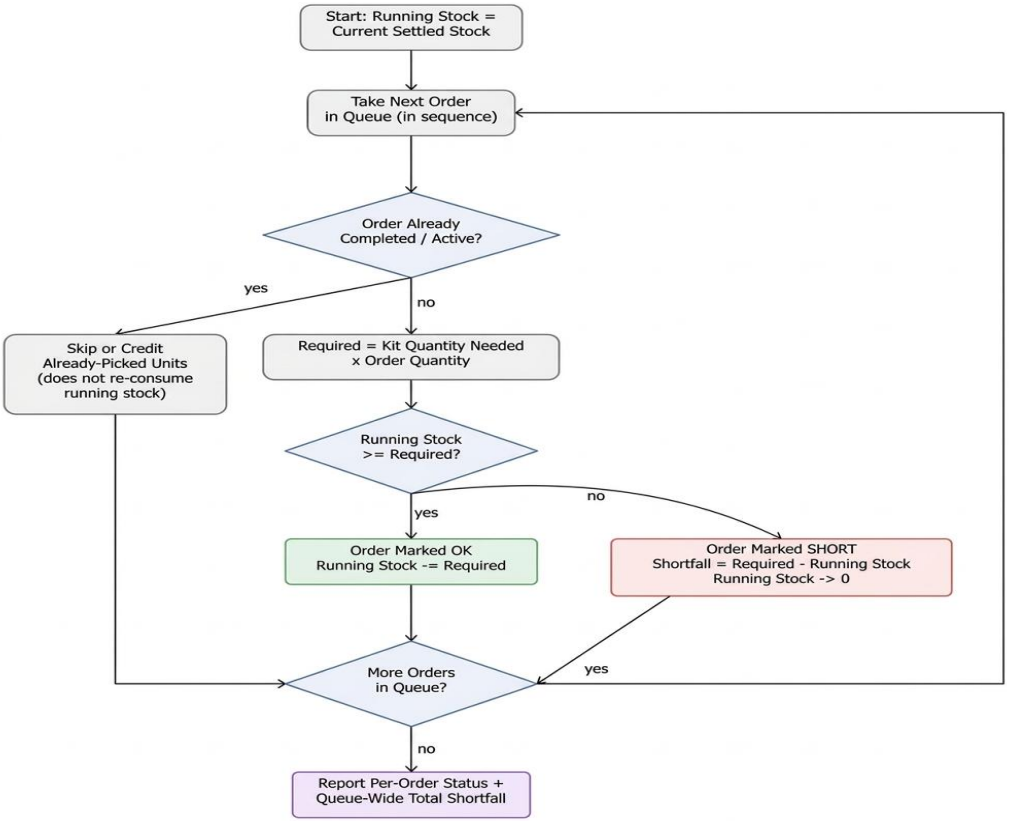


Figure 12. Demand-planning projection algorithm for the Order Queue.

This implementation therefore extends the kitting system from single-order execution to sequential demand planning, allowing both current stock availability and the cumulative requirements of future orders to be considered before physical production begins.

4.3.7 Tare and Unit-Weight Calibration

Two related but distinct operations are implemented to maintain the accuracy of the weight-based inventory measurement: tare and unit-weight calibration. The tare operation establishes the zero reference of the load-cell measurement and is intended to be performed with an empty bin. This compensates for changes in the mechanical setup that may occur over time, such as repositioning the bin or changing the surface on which it is placed. Unit-weight calibration, in contrast, determines the average weight of one component. Instead of using a fixed unit weight in the firmware, the operator places a known number of identical components on the bin and provides this quantity to the system. The measured total weight is then divided by the known quantity to obtain the calibrated weight of a single component, which is subsequently used when converting weight changes into component quantities.

Both operations can be initiated through the operator interface and are handled by the ESP32 firmware. The resulting tare offset and calibrated unit weight are stored in the ESP32's non-volatile flash memory, allowing the configured values to be retained across power cycles and normal firmware updates without requiring the calibration procedure to be repeated each time the system is restarted. The stored values are only lost if the corresponding persistent storage is explicitly erased.

Tare and calibration events are deliberately excluded from the system's normal stock-tracking logic because neither operation represents a genuine inventory change.

A tare operation establishes a measurement reference and therefore does not indicate that the bin has become empty from an inventory perspective. Similarly, the quantity of components placed on the bin during calibration represents a calibration reference rather than a production inventory update. During development, treating these measurements as normal stock changes resulted in two identifiable problems: a false low-stock condition could be generated immediately after a tare operation, and a calibration measurement could overwrite the actual inventory quantity. The stock-tracking logic was therefore modified to explicitly distinguish tare and calibration events from normal inventory measurements, preventing these operations from affecting the recorded stock level.

4.3.8 Inventory Accuracy, Predictions, and System Performance Metrics

Inventory accuracy is calculated as the percentage of physically detected picks that are successfully and correctly confirmed by the weight-sensing system. Cancelled or blocked requests are excluded from this calculation, since no physical pick measurement takes place for these operations. A pick is classified as inaccurate under two conditions: first, when the measured weight reduction does not correspond to a valid whole-number quantity of components, resulting in an uncertain pick; and second, when the measured quantity is physically valid but exceeds the quantity required by the active kit or order, resulting in an over-pick. This condition was explicitly incorporated into the implementation so that inventory accuracy reflects the complete pick-to-light outcome, including both reliable physical sensing and correct fulfilment of the requested quantity, rather than evaluating the sensing measurement in isolation.

The system also provides a stockout prediction based on the recent history of measured inventory levels. The prediction includes an estimated depletion rate and an estimated time until the available stock reaches zero. These values are calculated directly from the recorded stock history using deterministic numerical calculations and are not generated by the AI assistant. This separation ensures that numerical predictions related to inventory depletion are based on measured system data rather than on language-model reasoning.

Additional metrics are recorded to evaluate the interaction between the AI assistant, the sensing system, and the operator interface. Tool-call correctness measures how often an AI-generated trigger_pick request successfully passes the applicable deterministic and AI-based safety checks described in Sections 4.3.3 and 4.3.5. Pick latency is measured as the time between detection of the physical weight drop associated with a pick and the corresponding update becoming available on the dashboard. Assistant latency measures the time between an operator sending a message to the AI assistant and receiving the assistant's response. These metrics provide quantitative measures of the system's operational behaviour and correspond to the evaluation criteria used in Chapter 5.

The implementation also includes a smart recalibration recommendation that uses the existing inventory-accuracy measurements to identify a potential reduction in measurement reliability. The system continuously compares the accuracy of the ten most recent eligible picks with the overall accuracy recorded during the current session. When at least five recent picks are available and the recent accuracy is at least 15 percentage points lower than the overall session accuracy, the dashboard recommends that the operator re-tare or recalibrate the bin. The corresponding tare and calibration controls are presented together with the recommendation so that the operator can take corrective action directly.

This feature does not introduce an additional sensing method or independent accuracy measurement. Instead, it connects the existing accuracy monitoring, tare, and calibration functions into a single operator-support mechanism. A reduction in recent pick accuracy is treated as an indication that the weight measurement may require attention, rather than as proof of a specific hardware fault. The recommendation therefore provides an early indication of possible measurement drift while leaving the decision to perform tare or calibration to the operator.

4.3.9 Operator Dashboard and Bilingual Support

The operator dashboard was implemented as a single-page browser application that receives live system updates from the Node.js server through Socket.IO, without requiring manual page refreshes or periodic polling. The dashboard provides a central interface for monitoring and controlling the system. It presents live values for current stock, total units picked, the active kit, the number of uncertain picks, and inventory accuracy. When the stock level falls to or below a configured low-stock threshold, the dashboard visually indicates this through the stock display and a toast notification, alerting the operator immediately without requiring them to check the numeric value themselves. The bin interface provides manual pick controls, a continuously updated weight reading, and the tare and calibration controls described in Section 4.3.7. Additional dashboard functions include a live chart of stock and weight history, the stockout predictions described in Section 4.3.8, and a separate stock-availability check that allows the operator to quickly verify whether sufficient inventory is available for a specified quantity. The dashboard also displays the progress of the currently active kit independently of how the kit was initiated, a rolling activity feed containing system events, the Order Queue interface, the AI Kitting Assistant with text and voice interaction, and a system architecture panel reflecting the implemented system structure.

The dashboard and AI-assisted interaction were implemented with full English and Swedish language support. Language selection is applied dynamically to the interface, allowing the operator to switch languages without reloading the application. The AI assistant generates its responses directly in the selected language rather than translating a response generated in another language. Voice interaction similarly supports both languages for speech input and spoken responses. Generated outputs, including email alerts and the session report described below, are also produced in the currently selected language. Kit and component identifiers, such as Kit A, remain unchanged between languages because they represent fixed identifiers, while the surrounding descriptive text is translated according to the selected language.

Two additional operator-facing functions were implemented within the dashboard. First, a one-click session report generates a printable summary containing the performance metrics described in Section 4.3.8, the history of the Order Queue, and recent system activity. The report uses the information stored by the server at the time it is requested and is generated in the language currently selected by the operator. Second, optional email notifications can be enabled through a configuration setting. When enabled, notifications are generated for three conditions: a stock level falling to or below the configured low-stock threshold, which defaults to 2 units but can be changed through the system configuration, an operator requesting additional parts, and a shortage identified by the demand-planning projection described in Section 4.3.6. The email functionality is disabled by default during testing, allowing the notification mechanism to remain available without unintentionally sending messages during development and experimental operation.

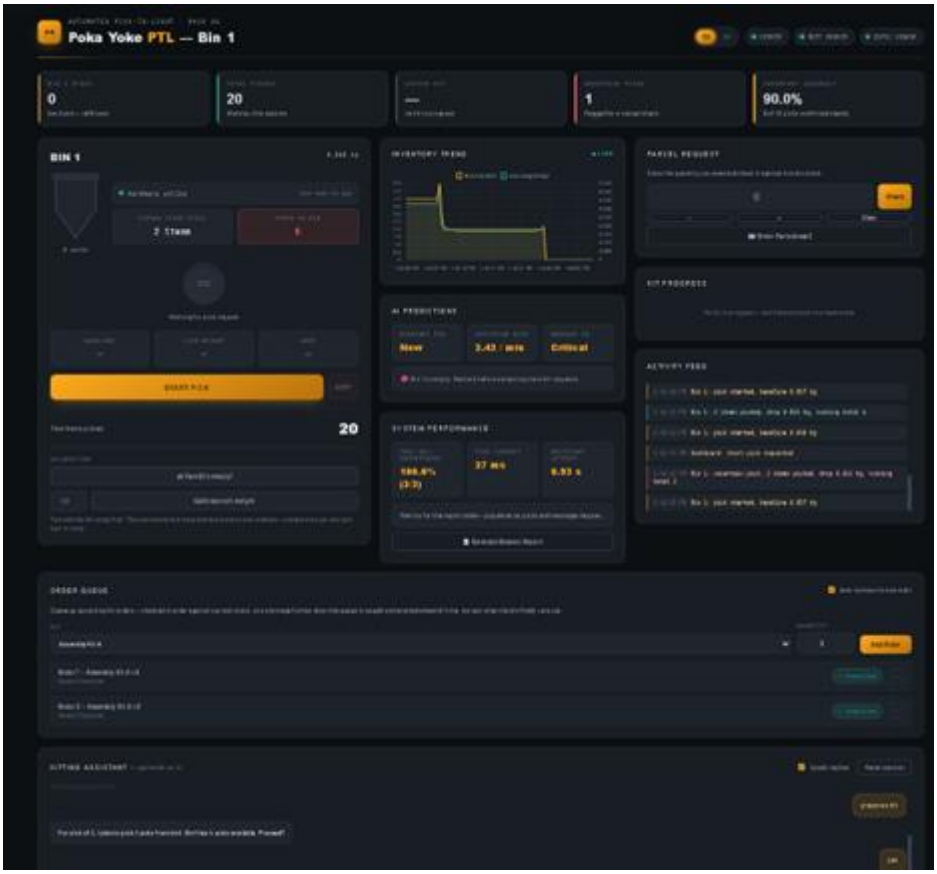


Figure 13. Operator dashboard implemented for the Phase 2 prototype.

4.3.10 Implementation Challenges

Several implementation challenges were identified and resolved during the development and testing of the Phase 2 prototype. These issues provided important feedback on the interaction between the physical sensing system, the server-side safety logic, the AI components, and the Order Queue. Rather than treating these problems as isolated software errors, the implementation was revised to ensure that the underlying state and safety logic remained consistent across different operating conditions.

One issue occurred when a kit was over-picked. In this situation, the bin's LED remained active and the AI assistant attempted to initiate an additional pick even though the kit had already reached or exceeded its required quantity. The deterministic safety logic correctly calculated the remaining required quantity as zero, but a separate fallback mechanism interpreted any value below one as an invalid quantity and replaced it with the originally requested quantity. This caused the valid zero value to be unintentionally overwritten. The problem was resolved by explicitly distinguishing between a missing or invalid quantity and a valid quantity of zero. Consequently, a completed kit can now correctly produce a remaining requirement of zero without triggering another physical pick.

A second issue occurred when an over-pick was corrected by returning the excess components to the bin. Although the physical inventory had been restored, the corresponding order could still display a shortfall. The cause was that returning the order to a pending state caused the system to recalculate its complete original requirement, without accounting for the quantity that had already been physically picked. This was re-

solved by introducing the dedicated needs_review state. Orders in this state are excluded from being treated as new demand, while completion requires an explicit operator confirmation. The confirmation is additionally checked against the measured stock increase, so that an order is not marked as completed solely because the operator pressed a confirmation control.

A third issue concerned natural-language requests that were informational rather than operational. During development, an operator could ask a question such as how many components were currently available in a bin, and the AI assistant could incorrectly interpret the message as a request to initiate a physical pick. The underlying cause was that the assistant's instructions described how to process a kit request but did not sufficiently distinguish a genuine pick request from a general stock-information question. This was addressed by explicitly defining informational questions as non-pick requests. In addition, the independent safety reviewer described in Section 4.3.5 receives the operator's original message and is instructed to reject a proposed pick when the original message is only an informational query. The combination of the revised assistant instructions and the independent review therefore provides two separate opportunities to prevent this type of incorrect interpretation from reaching the physical system.

These challenges demonstrated that reliable operation depended not only on individual sensor or software functions, but also on maintaining consistent state information between the physical picking process, server-side decision logic, AI interaction, and order management. The resulting corrections strengthened the separation between request interpretation, deterministic validation, physical execution, and order completion, forming the basis for the experimental evaluation presented in Chapter 5.

4.4 Phase 3: Multi-Bin Simulation

4.4.1 System Implementation Overview

The Phase 3 prototype extends the single-bin pick-to-light concept developed in Phase 2 to a multi-bin configuration with three simultaneously monitored bins. The implementation was developed and tested entirely in the Wokwi online simulation environment using an ESP32-C6 rather than physical hardware. Consequently, the firmware, communication, control logic, and timing behaviour are executed by the simulated microcontroller, while the physical sensing process, component movements, and operator interactions are represented within the simulation. This distinction is important when interpreting the results of this phase, since the implementation demonstrates the behaviour and scalability of the software and communication architecture but does not provide physical measurements from real load cells or components. The use of physical hardware for further validation is therefore considered future work and is discussed in Section 6.

The Phase 3 implementation retains the separation between sensing and control, messaging, and presentation introduced in Phase 2. The sensing and control layer is implemented on the ESP32-C6 firmware, which manages the three simulated bins and their corresponding pick-detection and control logic. The messaging layer uses MQTT through a shared public broker to exchange commands and status information between the firmware and dashboard. The presentation layer consists of a browser-based dashboard implemented as a single HTML file. Unlike Phase 2, where the dashboard communicated with a central Node.js application, the Phase 3 dashboard connects directly to the MQTT broker through a secure WebSocket connection. It can therefore both

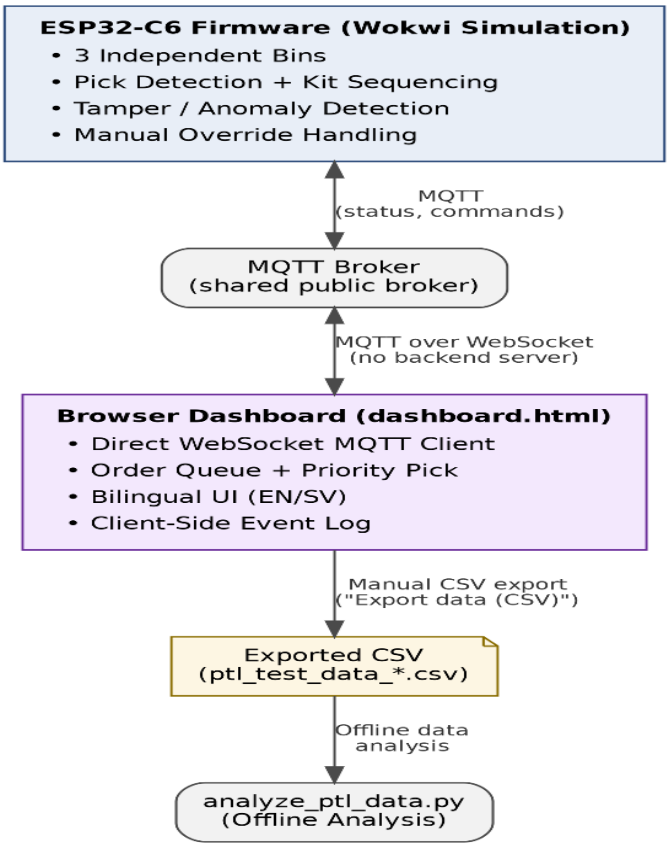


Figure 14. Software architecture and experimental data flow of the Phase 3 multi-bin prototype.

publish control commands and subscribe to system-status topics without requiring an intermediate backend server. Unlike Phase 1, where experiment data was logged automatically to a CSV file by a backend process, Phase 3 maintains the event log directly in the browser. The operator can export this client-side log as a CSV file using the dashboard's data-export function, after which the recorded data can be analysed offline.

Each bin is handled independently by the firmware while using the same underlying control structure. A kit-start request is received through the common MQTT communication channel and processed by the ESP32-C6 regardless of whether the request originated from a physical pushbutton, a manual dashboard control, or the dashboard's Order Queue. This separation between the command source and the firmware execution logic allows the same hardware-side behaviour to be reused across different control interfaces. It also allows additional dashboard functionality, such as the Order Queue described in Section 4.4.7, to be introduced without modifying the underlying pick-detection logic of the firmware.

4.4.2 Multi-Bin Weight Sensing and Calibration

Each of the three bins is represented in the ESP32-C6 firmware by an independent data structure containing its load-cell interface, current smoothed weight, baseline weight recorded when the bin is armed, operating mode, number of components confirmed as removed during the current pick session, number of components remaining to complete the active kit, and a flag indicating whether the bin is under manual dashboard control rather than using its live sensor value. The three bin structures are stored in an array, allowing the same sensing, calibration, and pick-detection functions to operate

on any bin using its index instead of requiring separate implementations for each bin. This provides a consistent software structure while keeping the state and measurements of the three bins independent.

The raw weight measurements are passed through an adaptive smoothing filter before being used by the pick-detection logic. The filtering approach follows the same general principle used in the weight-sensing channels of Phases 1 and 2. If the difference between a new reading and the previously smoothed value is small, the change is treated as sensor noise and the value is updated gradually using 70% of the previous smoothed value and 30% of the new reading. If the difference exceeds the defined significance threshold, the change is treated as a genuine physical event and the new reading is adopted immediately. In Phase 3, a change greater than 50 g is treated as significant, while values below 10 g are set to zero. These thresholds are higher than the corresponding 5 g and 2 g thresholds used in Phases 1 and 2, reflecting the characteristics of the simulated load-cell signals used in the Wokwi implementation while retaining the same underlying adaptive filtering principle.

Because the three simulated load cells in Wokwi have independently assigned sensitivities, a single calibration factor shared between all bins did not provide sufficiently consistent measurements. Calibration is therefore performed separately for each bin through a serial command that specifies the bin number and a known reference weight. The firmware uses the measured sensor response together with this reference value to determine the scale factor for the selected bin. Each bin consequently maintains its own calibration rather than relying on a common conversion factor. Unlike Phases 1 and 2, where the operator can calibrate the weight of an individual component through the dashboard, the component weight in Phase 3 is fixed in the firmware for each bin, set to 100 g for all three bins in the current implementation. This value is a compile-time constant rather than an operator-adjustable setting, and can only be changed by editing the firmware source code and re-uploading it. This is appropriate for the simulated environment used in this phase, where the components are represented digitally rather than as physically variable parts, though making this value configurable at runtime would be a natural extension if this implementation were adapted for physical hardware.

4.4.3 Pick Detection and Kit Sequencing

The core pick-detection logic is executed once per main-loop iteration for each bin that is currently armed. Four checks are evaluated in sequence. First, an inactivity timeout monitors the confirmed removed-part count. If no change is detected for 60 seconds, the active bin session is cancelled and, if the bin is currently part of a kit, the corresponding kit operation is cancelled as well. The timer is reset whenever genuine picking activity is detected, making it an inactivity timeout rather than a fixed deadline. Second, the firmware checks for an increase in weight. If the weight of an armed bin increases beyond the configured anomaly threshold, the event is interpreted as material being added rather than removed, and an anomaly notification is published. Third, the number of removed parts is estimated from the difference between the recorded baseline weight and the current smoothed weight. This weight difference is divided by the fixed component weight assigned to the bin and rounded to the nearest whole number. Finally, the estimated removal count is debounced using a one-second confirmation period. The confirmation timer restarts whenever the estimated removed count changes, and the value is accepted only when it remains unchanged for the full one-second period. This prevents short-lived disturbances, such as briefly bumping or moving the bin, from being immediately interpreted as a completed pick.

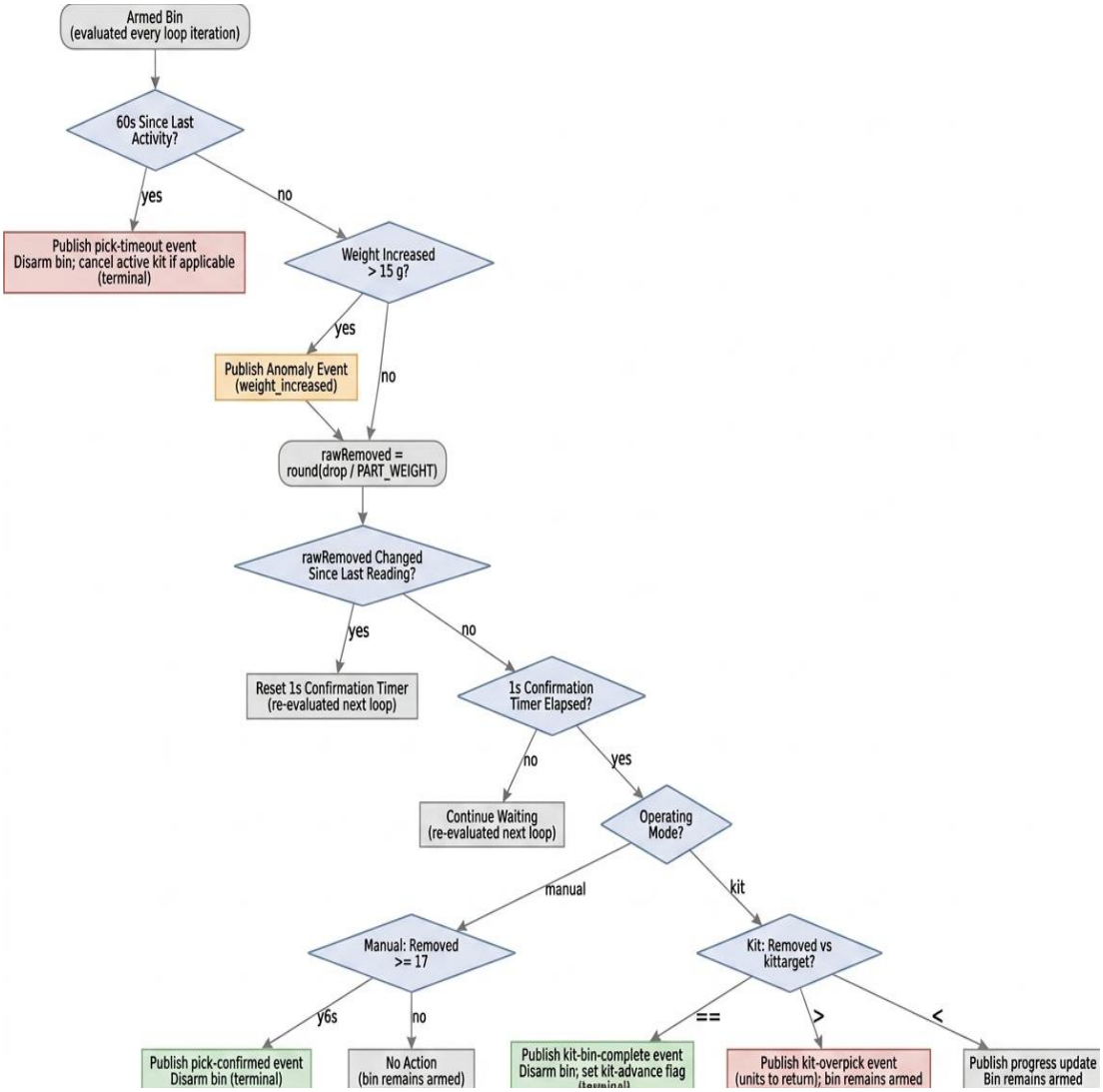


Figure 15. Phase 3 pick-detection and kit-sequencing logic implemented in the ESP32-C6 firmware.

Once a removal has been confirmed, the subsequent behaviour depends on the current operating mode. In manual mode, any confirmed removal of at least one component completes the current bin session. A pick-confirmed event is published and the bin is disarmed. In kit mode, the confirmed removal count is instead evaluated against the required quantity for the current kit stage. If the required quantity is reached exactly, the current bin is completed and the kit sequence advances to the next bin. If more components than required have been removed, an over-pick event is published specifying the number of components that must be returned, while the bin remains armed so that the correction can be detected. If fewer components than required have been removed, the system publishes a progress update and continues monitoring the same bin. An incomplete pick is therefore treated as normal progress, whereas an over-pick is treated as an explicit anomaly requiring correction.

The transition between bins within a kit is intentionally separated from the pick-detection function itself. During early development, advancing the kit directly from within the pick-detection routine resulted in a memory-corruption crash during rapid kit transitions. The cause was traced to several large temporary buffers becoming nested through multiple function calls, increasing stack usage during the transition. Rather than

attempting to resolve this only by reducing the immediate stack usage, the sequencing logic was reorganised. The pick-detection function now sets a dedicated flag indicating that the kit should advance, while the main loop processes this request afterwards and performs the transition separately. This keeps the call stack shallower during kit transitions and separates event detection from sequence management.

The resulting kit-sequencing mechanism operates as a strict one-bin-at-a-time state machine. A kit can only advance after the currently active bin has reached its required quantity, and the transition is handled by the main-loop sequencing logic before the next bin is armed. There is therefore no normal execution path in the implemented state machine that allows two bins belonging to the same kit to be armed simultaneously. This structure provides deterministic control over the order in which the three bins participate in a kit and ensures that each bin must complete its assigned stage before the sequence proceeds.

4.4.4 Tamper and Anomaly Detection

The firmware implements two complementary anomaly-detection mechanisms that operate according to the current state of each bin. The first is the weight-gain check described in Section 4.4.3, which is applied while a bin is armed and actively involved in a pick. If the measured weight increases beyond the defined anomaly threshold instead of decreasing as expected during a removal, the system interprets this as an unexpected addition of material and publishes an anomaly event. This check uses the same 15 g threshold as the idle-bin check described below, so the sensitivity to an unexpected weight change is identical whether or not the bin happens to be armed at the time.

A second check is applied continuously to bins that are not armed. For each idle bin, the firmware maintains the most recently recorded stable idle weight as a reference. If the live weight deviates from this reference by more than 15 g, an idle-bin anomaly event is published. The reference value is then immediately updated to the new weight, ensuring that the same settled change is reported once rather than generating repeated anomaly events during subsequent loop iterations.

The idle-bin check intentionally does not attempt to determine the cause of the weight change. Consequently, a legitimate restocking operation performed outside a formal picking session and an unexpected physical disturbance are treated in the same way. This conservative approach was chosen because the monitoring system cannot reliably infer operator intent from weight measurements alone. Reporting an unexpected change for further attention is therefore preferred to attempting to classify the event automatically and potentially overlooking a genuine tampering, handling, or sensor-related anomaly.

4.4.5 MQTT Command Handling and Manual Override Tools

All incoming MQTT commands are processed through a central command-handling function that matches each received topic against a predefined set of topic strings. These topic strings are constructed once during firmware initialisation and retained for subsequent message processing. This approach was adopted after repeated reconstruction of the topic strings during message handling was found to cause gradual memory consumption over time. Centralising the command dispatch also provides a consistent interface for the different control functions implemented in the firmware.

The command handler supports the main system operations, including starting a kit, cancelling the active kit, stopping and resuming the system, as well as a set of manual

override commands for individual bins. The override commands allow the dashboard to modify a bin's simulated weight or confirmed removed-part count directly. They were introduced primarily as controlled testing mechanisms, allowing different sensor and pick-detection conditions to be reproduced without manually adding or removing simulated components in the Wokwi environment.

The manual overrides are integrated with the normal pick-detection pipeline rather than being processed through a separate testing-only logic path. The pick-detection function operates on the bin's current smoothed weight and confirmed removal state without distinguishing how those values were produced. Consequently, a weight change generated through an override is processed by the same detection, debouncing, and kit-sequencing logic as a corresponding change originating from the simulated sensor. This allows the manual tools to test the complete detection and confirmation pipeline, including anomaly detection and kit sequencing, rather than testing only the MQTT communication or dashboard interface independently.

4.4.6 Order Queue and Priority Pick

The Order Queue maintains an ordered collection of kit orders, with each order storing the required quantity for each of the three bins, its current status, and the bins that have already been completed. When an order is added, its requirements are immediately compared with the current stock and an advisory shortage warning is presented if insufficient stock is detected; however, the warning does not prevent the order from being queued. Two separate stock-checking mechanisms are used for different purposes. The first evaluates the remaining requirements of an order against the latest available stock when the operator attempts to start it and acts as the operational safety check for that action. The second performs a forward-looking demand-planning projection across the queued orders in their execution order. It maintains a running stock balance for each bin, allowing the projected demand of each subsequent order to account for the requirements of all preceding orders. The running balance is allowed to become negative so that the magnitude of a projected shortage can be retained and reported. This projection is advisory and does not prevent an order from being started; the actual decision to permit a pick is instead made when the order is started using the current stock check. The same running-balance approach is also used by the Order Queue implemented in Phase 2, as described in Section 4.3.6.

A Priority Pick function allows the operator to interrupt the currently active order when another queued order requires more urgent attention. Selecting a different order and confirming the priority action does not start the new order immediately. Instead, the dashboard first requests cancellation of the currently active kit and records the order selected for priority execution. The new order is started only after the firmware has confirmed that the active kit has been cancelled. At that point, the interrupted order is returned to the pending state while retaining a record of the bins that had already been completed, after which the selected priority order is started. This confirmation-based sequence prevents a race condition in which the new order could otherwise be initiated before the firmware had completed cancellation of the previous kit.

When the interrupted order is resumed, bins that were already completed are requested with a quantity of zero rather than being picked again. The remaining bins are requested according to their outstanding requirements. This allows the same kit-request and sequencing mechanism to handle both newly started and resumed orders, while preventing components that were already successfully picked from being requested a second time.

4.4.7 Inventory Accuracy Tracking

The dashboard presents four live performance metrics to the operator: Inventory Accuracy, Bins Completed, Corrections Needed, and Anomalies Detected. These metrics are derived from four counters maintained internally by the dashboard: the total number of completed bin picks, the number of those picks completed accurately on the first attempt, the number of over-pick corrections, and the number of detected anomalies. The total number of completed bin picks is displayed as Bins Completed, the over-pick correction count as Corrections Needed, and the anomaly count as Anomalies Detected. The number of accurately completed picks is not displayed as a separate value; instead, it is combined with the total number of completed picks to calculate the Inventory Accuracy percentage shown on the dashboard. Inventory accuracy is therefore calculated as the percentage of accurately completed picks relative to the total number of completed picks. These counters are updated when the corresponding firmware events are received through MQTT rather than through periodic polling, allowing the dashboard metrics to reflect system events as they occur.

The over-pick correction counter is managed separately for each armed bin. When a bin is armed, its correction flag is reset, and the first over-pick detected during that picking session increments the correction counter. Further over-pick events for the same bin during the same session do not increase the counter again. This prevents repeated correction attempts while resolving a single over-pick from being counted as multiple independent errors.

This metric therefore reflects the proportion of completed picks that reached the required quantity without requiring an over-pick correction and provides a direct measure of picking performance for the Phase 3 evaluation presented in Chapter 5.

4.4.8 Dashboard Implementation and Bilingual Natural-Language Input

The Phase 3 dashboard connects directly to the MQTT broker through a WebSocket connection when the page is loaded and subscribes to the status, event, kit, and system topics required for operation. Unlike Phase 2, no backend server is used to receive, process, or relay this information to the dashboard, as described in Section 4.4.1. Bilingual English and Swedish support is implemented through a central translation lookup function that covers both static interface labels and dynamically generated messages. Functions responsible for rendering live information, including the bin displays, kit progress, Order Queue, and event log, use the currently selected language whenever their content is generated. Consequently, changing the language causes both existing interface elements and currently displayed dynamic information to be regenerated in the selected language rather than translating only the static elements.

The dashboard also supports natural-language kit requests using deterministic input-processing logic based on regular expressions rather than a language model. This approach was selected because the expected request structure is sufficiently constrained to be handled predictably without introducing the uncertainty associated with AI-based interpretation. The input-processing logic uses two sequential passes. The first pass searches for explicit bin-specific expressions and takes precedence whenever a corresponding value is identified. The second pass handles more general expressions and is used only to populate bin quantities that were not already identified by the first pass. This ordering was introduced to prevent a possible misinterpretation identified during development. For example, in a request such as *"Kit of 15, bin 1 has 10"*, a general pattern could incorrectly interpret the first number, 15, as the quantity for bin 1 instead of using

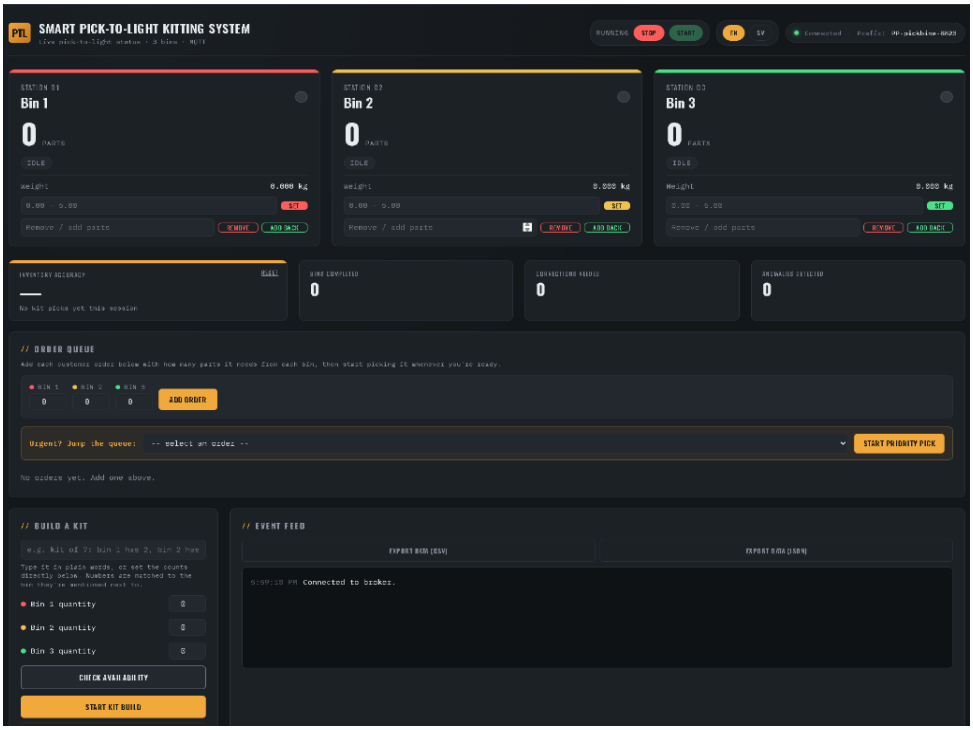


Figure 16. Operator dashboard implemented for the Phase 3 prototype.

the explicitly stated value of 10. By giving the bin-specific pattern priority, the input-processing logic correctly distinguishes the overall kit quantity from the quantity associated with an individual bin.

The same input-processing logic supports equivalent request expressions in both English and Swedish. The interpretation of the request therefore remains consistent with the selected interface language, while the deterministic nature of the approach ensures that equivalent inputs are processed according to the same predefined rules.

4.4.9 Implementation Challenges

Several implementation problems were identified and resolved during the development and testing of the Phase 3 prototype. These issues affected memory usage, communication reliability, startup behaviour, and dashboard consistency, and their resolution required changes to both the firmware architecture and the dashboard logic.

The most significant firmware issue was a recurring memory-related crash during operation. After investigation, the problem was traced to three separate causes: insufficiently controlled memory allocation for JSON messages, excessive temporary memory usage during rapid transitions between bins in a kit, and an MQTT message-buffer limitation. These issues were addressed by explicitly controlling the memory used for JSON messages, restructuring the kit-transition process using the deferred kit-advance mechanism described in Section 4.4.3, and increasing the MQTT buffer capacity. After these changes, the recurring memory-related crashes were no longer observed during testing.

A separate reliability issue concerned the firmware startup and communication behaviour. The firmware connects to WiFi before performing the initial tare operation, which also provides time for the simulated load cells to settle. Each bin is then tared and its reading is checked to ensure that the result is sufficiently close to zero; if necessary, the tare operation is repeated. MQTT communication was also changed to operate

without blocking the main control loop, since an earlier reconnect approach could temporarily stop the firmware when the broker was unavailable. Finally, bin status messages are published only when the corresponding value changes rather than continuously, reducing unnecessary communication and memory usage during longer test sessions.

A further issue was identified in the dashboard's Order Queue display. A queued order could continue to show an outdated stock availability status after the actual bin stock had changed, because the queue display was only regenerated when another dashboard event happened to trigger a refresh. The issue was resolved by explicitly re-rendering the Order Queue whenever a bin's live stock value changes. This ensures that the information presented to the operator remains synchronized with the latest stock data received from the firmware.

Overall, these implementation changes improved the stability and consistency of the Phase 3 prototype while also demonstrating the importance of considering memory management, non-blocking communication, startup validation, and event-driven interface updates when extending the system to multiple simultaneously monitored bins.

4.5 Chapter Summary

This chapter has presented the practical implementation of the three prototypes developed for this thesis, building on the methodology described in Chapter 3. Phase 1 established and tested the core sensor-fusion concept by combining a load cell and a depth camera to produce a combined inventory estimate that is more robust than either sensing channel alone, including a fusion-specific capability for inferring inventory that is not directly visible to either sensing method. Phase 2 then moved to a weight-based inventory-monitoring approach in a real industrial pilot deployment at Binar Solutions AB, where weight sensing was integrated into an existing Poka Yoke pick-to-light system. The phase further extended the system with a natural-language AI kitting assistant and a layered safety architecture combining deterministic verification with independent AI review. Phase 3 then explored the extension of the weight-based pick-to-light concept to a multi-bin configuration, including sequential kit handling and demand planning across a queue of orders, implemented and verified in a Wokwi-based hardware simulation.

Across the three phases, several implementation decisions were applied consistently rather than independently. These include the adaptive weight-smoothing approach used across the weight-sensing channels and the forward-looking, running-balance demand projection implemented independently in the Order Queue of both Phase 2 and Phase 3. Each phase also revealed practical engineering challenges during development, several of which required changes to the system architecture rather than only local corrections. These challenges, together with the implemented solutions, have been described throughout the chapter to document how the prototypes evolved during development.

The following chapter presents the experimental results obtained from testing the three prototypes and discusses the findings in relation to the research questions introduced in Section 1.3.

5 Results and discussion

5.1 Overview and Evaluation Approach

This chapter presents and discusses the experimental results obtained from testing the three prototypes described in Chapter 4, following the evaluation methodology and performance metrics established in Sections 3.3.5, 3.4.5, 3.5.5, and 3.6. Each phase is presented through a results section followed by a corresponding discussion section, reflecting the different experimental conditions, objectives, and evaluation criteria associated with each prototype. Section 5.2 presents the results of Phase 1, followed by the discussion in Section 5.3. Section 5.4 presents the results of Phase 2, followed by the discussion in Section 5.5. Section 5.6 presents the results of Phase 3, followed by the discussion in Section 5.7. Section 5.8 then provides a cross-phase discussion of the findings in relation to the research questions introduced in Section 1.3, followed by the chapter summary in Section 5.9.

The quantitative results presented in this chapter are based on data collected during prototype testing using the logging and measurement mechanisms described in Chapter 4. Where applicable, recorded system data are compared with independently verified ground-truth counts to assess measurement and detection performance. Results that could not be captured automatically, or observations that occurred qualitatively during testing, are identified explicitly and are not presented as quantitative measurements.

5.2 Phase 1 Results

5.2.1 Overview of Testing Conducted

Phase 1 testing was conducted according to the experimental protocol described in Sections 3.3.5 and 3.6. The testing consisted of three main data-collection activities. First, a controlled trial was performed to compare inventory-count accuracy across the three sensing modes using identical part arrangements with independently verified ground-truth counts. Second, a continuous recording was conducted during normal Fusion-mode operation under both baseline conditions, where only the correct parts were present, and mixed-part conditions, where both correct and incorrect parts were present. Third, two targeted test sessions were conducted to examine specific system behaviours: the first investigated mode transitions during sensor failure and subsequent recovery, while the second examined the system's hidden-part inference capability through repeated test cases.

5.2.2 Count Accuracy: Matched Controlled Trial

To directly compare the counting accuracy of the three sensing modes under identical physical conditions, ten matched trial cycles were conducted. In each cycle, a fixed arrangement of parts was left undisturbed while the system was evaluated in turn using Fusion, Weight-Only, and Vision-Only modes, resulting in thirty trials in total. The trials were evenly divided between a baseline condition using only correct parts, with fifteen trials, and a mixed condition containing both correct and deliberately incorrect or

damaged parts, with fifteen trials. For each trial, the system-reported count was compared with a manually verified ground-truth count appropriate to the sensing mode. For Fusion and Vision-Only modes, the reference count was the number of correct parts present, since these modes can distinguish correct parts from incorrect parts. For Weight-Only mode, the reference count was the total number of parts present, since weight-based sensing alone cannot distinguish between correct and incorrect parts of similar weight.

Table 1. Count accuracy per sensing mode across ten matched trial cycles, corresponding to thirty individual trials in total.

Sensing mode	Trails	Exact-match accuracy	Mean absolute error (parts)	Mean confidence
Fusion	10	100%	0	84.5%
Weight-Only	10	100%	0	50%
Vision-Only	10	100%	0	40%

All three sensing modes produced an exact count match in every trial, resulting in an exact-match accuracy of 100% and a mean absolute error of zero parts for each mode. The confidence values, however, differed considerably between the modes. Fusion produced the highest mean confidence at 84.5%, followed by Weight-Only at 50.0% and Vision-Only at 40.0%. These confidence values should not be interpreted as accuracy measurements; rather, they represent the confidence associated with the system's respective count output.

The results should be interpreted within the conditions tested. The part counts ranged from one to nine, and the trials were conducted under controlled lighting and controlled physical arrangements. Occlusion beyond the deliberate hidden-part cases was not included in this matched trial and is evaluated separately in Section 5.2.5. Therefore, the results demonstrate that all three sensing modes achieved exact counting under the tested conditions, while the confidence values indicate differences in how strongly each mode supported its corresponding count estimate. The implications of these differences are discussed further in Section 5.3.

5.2.3 Mode Comparison Across a Continuous Session

To evaluate system behaviour under extended operating conditions rather than discrete controlled trials, a continuous monitoring session was recorded while the system transitioned between operating modes as sensor availability changed. A total of 37,350 logged frames were recorded during the session, allowing the operating modes to be compared in terms of their occurrence, confidence, sensor agreement, wrong-part detection, and processing latency.

Fusion mode accounted for the largest proportion of the recorded session, representing 69.3% of all logged frames. Vision-Only operation accounted for 26.1%, Weight-Only operation for 4.4%, and System Failure for 0.2%. Fusion mode also produced the highest mean confidence at 79.8%, compared with 50.0% for Weight-Only and 39.2% for Vision-Only operation. Sensor agreement was 75.6% during Fusion operation. This metric is only reported for Fusion mode because it represents the consistency between the weight-based and vision-based count estimates and therefore requires both sensing channels to be active simultaneously.

Table 2. Mode comparison across a continuous monitoring session (Session 1).

Sensing mode	Frames	% of session	Wrong-part rate (%)	Mean confi-dence(%)	Mean agree-ment (%)	Mean latency (ms)
Fusion	25,895	69.3	63.78	79.8	75.6	33.8
Vision-Only	9,755	26.1	53.99	39.2	N/A	8.1
Weight-Only	1,638	4.4	N/A	50.0	N/A	273.1
System Failure	62	0.2	N/A	0.0	N/A	N/A

The wrong-part rate, defined as the percentage of frames in which the system flagged at least one incorrect or damaged part, was 63.78% for Fusion and 53.99% for Vision-Only operation. This metric is not applicable to Weight-Only operation or System Failure because wrong-part detection depends on the vision channel. Mean latency was 8.1 ms for Vision-Only operation, 33.8 ms for Fusion operation, and 273.1 ms for Weight-Only operation.

The mean confidence values in Table 2 represent averages across all logged frames in the continuous session and therefore differ from the confidence values in Table 1, which were calculated from the manually verified trials in the matched controlled experiment. Table 1 represents confidence during specific ground-truth-verified trials, whereas Table 2 represents confidence across the complete session, including transitional and settling periods that were not individually verified against ground truth.

The differences in mean confidence and wrong-part detection between the sensing modes are illustrated in Figure 17.

The relatively high wrong-part rates observed for both Fusion and Vision-Only operation are reported here as an observation from the recorded data. Their interpretation, including whether the values may be related to the sensitivity of the vision-based classification rather than the actual occurrence of incorrect or damaged parts, is discussed further in Section 5.3.

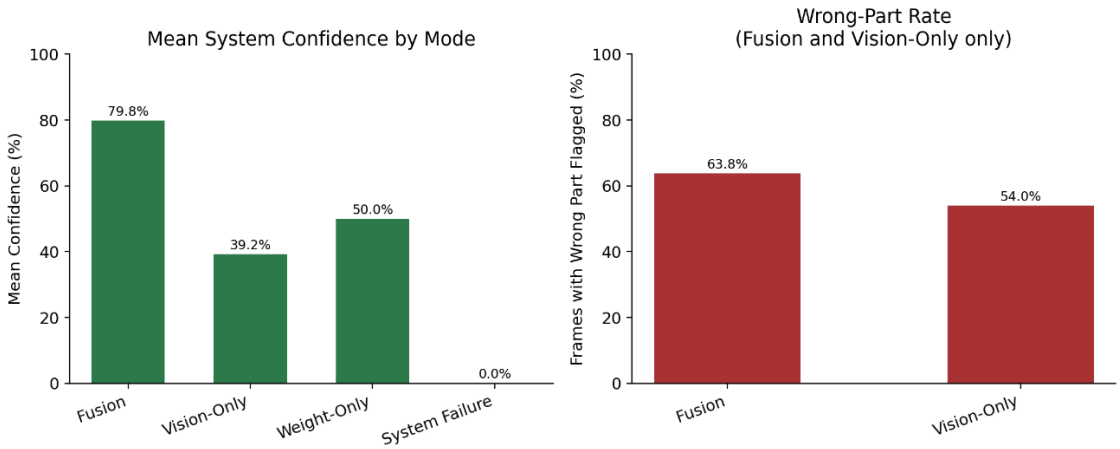


Figure 17. Mean system confidence by sensing mode (left) and wrong-part rate for Fusion and Vision-Only operation (right) during the continuous monitoring session.

5.2.4 System Failure and Fusion Recovery

To evaluate the system's fault-tolerant behaviour, a dedicated test session was conducted in which the weight and vision sensing channels were deliberately disabled and subsequently restored. The system remained running throughout the test without requiring an application restart. A total of 1,947 frames were logged, covering the sequence of operating modes shown in Table 3.

Table 3. Sequence of operating-mode segments during the system failure and Fusion recovery session.

Segment	Sensing mode	Frames	Approx. time	Mean confidence (%)	Mean agreement (%)
1	Fusion	245	19:11:07-19:11:15	69.8	74.5
2	Vision-Only	19	19:11:15-19:11:32	45.8	N/A
3	System Failure	1,220	19:11:32-19:15:26	0	N/A
4	Weight-Only	76	19:15:41-19:15:47	50.0	N/A
5	Fusion (Recovery)	387	19:15:47-19:16:00	69.9	74.1

The session began in Fusion mode, with a mean confidence of 69.8% and mean agreement of 74.4%, consistent with the continuous-session results presented in Section 5.2.3. When the weight channel was disabled, the system transitioned to Vision-Only operation, where the confidence decreased to 30%. When the vision channel was subsequently disabled, the system entered System Failure mode. During this segment, the mean confidence was 0.0%, as neither sensing channel was providing a valid current measurement.

System Failure persisted for 1,220 consecutive frames, making it the longest operating-mode segment in the session. During this segment, the logged system state showed a retained weight reading of 8, zero vision detections, and a confidence of 0.0%.

The weight channel recovered first, causing the system to transition to Weight-Only operation for 76 frames. After the vision channel was restored, the system returned to Fusion mode. During the recovery segment, confidence increased from 60% to 70% within the first five frames and then remained stable. The resulting mean confidence was 69.9%, while the mean agreement was 74.1%. These values closely matched those recorded during the initial Fusion segment, where the corresponding values were 69.8% and 74.4%.

The weight reading remained at 8 throughout the session, indicating that the observed mode transitions were caused by the deliberate sensor disconnections and recoveries rather than by a change in the physical inventory during the test.

5.2.5 Hidden-Part Inference

A dedicated session comprising 11,761 logged frames was recorded to evaluate the system's ability to infer parts that were physically stacked underneath other parts and therefore not visible to the camera. Hidden-part inference operates only in Fusion mode, where the weight- and vision-based counts can be compared to identify an unexplained difference. Four hidden-part configurations were deliberately created and held stable in sequence: one part hidden, two parts hidden, four parts hidden, and three parts hidden.

The number of physically hidden parts was independently verified for each configuration and used as the ground truth for comparison with the system's inferred hidden-part count.

Table 4. Hidden-part inference accuracy for four configurations with independently verified ground-truth hidden-part counts.

Configuration	Time window	Frames	Correct inferred count (%)	Suspected = True (%)	Mean confidence, correct frames (%)
1 part hidden	17:26:14-17:27:02	1,469	99.5	99.6	95.0
2 parts hidden	17:27:07-17:28:34	2,638	94.0	100	79.3
4 parts hidden	17:30:09-17:31:29	2,428	98.3	99.5	60.0
3 parts hidden	17:31:30-17:32:00	929	88.2	100	60.0

Across the four configurations, the system reported the correct hidden-part count in 7,146 of the 7,464 evaluated frames, corresponding to an overall accuracy of 95.7%. The hidden-part indicator remained active for the tested configurations, with the HiddenPartSuspected state being true in at least 99.5% of the evaluated frames for each configuration. This indicates that the system consistently identified the presence of an additional hidden quantity once the hidden-part condition had been established.

The results also show a difference in confidence between the tested configurations. The highest mean confidence was recorded for the one-part configuration at 95.0%, while the mean confidence decreased to 79.3% for two hidden parts and 60.0% for both the three- and four-part configurations. Although the inferred count remained highly accurate in most configurations, the lower confidence at higher hidden-part counts indicates that the system became less certain about its estimate as the number of hidden parts increased.

An additional period between the two-part and four-part configurations was used to manually reconfigure the bin and verify the dashboard output rather than to maintain a stable test condition. This period was therefore excluded from the accuracy calculations presented in Table 4. The implications of the observed relationship between hidden-part count, inference accuracy, and confidence are discussed further in Section 5.3.

5.3 Phase 1 Discussion

Phase 1 investigated whether combining weight-based and vision-based sensing could improve the reliability of inventory monitoring compared with either sensing modality operating independently, and whether the combination could provide information that neither modality could obtain alone. The results presented in Section 5.2 provide evidence for both aspects, although the improvements are more apparent in confidence, additional sensing capability, and fault handling than in raw counting accuracy.

Regarding counting accuracy, all three sensing modes achieved a 100% exact-match rate with a mean absolute error of zero in the matched controlled trial presented in Section 5.2.2. Therefore, sensor fusion did not improve the measured counting accuracy under the specific conditions tested, where the number of parts was limited and

the physical arrangements were controlled. However, the confidence associated with the reported counts differed substantially between the modes. Fusion achieved a mean confidence of 84.5%, compared with 50.0% for Weight-Only and 40.0% for Vision-Only operation. The main benefit of fusion in this experiment was therefore not the correction of counting errors that occurred in the individual sensing modes, since no such errors were observed in the controlled trials, but the stronger confidence associated with the combined estimate. This observation is also supported by the continuous-session results in Section 5.2.3, where Fusion mode produced the highest mean confidence of the operating modes observed, at 79.8%.

The strongest evidence that fusion provides information beyond the individual sensing modalities is given by the hidden-part inference results in Section 5.2.5. Vision sensing alone cannot detect a part positioned beneath another visible part, while weight sensing alone cannot determine whether the measured weight corresponds to visible parts or to additional parts that are not visible. By comparing the two sensing channels, the system was able to identify a difference between the expected visible inventory and the weight-based estimate and use this information to infer hidden parts. Across the four tested configurations, the correct hidden-part count was obtained in 95.7% of the analysed frames. This demonstrates a capability that is not available from either Weight-Only or Vision-Only operation independently. The results therefore support the use of complementary sensing modalities not only to strengthen an existing measurement, but also to enable observations that cannot be obtained from either modality alone. This capability is a direct result of the sensor-fusion algorithm and the debounced hidden-part detection mechanism described in Sections 4.2.5 and 4.2.6, which cross-check the two sensing channels rather than treating them as independent count sources.

This finding is consistent with the related work discussed in Section 2.10, where multi-sensor fusion has been shown to improve inventory monitoring under conditions such as occlusion and partial sensor visibility [15]. The present results provide similar evidence at the workstation level: the combination of weight and vision sensing enabled the prototype to identify hidden inventory that could not be reliably observed by either sensing modality alone. At the same time, the results illustrate the trade-off identified in the literature between improved monitoring capability and increased system complexity. In contrast to more complex fusion architectures described in previous work, the present prototype uses a deterministic, rule-based approach with two sensing modalities, demonstrating that useful fusion behaviour can be achieved without requiring a large number of sensors or a probabilistic fusion framework.

The sensor-failure and recovery results in Section 5.2.4 provide additional evidence regarding system reliability, particularly its ability to handle loss and subsequent recovery of the sensing channels. When both sensing channels became unavailable, the system entered System Failure and reported a confidence of 0%, rather than continuing to provide a potentially misleading inventory estimate. After the sensors were restored, the system returned to Fusion mode without requiring an application restart. The recovery segment produced a mean confidence of 69.9% and a mean agreement of 74.1%, which were close to the corresponding values of 69.8% and 74.4% recorded before the failure. This indicates that the system was able to recognise sensor unavailability and recover predictably once the sensing channels became available again. This behaviour reflects the sensor-health evaluation strategy described in Section 4.2.4, which allows the system to distinguish between available and unavailable sensing states and select the corresponding operating mode.

Two limitations were also identified from the Phase 1 results. First, the wrong-part rate recorded during the continuous session was relatively high, at 63.78% for Fusion and 53.99% for Vision-Only operation. This result contrasts with the 100% exact-match counting accuracy observed in the controlled trials. Since wrong-part classification and quantity counting are separate outputs of the system, the high wrong-part rate did not affect the counting accuracy reported in Section 5.2.2. However, the result suggests that the vision-based classification threshold may have been sensitive to variations in the observed parts or test conditions and may have resulted in some parts being classified as incorrect more frequently than intended. The wrong-part detection result should therefore be interpreted with caution, and further threshold calibration and validation under a broader range of part conditions would be required before drawing conclusions about actual part quality.

Second, the hidden-part experiments showed that mean confidence decreased as the number of hidden parts increased, from 95.0% with one hidden part to 60.0% with three or four hidden parts, even though the inferred hidden-part count remained correct for the large majority of frames. This indicates that the confidence value did not decrease in direct proportion to counting correctness across all tested conditions. Consequently, a lower confidence value at higher hidden-part counts should not automatically be interpreted as an incorrect inventory estimate. The relationship between confidence and actual correctness would therefore benefit from further investigation and calibration if the confidence value is to be used as an operator-facing indication of measurement reliability.

The generalizability of these findings is limited by the conditions under which Phase 1 was tested. The experiments used controlled workstation conditions, relatively small part counts, and a limited set of part arrangements, while the hidden-part experiments were conducted using deliberately constructed configurations. The results therefore demonstrate the behaviour of the prototype under the tested conditions rather than establishing performance for all industrial kitting environments. Further validation with larger part quantities, greater variation in part geometry and arrangement, changing environmental conditions, and physical workstation operation would be required before the results could be generalized to broader industrial applications.

Overall, the Phase 1 results support the first research question by demonstrating that combining weight-based and vision-based sensing provided benefits beyond those observed from either sensing modality alone. Although no improvement in raw counting accuracy was demonstrated under the controlled conditions, Fusion mode produced higher confidence, enabled hidden-part inference, and allowed the system to identify and recover from sensor failures in a predictable manner. At the same time, the results revealed limitations in wrong-part classification and in the relationship between confidence and correctness. These limitations do not indicate a fundamental limitation of the fusion concept under the tested conditions, but rather identify specific aspects of the implementation that would require further calibration and validation under a wider range of operating conditions.

5.4 Phase 2 Results

5.4.1 Overview of Testing Conducted

Phase 2 testing evaluated the AI-assisted pick-to-light prototype across seven testing activities. Several tests used predefined named kits - Kit A, Kit B, Kit C, and Kit D corresponding to fixed quantities of 1, 2, 3, and 4 units respectively, while other tests

used requests specifying an arbitrary quantity directly, such as a request for a kit of 5 units. A baseline session of 16 picks, described in Section 5.4.2, covered manual, AI-triggered, and Order Queue-triggered picking, with the quantities reported by the system independently cross-checked through a separate manual verification process. Six additional targeted test sets were conducted to examine specific system capabilities: AI-assisted single-unit kit requests (7 trials), AI-assisted multi-unit kit requests (6 trials), over-pick detection and recovery (4 trials), informational query handling (5 trials), Order Queue shortage detection (5 queued orders), and bilingual English/Swedish operation (4 trials).

5.4.2 Pick Tracking Accuracy and Guardrail Behaviour

Table 5. Baseline pick trials by trigger source (Session 1).

Trigger source	Trials	Result: pick_confirmed	Result: pick_uncertain (over-pick)
Manual	6	6	0
AI	4	2	2
Order queue	6	4	2
Total	16	12	4

Across all 16 baseline trials, the quantity reported by the system matched the quantity physically removed in every trial. The four trials resulting in pick_uncertain were intentional over-pick cases in which the operator removed more units than requested. In these cases, the system correctly detected the mismatch and withheld automatic confirmation rather than accepting an incorrect pick. The reported quantities from all 16 trials were independently checked against the manually verified quantities in a separate verification pass, resulting in 100% agreement between the system-reported and manually verified quantities. Mean pick latency across all 16 trials was 9.8 ms, ranging from 7 to 14 ms.

The guardrail behaviour was also evaluated for AI-triggered picks, where an incorrect AI tool call could potentially result in an inappropriate pick request or an incorrect confirmation. All four AI-triggered trials, including the two intentional over-pick cases, passed the guardrail check correctly. This resulted in a Tool-Call Correctness rate of 100% for the AI-triggered picks in this session.

5.4.3 AI-Assisted Kit Requests: Single- and Multi-Unit

Table 6. Single-unit AI-assisted kit requests.

Trials	Message	Result matched request	Latency (s)
1	"Prepare a kit of 2 units"	Yes	7.0
2	"Prepare a kit C"	Yes	9.0
3	"Prepare a kit of 5 units"	Yes	8.0
4	"Prepare a kit D"	Yes	7.0
5	"Make a kit of 1 unit"	Yes	8.0
6	"Make a kit of two units"	Yes	7.0
7	"Prepare a kit A"	Yes	8.0

All seven single-kit request trials were correctly interpreted and resulted in a kit matching the requested quantity, giving a 100% request-match rate. The mean assistant response latency was 7.7 s, with individual response times ranging from 7.0 to 9.0 s. The requests included different forms of input, including explicit quantity requests such

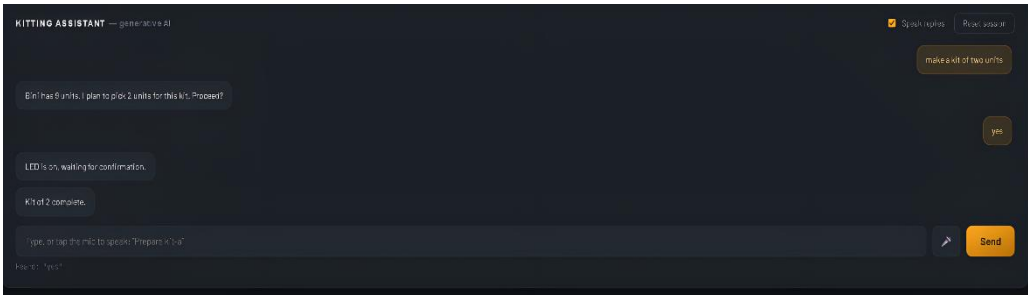


Figure 18. Kitting Assistant chat interface for a kit of two units (Trial 6), showing the assistant's stock check, confirmation request, and completion message.

as "Prepare a kit of 5 units" and "Make a kit of two units", named-kit requests such as "prepare a kit C". Trials 6 is shown in Figure 18, illustrating the assistant's stock check, confirmation request, and kit-completion response.

Table 7. Multi-unit AI-assisted kit request trials.

Tri-als	Message	Units re-quested	Pick cycles	Total picked	Match	Confirma-tion asked only once
1	"Prepare a kit of 3 units"	3	2	3	Yes	Yes
2	"Prepare a kit B"	2	1	2	Yes	Yes
3	"Prepare a kit of 4 units"	4	2	4	Yes	Yes
4	"Prepare a kit A"	1	1	1	Yes	Yes
5	"Make a kit of 2 unit"	2	2	2	Yes	Yes
6	"Prepare a kit D"	4	3	4	Yes	Yes

All six multi-unit request trials resulted in a total picked quantity matching the requested quantity, giving a 100% match rate. Requests involving more than one unit were generally completed through multiple pick cycles, with the number of cycles ranging from one to three depending on the requested quantity and the available picking sequence. In every trial, the assistant requested operator confirmation only once for the complete kit rather than repeating the confirmation for each individual pick cycle. This indicates that the multi-step picking process was handled as a single kitting operation rather than as a series of independent kit requests.

5.4.4 Order Queue Shortage Detection

The Order Queue allows multiple kit requests to be queued in sequence. Rather than checking each order only against the bin's current stock, the system checks each queued order against the projected stock balance after considering all orders ahead of it in the queue. This allows a shortage several orders down the queue to be identified and flagged in advance, rather than only after the available stock has actually been consumed.

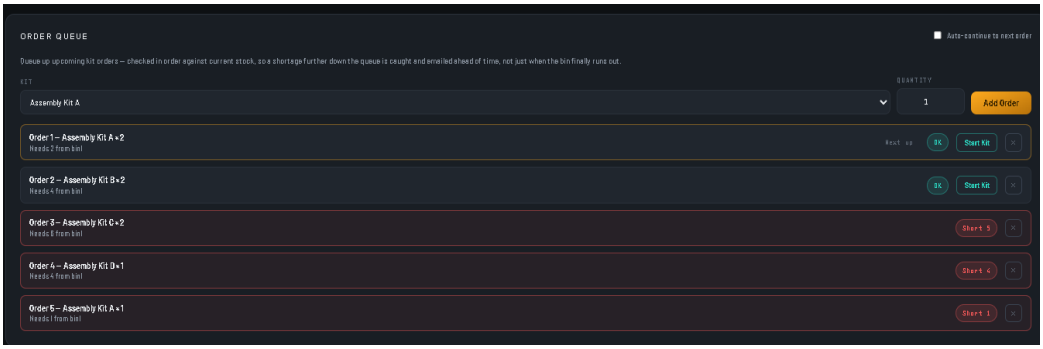


Figure 19. Order Queue interface showing projected shortages before picking begins.

Table 8. Order queue shortage detection across five queued orders.

Order	Kit	Qty needed	Stock at queue creation	Projected balance after prior orders	Flagged short by system
1	Kit A	2	7	7	No (correct)
2	Kit B	4	7	5	No (correct)
3	Kit C	6	7	1	Yes, short by 5 (correct)
4	Kit D	4	7	0	Yes, short by 4 (correct)
5	Kit A	1	7	0	Yes, short by 1 (correct)

All five queued orders were correctly classified: two orders were identified as fulfillable from the projected stock balance, while three were correctly flagged as short. In each of the three shortage cases, the system also reported the correct shortfall quantity, giving an order classification accuracy of 100% (5 of 5 orders). Figure 19 shows the Order Queue interface at the point of queue creation, with three of the five orders already projected to fall short before any picking had begun, based on the running stock balance after considering the preceding queued orders.

5.4.5 Over-Pick Detection and Recovery

Over-picking, defined as removing more units than requested, was examined for both AI-triggered and Order Queue-triggered picks. For AI-triggered picks, this behaviour was already captured in the baseline session described in Section 5.4.2. In both cases, the system detected the mismatch between the requested and removed quantities and returned a pick_uncertain result rather than automatically confirming the pick. These trials demonstrated that the guardrail detected the over-pick condition, while the subsequent recovery process was not evaluated as part of those baseline trials.

For Order Queue-triggered picks, over-picking was evaluated separately in a dedicated four-trial test that continued into the recovery workflow. When an over-pick was detected, the system generated a discrepancy event and moved the order into a needs-review state. The order was then only marked as resolved after the excess units had been physically returned to the bin.

Table 9. Order Queue over-pick detection and recovery.

Trials	Requested	Removed	Extra units	Event Fired	Moved to needs-review	Resolved correctly after return	Inventory accuracy before (%)	Inventory accuracy after (%)
1	2	3	1	Yes	Yes	Yes	100	83.3
2	3	5	2	Yes	Yes	Yes	83.3	66.7
3	5	8	3	Yes	Yes	Yes	77.7	70
4	4	6	2	Yes	Yes	Yes	100	50

In all four trials, the over-pick event was generated correctly, the order was moved to the needs-review state, and the discrepancy was resolved after the excess units were returned to the bin. This resulted in a 100% recovery success rate (4 of 4 trials). The inventory accuracy values shown in the final two columns represent the dashboard value immediately before and after the over-pick event, respectively. They therefore show the temporary change in the dashboard metric associated with the discrepancy and its subsequent resolution rather than measuring the accuracy of over-pick detection itself.

5.4.6 Informational Query Handling

Informational questions, consisting of queries about stock levels or system status that should be answered without triggering a pick, were evaluated across five trials. The tests examined both whether a pick was incorrectly triggered and whether the assistant's reported stock information matched the actual stock at the time of the question.

Table 10. Informational query handling.

Trials	Question asked	Assistant's answer	Pick triggered	Answer correct vs. real stock
1	"How many parts are available in bin1"	"Bin1 has 7 units."	No	Yes
2	"How many parts are in bin 1" (asked immediately after restocking)	"Bin1 has 5 units."	No	No
3	"Please check restock has been done"	"Bin1 has 8 units."	No	Yes
4	"is bin 1 online"	"Yes, bin 1 is online."	No	Yes
5	"Thank you"	"You're welcome."	No	Yes

All five informational queries received an appropriate response and none incorrectly triggered a pick, giving a 100% correct response rate (5 of 5). Among the three queries that requested or included stock information, two returned the correct stock value, while Trial 2 reported 5 units when 8 units were physically present. The subsequent query in Trial 3 correctly reported the updated stock level of 8 units.

5.4.7 Bilingual Operation

The assistant's ability to operate in Swedish alongside English was evaluated across four trials covering kit requests, a stock question, and a general conversational message. The

tests evaluated whether the Swedish input was correctly understood and whether the resulting responses were correct in both the voice output and the dashboard interface.

Table 11. Bilingual (Swedish) operation.

Trials	Test type	Swedish message sent	Assistant's Swedish reply	Voice correct	UI correct
1	Prepare kit	"Förbered kit-a"	"Kit-a behöver 1 enhet från bin1. Bin1 har 4 enheter tillgängliga. Fortsätt? (ja/nej)"	Yes	Yes
2	Stock question	"Hur många delar finns i behållare 1?"	"Bin1 har 7 enheter."	Yes	No
3	Prepare kit with quantity	"starta ett kit med 3 delar"	"Behållare 1 har 5 enheter, tillräckligt för att täcka 3 delar. Fortsätt? (ja/nej)"	Yes	Yes
4	Thank you message	"Tack så mycket"	"Välkommen"	Yes	Yes

All four trials were handled correctly in both voice and interface output, giving a 100% correctness rate. The assistant correctly interpreted a named-kit request (Trial 1), a direct stock question (Trial 2), a custom-quantity kit request (Trial 3), and a general conversational message (Trial 4). For both kit-request trials, the assistant checked the available stock against the requested quantity before asking for confirmation. All responses were generated in Swedish, including the surrounding conversational content, while kit and bin identifiers remained unchanged.

5.5 Phase 2 Discussion

Phase 2 investigated whether a manually confirmed pick-to-light system could be enhanced to provide reliable real-time stock verification while also supporting natural-language interaction for operator assistance. This directly addresses the limitation identified in Section 2.8, where conventional pick-to-light systems guide the operator to the correct location without independently verifying that the correct quantity has been removed. The results presented in Section 5.4 address both aspects of this objective.

Across all 16 baseline picks (Section 5.4.2), the system-reported quantity matched the physically removed quantity in every trial. This result was independently confirmed through a separate manual verification pass, giving 100% agreement. Under the tested conditions, these results demonstrate that continuous weight sensing can provide reliable automated stock verification when integrated into an existing pick-to-light workstation. A single weight-sensing channel was used because the prototype was designed to integrate directly into Binar Solutions AB's existing pick-to-light installation and therefore had to operate within the constraints of that setup. The results indicate that this weight-based approach provided reliable verification within those constraints. The integration of the sensor-fusion approach developed in Phase 1 into such a workstation is considered as a direction for future work in Chapter 6.

The system also went beyond simple quantity verification by distinguishing between confirmed and uncertain picks. When an over-pick occurred, the system did not silently

accept the discrepancy; instead, it withheld automatic confirmation. For order-queue-triggered picks, this was followed by an explicit recovery workflow in which the order was flagged for review and correctly resolved after the excess units were returned, with successful recovery in all four tested cases (Section 5.4.5). This directly addresses the gap identified in related work: Trojanowska et al. [16], report a Poka-Yoke pick-to-light device that confirms a pick using an optical-reflective sensor and automatically advances to the next step, but does not verify whether the quantity removed was correct. The present system therefore extends pick-to-light verification by checking quantity in addition to location and by providing a defined recovery path when a discrepancy is detected.

The natural-language interaction results also support the second part of RQ2. AI-assisted kit requests were correctly interpreted and fulfilled in all seven single-unit and all six multi-unit trials (Section 5.4.3). Order-queue shortages were correctly predicted in advance of stock running out in all 5 tested orders (Section 5.4.4), and bilingual operation in Swedish was handled correctly across all four tested trials (Section 5.4.7). Together, these results indicate that the natural-language interface could be integrated without replacing the underlying stock-verification and guardrail logic. The AI layer operated on top of the deterministic verification mechanisms rather than being responsible for the physical quantity verification itself.

Two limitations qualify these findings. First, an over-pick made through an AI-triggered request was correctly identified as pick_uncertain (Section 5.4.2), but did not proceed through the same needs-review and resolution workflow implemented for order-queue-triggered over-picks (Section 5.4.5). This reflects the constraints of testing with free-tier AI access: implementing and verifying an equivalent review step for AI-triggered picks would require substantially more language-model usage than was available, and this workflow was therefore only fully implemented and evaluated for the order-queue path. Extending the same recovery mechanism to AI-triggered picks is identified as future work in Chapter 6.

Second, one of the five informational queries returned a stock value that was outdated immediately after a restocking event (Section 5.4.6), while the following query returned the updated value. The underlying weight-sensing system was functioning correctly; the discrepancy occurred in the interaction between the AI assistant and the current stock information. This indicates that the natural-language layer did not always provide the most recent stock value immediately after a change, and therefore that further improvement is required before treating the AI interface as a fully reliable real-time information source.

More generally, the reliability of the AI-assisted interaction is expected to vary with the underlying language-model access used, since testing relied on free-tier access with usage limits, and a separate corroborating session produced a lower tool-call correctness rate (50%, n = 4) than the 100% observed in the main baseline session (n = 4). This variation suggests that the AI interaction layer can introduce variability that is not present in the underlying deterministic verification logic. In addition, most individual test categories in Section 5.4 involved a small number of trials (4 to 16). The results should therefore be interpreted as evidence that the prototype performed the tested functions successfully under the evaluated conditions, rather than as statistically generalizable reliability rates for sustained industrial operation. Larger and more consistent test sets would be required to establish performance beyond the tested pilot deployment at Binar Solutions AB.

Overall, the Phase 2 results support the second research question. The prototype demonstrated that a weight-based pick-to-light system can provide independently verified stock information, detect over-pick discrepancies, and provide a defined recovery process for order-queue picks. It also demonstrated natural-language interaction across single-unit, multi-unit, order-queue, and bilingual scenarios. The results further show that the deterministic verification layer can remain independent of the AI interaction layer, which is important because limitations in the AI component did not prevent the underlying weight-based verification from operating correctly. The main remaining limitations concern the completeness and consistency of the AI-assisted interaction and are therefore appropriate targets for future work discussed in Chapter 6.

5.6 Phase 3 Results

5.6.1 Overview of Testing Conducted

Phase 3 testing evaluated the multi-bin, simulation-based pick-to-light prototype across four areas of functionality, grouped according to the behaviour under evaluation. Core multi-bin operation was evaluated through 29 trials covering both kit-build picking (triggered via typed instruction, manual quantity entry, or the order queue) and manual pushbutton picking (Section 5.6.2). Guardrail and safety-event handling was evaluated across 20 trials covering over-pick correction, inactivity timeout, and idle-bin tamper detection (Section 5.6.3). Natural-language parsing, bilingual operation, and order-queue resume behaviour following interruption were evaluated together (Section 5.6.4). Demand-planning shortage projection across a queue of orders was evaluated using the live dashboard interface, since this calculation is performed entirely in the browser and is not captured in the event log (Section 5.6.5). Throughout the Phase 3 testing, four dashboard performance indicators were tracked: Inventory Accuracy, Corrections Needed, Anomalies Detected, and Bins Completed. The corresponding results are reported within the relevant test sections, with a cumulative summary across the complete testing session provided at the end of Section 5.6.

5.6.2 Multi-Bin Kit Sequencing and Manual Pick Tracking

Table 12. Multi-bin kit sequencing and manual pick tracking summary.

Test	Trigger source	Trials	Correct completion
Full multi-bin kit sequence	Kit build (typed instruction / quantity entry)	10	10/10 (100%)
Manual mode picks	Manual pushbutton	19	19/19 (100%)

The full multi-bin kit sequence test covered ten kit requests spanning different combinations of the three bins, with quantities per bin ranging from 1 to 3 units. In every trial, the required bins were activated and picked in the correct order, the picked quantity matched the requested quantity for each bin, and the kit was correctly marked as complete once all required bins had been picked. The manual mode picks test covered nineteen individual picks made using the physical pushbutton, with all three bins available and one bin picked at a time, with quantities ranging from 1 to 3 units. In every trial, the system correctly confirmed the pick with a count matching the quantity actually removed. Together, these results show that pick-tracking accuracy was consistent across both kit-build sequencing and manual single-bin picking, giving an Inventory Accuracy

of 100% (29 of 29 bin picks correct on the first attempt) across this combined baseline evaluation.

5.6.3 Guardrail and Safety-Event Handling

Table 13. Over-pick detection and correction in kit mode.

Trials	Bin	Target qty	Picked qty	Extra removed	Put-back qty required	Correctly resolved
1	1	3	4	1	1	Yes
2	2	2	4	2	2	Yes
3	3	4	7	3	3	Yes
4	3	3	4	1	1	Yes
5	2	3	5	2	2	Yes
6	1	1	3	2	2	Yes
7	1	1	4	3	3	Yes

Across all seven over-pick trials, the system correctly detected the discrepancy, calculated the exact quantity requiring put-back, and marked the bin complete only once the required quantity had been returned, giving a correction success rate of 100% (7 of 7). Trials 6 and 7 occurred on the same bin within the same kit, where two consecutive over-picks were made on Bin 1 before it was corrected. The system correctly handled both corrections in sequence, demonstrating that the correction logic was able to respond to repeated discrepancies within the same pick sequence.

Table 14. Inactivity timeout auto-disarm.

Trial	Mode	Idle condition	Result
1	Manual	All 3 bins armed, idle >60 s	All 3 bins auto-disarmed
2	Kit	Bin 1 active in a kit, idle >60 s	Bin 1 auto-disarmed, kit cancelled
3	Manual	All 3 bins armed, idle >60 s	All 3 bins auto-disarmed
4	Kit	Bin 1 active in a kit, idle >60 s	Bin 1 auto-disarmed, kit cancelled

In all four trials, the system correctly detected 60 seconds of inactivity and automatically disarmed the affected bin(s), returning to a ready state without operator intervention. In kit mode, this also triggered cancellation of the in-progress kit, preventing the system from remaining in an incomplete or unresponsive state.

Table 15. Idle-bin weight-change detection.

Direction	Bins tested	Trials	System response	Correctly handled
Weight decreased (parts removed while idle)	1, 2, 3 (two rounds)	6	Flagged as anomaly	6/6
Weight increased (parts added while idle)	1, 2, 3	3	Logged as informational message, not flagged as anomaly	3/3

Idle-bin weight-change detection was evaluated by changing the weight on each bin while the system was fully idle and no bin was armed, in both directions. By design, only unauthorized weight decreases are treated as anomalies, since an unexplained loss of parts could indicate an unrecorded pick or other unauthorized inventory change and

therefore requires operator attention. Unauthorized weight increases, such as restocking, do not carry the same risk to inventory reliability and are instead logged as informational messages rather than raised as anomalies. In all nine trials, the system applied this distinction correctly: every weight decrease while idle was flagged as an anomaly, while every weight increase was correctly logged without triggering a false anomaly alert.

5.6.4 Natural-Language Parsing, Bilingual Operation, and Order-Queue Resume

Natural-language kit parsing was evaluated for a specific edge case in which a leading total quantity (e.g. "kit of 15") could potentially be interpreted as a bin quantity rather than as the total quantity of the requested kit. This behaviour was tested across five trials using different total quantities and combinations of per-bin quantities.

Table 16. Natural-language parser edge-case handling.

Trials	Instruction typed	Parsed result	Total correctly excluded from bin fields
1	"kit of 15, bin 1 has 5, bin 2 has 5, bin 3 has 5"	bin1=5, bin2=5, bin3=5	Yes
2	"kit of 6: bin 1 has 2, bin 2 has 2, bin 3 has 2"	bin1=2, bin2=2, bin3=2	Yes
3	"kit of 4, bin 1 has 1, bin 2 has 2, bin 3 has 1"	bin1=1, bin2=2, bin3=1	Yes
4	"kit of 10, bin 1 has 4, bin 2 has 6"	bin1=4, bin2=6	Yes
5	"kit of 3, bin 2 has 2, bin 3 has 1"	bin2=2, bin3=1	Yes

In all five trials, the parser correctly extracted the quantity specified for each bin without allowing the leading total quantity to overwrite a bin field. This remained correct even when the total quantity was equal to one of the per-bin quantities. Figure 20(a) shows the corresponding English-language interface, including the parsed per-bin quantities and stock-availability check before the kit was built.

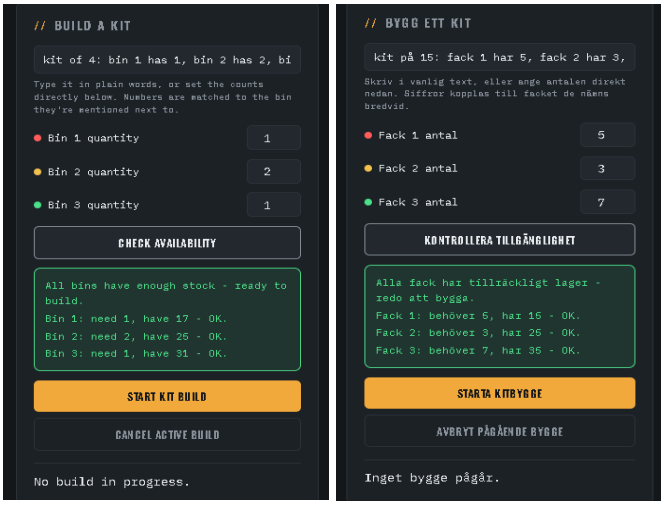


Figure 20. Kit-build interface showing correct per-bin parsing in (a) English and (b) Swedish.

The same parsing behaviour was evaluated in Swedish across three additional trials, using "fack" (compartment) instead of "bin."

Table 17. Bilingual (Swedish) parser verification.

Trials	Instruction typed	Parsed result
1	"kit på 15: fack 1 har 5, fack 2 har 3, fack 3 har 7"	fack1=5, fack2=3, fack3=7
2	"kit på 5: fack 1 har 2, fack 2 har 1, fack 3 har 2"	fack1=2, fack2=1, fack3=2
3	"kit på 2: fack 1 har 1, fack 2 har 1"	fack1=1, fack2=1

All three Swedish-language trials were parsed correctly, with the interface operating in Swedish and the per-bin quantities extracted correctly in the same manner as in the English-language trials. Figure 20(b) shows Trial 1, in which the interface confirmed sufficient stock for all three compartments before proceeding.

Order-queue resume behaviour under repeated interruption was evaluated using a single multi-bin order that was interrupted twice by unrelated priority orders. Bin 1 of the original order was completed first. A priority order was then started before Bin 2 of the original order began. After the priority order was completed, the original order was resumed, and the system correctly skipped Bin 1 and activated Bin 2 rather than requesting the already-completed bin again. Figure 21 shows this behaviour, with Bin 1 marked as "done" and the interface reporting "Resuming: 1 of 3 bins already done" before continuing with the remaining bins.

After Bin 2 was completed, the original order was interrupted a second time by another priority order. Once the second priority order was completed, the original order was resumed again, correctly skipping both Bin 1 and Bin 2 and proceeding directly to Bin 3, which completed the order. The result demonstrates that the resume logic retained the completion state across multiple interruptions and resumed the order from the next incomplete bin.

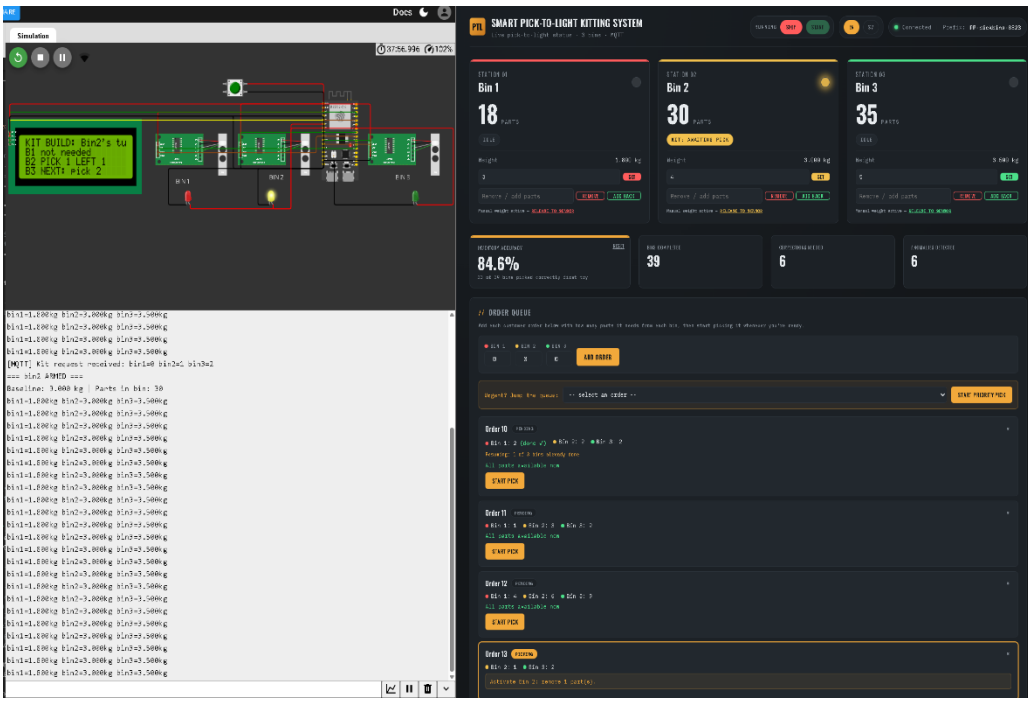


Figure 21. Order-queue resume after a priority interruption.

5.6.5 Demand-Planning Shortage Projection

Since the demand-planning shortage projection is calculated entirely within the dashboard interface rather than recorded as a discrete event, this behaviour was verified directly against the live interface, consistent with the approach described in Section 5.6.1. Two snapshots of the same order queue were captured: one before the bin stock was refilled and one after refilling, with an additional order subsequently added to the queue.

Table 18. Demand-planning shortage projection before and after bin refilling

Order	Kit demand (Bin 1 / Bin 2 / Bin 3)	Status before refill (stock 17/26/30)	Status after refill (stock 20/30/40)
15	5 / 5 / 5	All parts available	All parts available
16	10 / 15 / 20	All parts available	All parts available
17	5 / 10 / 15	Short by 3 / 4 / 10	All parts available
18	2 / 2 / 2	-	Short by 2 / 2 / 2

Before refilling, Orders 15 and 16 were correctly identified as fulfillable because their combined demand did not exceed the projected stock remaining after preceding orders in the queue. Order 17 was correctly flagged as short, with the shortfall in each bin (3, 4, and 10 units) matching the projected remaining balance after Orders 15 and 16. Figure 22 shows this state.

After the bins were refilled to 20, 30, and 40 units respectively, the projection was automatically recalculated. Order 17, which was previously flagged as short, was now correctly shown as fully available because the combined demand of Orders 15 to 17 (20, 30, and 40 units across the three bins) exactly matched the refilled stock levels. Order 18 was then added to the queue requesting two additional units from each bin. Since no stock remained after accounting for Orders 15 to 17, Order 18 was correctly flagged as short by two units in each bin. Figure 23 shows this updated state.

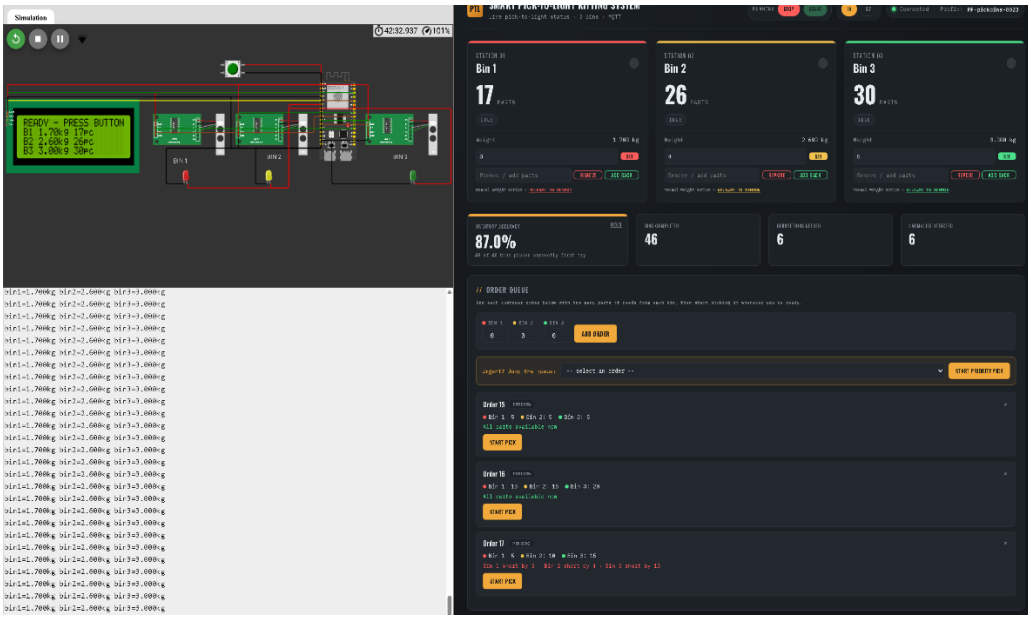


Figure 22. Order queue showing Order 17 correctly flagged as short based on projected stock after Orders 15 and 16.

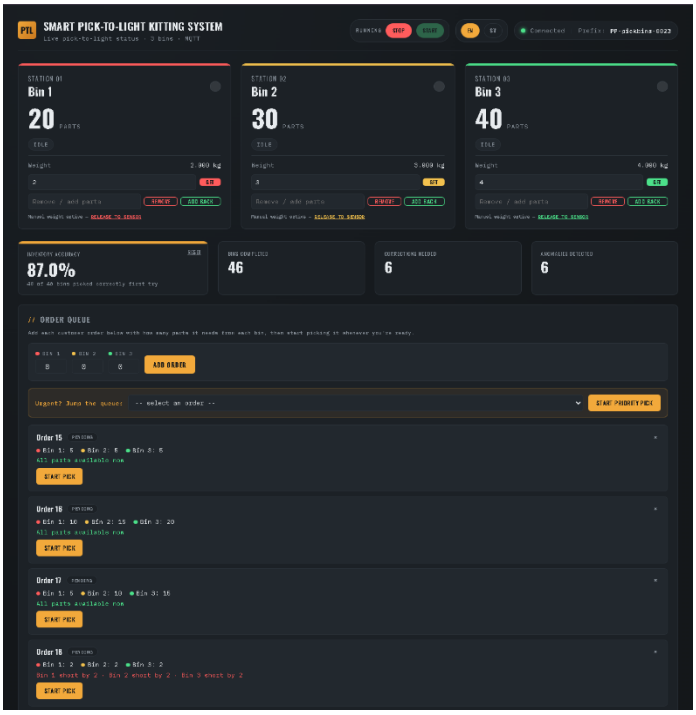


Figure 23. Order queue after refilling, showing Order 17 available and the newly added Order 18 flagged as short.

Together, the two snapshots demonstrate that the shortage projection was recalculated dynamically in response to both a change in available stock and the addition of a new order to the queue, rather than remaining fixed at the time an order was created.

Taken together, the dashboard's cumulative counters across the full Phase 3 testing session reported 46 Bins Completed, 6 Corrections Needed, 6 Anomalies Detected, and an overall Inventory Accuracy of 87.0% (40 of 46 bins picked correctly on the first attempt), corresponding to the four evaluation metrics defined in Section 3.5.5. The overall accuracy is lower than the 100% accuracy reported for the clean multi-bin and manual-pick trials in Section 5.6.2 because it is a running total across the entire session, including the deliberately triggered over-pick trials in Section 5.6.3 that were designed to test the correction logic. The 6 Corrections Needed corresponds to the six correction events recorded during the session, with the two consecutive over-picks on the same bin within Kit #4 handled as a single correction event. The 6 Anomalies Detected corresponds exactly to the six idle weight-decrease trials reported in Table 15, consistent with the system design in which unauthorized weight decreases are treated as anomalies while weight increases are logged as informational events. These cumulative figures should therefore be interpreted in the context of the deliberately fault-injected testing session, rather than as a measure of baseline reliability under normal operation; the individual results in Section 5.6.2 separately demonstrate 100% accuracy under clean picking conditions.

5.7 Phase 3 Discussion

Phase 3 was designed to investigate how the single-bin inventory monitoring concept developed in Phase 2 could be extended to a coordinated multi-bin kitting station, and what additional challenges emerge when scaling the approach beyond a single workstation. The results in Section 5.6 address both aspects of this question.

Basic pick-tracking accuracy was preserved when scaling from one bin to three. Across 29 trials combining kit-build sequencing and manual pushbutton picking, with varying combinations of the three bins (Section 5.6.2), pick-tracking accuracy remained at 100%. Within the simulated test environment, this indicates that coordinating multiple independent sensing channels under a single controller did not introduce additional tracking errors under the tested conditions.

Scaling to multiple bins did, however, introduce specific coordination challenges that do not arise in the same form in a single-bin system, and these were tested directly. First, free-text kit requests introduce a parsing uncertainty that has no equivalent when only one bin is involved: a stated total quantity could potentially be interpreted as a bin quantity rather than as a total across the requested bins. This was correctly handled in all five English-language and all three Swedish-language trials (Section 5.6.4), with the total consistently excluded from the individual bin quantities regardless of phrasing or language. Second, guardrail behaviour needed to operate correctly on a per-bin basis within a coordinated multi-bin kit, rather than only at the level of a single pick. Over-pick correction (Section 5.6.3) correctly handled both individual discrepancies and two consecutive over-picks on the same bin within one kit. Inactivity timeout also correctly distinguished between disarming an individual active bin and cancelling an in-progress kit, depending on the operating context. Third, tracking partial completion state across an interrupted multi-bin order introduced a further coordination requirement, since a resumed order must identify which bins have already been completed. This was verified across two consecutive interruptions of the same order (Section 5.6.4), with the system correctly resuming from the appropriate remaining bin each time rather than repeating a completed bin.

Scaling to multiple bins also extended the demand-planning shortage projection capability evaluated in Phase 2 to a multi-bin setting (Section 5.6.5). While the Phase 2 implementation projected shortages for orders involving a single bin, the Phase 3 implementation had to account for the combined demand across multiple bins and multiple queued orders. The system correctly identified projected shortages before the available stock was exhausted and recalculated the projection after both a stock change and the addition of a new order to the queue. This demonstrates that the demand-planning logic was extended from single-bin projection to coordinated multi-bin demand planning and responded dynamically to changes in both stock levels and queue state.

These results are broadly consistent with the reported benefits of IoT-based inventory monitoring, where Akeem and Sonu [1] report improvements in inventory accuracy and near-real-time updating compared with more traditional approaches. The present results extend this perspective from monitoring at a single sensing point to coordinated operation across multiple sensing points, although at a considerably smaller scale of three bins than typical larger-scale inventory-monitoring deployments.

An important qualification is that Phase 3 was implemented and evaluated entirely within the Wokwi electronics simulator rather than on physical multi-bin hardware. The results presented in Section 5.6 therefore demonstrate that the coordination logic, guardrail behaviour, parsing, and demand-planning projection operated correctly under the controlled simulated environment, at the scale of three bins. They do not, however, directly demonstrate behaviour under the physical-world conditions evaluated in Phases 1 and 2, such as sensor noise, mechanical settling effects, or real network latency, nor do they establish how the system would behave with a larger number of simultaneously coordinated bins. Validation on physical multi-bin hardware and at a larger scale would

therefore be required before these results could be generalized to a full industrial deployment.

Overall, the Phase 3 results support the third research question by demonstrating that the single-bin monitoring concept can be extended to a coordinated three-bin kitting station under the tested conditions. The results show that the extended system maintained correct operation across the tested multi-bin workflows, while the main limitation remains the simulation-based validation environment and the relatively small three-bin scale. Further validation on physical multi-bin hardware and at a larger scale would therefore be required before generalizing these results to industrial deployment.

5.8 Cross-Phase Discussion

The aim of this thesis was to investigate how intelligent real-time inventory monitoring can improve inventory visibility and automate stock verification in manufacturing kitting operations through three progressively developed prototypes. Individually, Sections 5.3, 5.5, and 5.7 discussed how each phase addressed its respective research question. Taken together, the three phases address complementary aspects of the overall aim: sensing reliability under occlusion and sensor failure (Phase 1), automated stock verification combined with natural-language operator interaction under real industrial constraints (Phase 2), and coordinated operation across multiple simultaneously monitored bins (Phase 3).

Table 19. Summary comparison across the three prototype phases.

Phase	Sensing approach	Environment	Key strength demonstrated	Key limitation identified
1	Weight + vision sensor fusion	Laboratory	Hidden-part inference (95.7% accuracy); correct detection and recovery from total sensor failure	Wrong-part classification threshold miscalibration; confidence did not track correctness uniformly for hidden-part inference
2	Weight sensing + natural-language AI assistant	Industrial pilot (Binar Solutions AB)	100% verified pick-tracking accuracy; correct guardrail behaviour and order-queue recovery workflow	AI-triggered recovery workflow incomplete; reliability affected by free-tier AI access and usage limits
3	Weight sensing, coordinated across three bins	Simulation	100% pick-tracking accuracy scaled across bins; correct multi-bin demand-planning projection and priority-order resume under repeated interruption	Simulation-only validation; limited to a three-bin scale

Beyond the results reported above, each prototype also includes supporting features that were implemented but not directly evaluated through dedicated trials in this chapter, such as monitoring-area selection and scale manual and auto-calibration in Phase 1,

and parts ordering and manual calibration in Phase 2. These features are described in Chapter 4 but are not claimed as experimentally demonstrated results here.

Across the three phases, the tested weight-based pick-tracking mechanism did not produce an incorrect quantity in any of the verified trials. Phase 1 further demonstrated that combining weight and vision sensing could support additional functions, such as hidden-part inference, while Phase 2 showed that weight-based verification could be integrated into an existing industrial pick-to-light installation. Phase 3 then extended the weight-based approach to coordinated operation across three simulated bins. The limitations identified across the phases instead arose primarily in higher-level classification, decision, recovery, and validation aspects: wrong-part classification in Phase 1, AI-triggered recovery in Phase 2, and the simulation-based validation environment in Phase 3. This progression suggests that further development should focus not only on sensing, but also on the decision and recovery layers built around the sensing results.

A second observation emerges from comparing Phase 2 and Phase 3 directly. Phase 2 relied on a generative AI assistant to interpret natural-language operator requests, and its reliability was shown to vary with the underlying language-model access used (Section 5.5). Phase 3, by contrast, used a deterministic natural-language parser without an AI model and achieved 100% correct parsing across all eight tested trials in both English and Swedish (Section 5.6.4). This represents a trade-off rather than a direct improvement: the deterministic parser provides predictable behaviour without dependence on external model access or usage limits, but its interpretation is constrained by the patterns it was designed to recognise. The AI assistant in Phase 2 provides greater flexibility in interpreting varied operator phrasing, but introduces an additional dependency on language-model behaviour and availability. The more appropriate approach therefore depends on the required balance between flexibility, predictability, and external-service dependency, and is discussed further as a future-work consideration in Chapter 6.

Finally, the sensor-fusion capability developed in Phase 1, particularly hidden-part inference, was not implemented into the industrial prototype in Phase 2, which was designed to integrate directly with Binar Solutions AB's existing pick-to-light installation. The multi-bin coordination developed in Phase 3 was evaluated separately in simulation. Together, these results provide a basis for future development in which the capabilities demonstrated in Phase 1 and Phase 3 are integrated into the industrial architecture established in Phase 2, enabling an industrial multi-bin system combining complementary sensing, AI-assisted interaction, and coordinated inventory monitoring.

5.9 Chapter Summary

This chapter presented the experimental results from the three prototypes developed in this thesis and discussed them in relation to the three research questions introduced in Section 1.3. Phase 1 demonstrated the value of combining weight and vision sensing for inventory monitoring, most notably through hidden-part inference, while also identifying specific limitations in wrong-part classification and confidence behaviour. Phase 2 demonstrated that weight-based sensing could provide independently verified stock tracking within an industrial pick-to-light installation while supporting natural-language operator interaction and defined guardrail and recovery behaviour. Phase 3 extended the monitoring approach to a coordinated three-bin kitting station, maintaining pick-tracking accuracy while demonstrating multi-bin coordination, interruption handling, and demand-planning shortage projection within a simulated environment.

Considered together, the three phases demonstrate a progression from sensing and inventory-monitoring reliability, through industrial integration and operator interaction, to coordinated multi-bin operation. The results also showed that the principal limitations identified during testing arose in higher-level classification, recovery, AI interaction, or validation conditions rather than from the underlying pick-tracking mechanism. The capabilities developed across the three phases were not combined into a single integrated prototype; in particular, the sensor-fusion capability from Phase 1 and the multi-bin coordination from Phase 3 were developed separately from the industrial implementation in Phase 2. Their integration therefore provides a basis for future development.

The following chapter presents the conclusions drawn from this work in relation to the research questions and overall aim introduced in Chapter 1, followed by the resulting directions for future work.

6 Conclusion

The aim of this thesis was to investigate how intelligent real-time inventory monitoring can improve inventory visibility and automate stock verification in manufacturing kitting operations through three progressively developed prototypes. The results presented in Chapter 5 allow each research question to be answered directly.

Regarding the first research question, combining weight and vision sensing was shown to provide stronger confidence in inventory estimates under the tested conditions and, more significantly, to enable hidden-part inference, a capability unavailable to either sensing modality independently. This addresses the gap identified in related work, where vision-based and weight-based approaches have generally been considered separately [15]. A specific limitation was identified in the calibration of the wrong-part classification threshold; this affected the classification result but had no impact on correct quantity counting or hidden-part inference in the tested configurations.

Regarding the second research question, a single weight sensor integrated into an existing industrial pick-to-light installation at Binar Solutions AB was shown to provide independently verified stock tracking while supporting natural-language operator interaction. These results demonstrate successful operation for this specific workstation and its tested operating conditions; they do not constitute evidence of readiness for broader industrial deployment, which would require validation across multiple bins, multiple operators, and sustained production use. This extends prior Poka-Yoke pick-to-light work, which confirms pick location but does not verify the quantity removed [16], by providing quantity verification and a defined recovery path when a discrepancy occurs. The reliability of the AI-assisted interaction was also shown to depend on the language-model access available during testing, reflecting a constraint of the tested AI service rather than of the underlying weight-based verification architecture.

Regarding the third research question, the single-bin monitoring approach was successfully extended to a coordinated three-bin kitting station without loss of pick-tracking accuracy. The prototype correctly handled multi-bin coordination challenges, including parsing ambiguity, guardrail scoping, interrupted-order resumption, and demand-planning projection. These results were obtained in a simulated environment rather than on physical multi-bin hardware.

Considered together, these results support the overall aim of the thesis: intelligent real-time inventory monitoring can improve inventory visibility and automate stock verification through the combination of reliable sensing, automated verification, and higher-level interaction and coordination capabilities demonstrated across the three phases. The validity of these conclusions is bounded by the conditions under which the results were obtained. The number of trials for individual test conditions was relatively small, typically ranging from four to thirty trials, which was sufficient to demonstrate correct behaviour under the tested conditions but insufficient to establish reliability rates that can be generalized to larger-scale or sustained industrial operation. In addition, the industrial results were obtained from a single deployment site, while the multi-bin results were obtained entirely in simulation. The conclusions should therefore be understood as evidence that the developed approaches functioned correctly under the tested conditions, rather than as generalized reliability claims that necessarily apply across different deployment environments, system scales, or operating conditions.

6.1 Future Work and Research

Several directions for future work follow directly from the limitations and findings identified in Chapter 5:

- **Developing an integrated industrial prototype:** The sensor-fusion capability demonstrated in Phase 1 and the multi-bin coordination developed in Phase 3 could be integrated into the industrial architecture established in Phase 2. This would provide an industrial multi-bin system combining complementary sensing, AI-assisted interaction, and coordinated inventory monitoring.
- **Extending the over-pick recovery workflow:** The needs-review and resolution workflow currently implemented for order-queue-triggered over-picks in Phase 2 could be extended to AI-triggered over-picks, providing consistent recovery behaviour across different trigger sources.
- **Evaluating higher-capacity AI access:** The AI-assisted interaction could be evaluated using a higher-tier language-model service with greater usage capacity to determine whether the variability in tool-call correctness and the brief post-restock reporting delay observed during Phase 2 testing are reduced.
- **Recalibrating wrong-part classification:** The wrong-part classification threshold identified in Phase 1 could be recalibrated and the classification performance re-evaluated under the corrected threshold.
- **Validating multi-bin operation on physical hardware:** The coordination logic developed in Phase 3 could be implemented and evaluated on physical multi-bin hardware and extended beyond the three-bin configuration tested in simulation. This would allow the effects of physical sensor behaviour, mechanical conditions, and communication delays to be evaluated at a larger scale.
- **Investigating hybrid interaction:** A hybrid interaction approach could be investigated that combines the flexibility of AI-assisted natural-language interpretation demonstrated in Phase 2 with the speed and predictability of deterministic parsing demonstrated in Phase 3. This could provide broader language flexibility while reducing dependence on external language-model access for predefined or safety-critical commands.

6.2 Critical Discussion

Working on this thesis across three different settings - a laboratory prototype, an industrial pilot at Binar Solutions AB, and a simulated multi-bin environment required adapting both the technical approach and the evaluation strategy at each stage. Several practical constraints became apparent only during implementation and influenced the subsequent design decisions.

Phase 1 began with exploring several sensing approaches, including ultrasonic, time-of-flight, and 2D camera sensing, alongside weight sensing, before settling on the combined weight-and-vision design. Weight-sensor calibration required several iterations before measurements could be reliably converted into part counts, and camera placement and lighting also required repeated adjustment before a reliable combined-sensor setup was achieved.

Phase 2 developed from an opportunity to visit Binar Solutions AB and observe its existing manual pick-to-light process, in which an operator picks a part and confirms the pick using a button. This led to the idea of automating pick confirmation while retaining the existing installation. Integrating the sensor into the existing bin was the

key challenge: a custom load-cell mounting plate required several iterations before it fit the bin and gave reliable measurements. The Phase 1 sensor-fusion approach could not be carried into this installation because of insufficient lighting and insufficient spacing between bins for a camera, and this was left as a direction for future development. Once automated pick detection was established, the work was extended to natural-language interaction, resulting in the kitting assistant and subsequently, order-queue handling, and stock verification.

After the single-bin system was implemented, the next challenge was extending the approach to multiple bins. Rather than building additional physical hardware, simulation was used to investigate the multi-bin architecture: Tinkercad Circuits, MATLAB Simulink, Multisim, and LTspice were considered but did not adequately represent the sensors required for this project; Wokwi, an online electronics simulator was selected because it supported the required combination of simulated components. Development began with a single simulated bin before being extended to three bins, with the main technical challenge being to ensure that activity on one bin did not incorrectly affect the state of another, requiring several iterations of the control logic. The multi-bin prototype was then extended with natural-language kit instructions, parts shortage projection, and priority-order handling, allowing an urgent order to interrupt an active order and the original order to resume correctly after the priority order was completed.

Overall, the development process showed that the final system architecture was shaped not only by the intended functionality but also by the practical constraints of each development environment.

6.3 Generalization of the result

The core approach developed in this thesis, particularly weight-based stock verification combined with guardrail-driven discrepancy detection and recovery, is transferable to other manual or semi-automated kitting and assembly environments where operators pick physical components from defined locations. The specific calibration values, classification thresholds, sensor configurations, and natural-language expressions would need to be adapted to the parts, hardware, and workflow of each deployment. However, the underlying approach of verified sensing, distinction between confirmed and uncertain picks, and defined discrepancy recovery is not dependent on the specific parts or bin configuration used in this study. The results therefore provide a basis for applying the approach to similar inventory-monitoring and kitting scenarios, although further validation would be required before generalizing the measured performance to other environments.

The main distinction of this study lies in the way these capabilities were developed and evaluated across the three phases. The related work reviewed in Chapter 2 generally considers sensor fusion, Poka-Yoke-based pick confirmation, and AI-assisted interaction as separate areas. This thesis instead investigated these complementary capabilities progressively, from laboratory-based sensor fusion, through an industrial AI-assisted pick-to-light deployment, to simulated multi-bin coordination. The contribution is therefore not based on a single novel technique in isolation, but on demonstrating how these complementary approaches can be developed and connected as part of a broader intelligent inventory-monitoring architecture.

7 References

[1] F. Akeem and R. Sonu, "Design and Implementation of an IoT-Enabled Real-Time Inventory Tracking System for Manufacturing Industries."

[2] V. Krishna and P. Mattegunta, "Retail-Time Inventory Visibility: Transforming Retail Operations Through Real-Time Data Integration," 2025, doi: 10.32996/jcsts.

[3] S. Paul, A. Chatterjee, and D. Guha, "STUDY OF SMART INVENTORY MANAGEMENT SYSTEM BASED ON THE INTERNET OF THINGS (IOT)," 2019.

[4] V. Nandhini, B. Senthil Kumar, R. R. Al-Fatlawy, E. Poornima, S. S. Afreen, and Z. Atamuratova, "Smart IoT-Enabled Real-Time Inventory Management System (SIRIMS) for Retail," in *2025 International Conference on Metaverse and Current Trends in Computing, ICMCTC 2025*, Institute of Electrical and Electronics Engineers Inc., 2025. doi: 10.1109/ICMCTC62214.2025.11196150.

[5] H. Brynz~r A'b' and M. I. Johansson, "journal of production economlcs Design and performance of kitting and order picking systems," 1995.

[6] Y. Kang and S. B. Gershwin, "Information inaccuracy in inventory systems: Stock loss and stockout," *IIE Transactions (Institute of Industrial Engineers)*, vol. 37, no. 9, pp. 843–859, Sep. 2005, doi: 10.1080/07408170590969861.

[7] S. Bousselmi, M. Gannouni, and K. Ouni, "IoT Application for Smart Inventory Management System Based on RFID," in *2024 IEEE International Multi-Conference on Smart Systems and Green Process, IMC-SSGP 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/IMC-SSGP63352.2024.10919846.

[8] S. G. Kumar, S. Prince, and B. M. Shankar, "Smart tracking and monitoring in supply chain systems using RFID and BLE," in *2021 3rd International Conference on Signal Processing and Communication, ICPSC 2021*, Institute of Electrical and Electronics Engineers Inc., May 2021, pp. 757–760. doi: 10.1109/ICSPC51351.2021.9451700.

[9] L. Fu Yuan, Y.-H. Choo, and A. Kamilah Muda, "SMARTSTOCK: Real-Time AI-Powered Vision System for Automated Warehouse Inventory Management", doi: 10.47772/IJRISS.

[10] P. Meenalochini, "Smart Robotic Framework for Automated Pick-and-Place and Real-Time Inventory Management in Industry 4.0 Environments 1," 2025.

[11] V. S. Yosephine, M. Batara, and M. Setiawati, "Scalable and Affordable IoT-based Inventory Control with Real-Time Monitoring for Small and Medium Enterprises," *Jurnal Teknik Industri: Jurnal Keilmuan dan Aplikasi Teknik Industri*, vol. 27, no. 1, pp. 121–136, May 2025, doi: 10.9744/jti.27.1.121-136.

[12] K. P. Kshirsagar, M. I. Badwe, A. A. Gadpale, R. S. J. S. Avdeel, A. B. Bachchhav, and A. S. Awachat, "RFID And Ultrasonic Sensor based Dualmodule Inventory Management System with Two Layer Scrutiny," in *2025 Third International Conference on Emerging Applications of Material Science and Technology (ICEAMST)*, IEEE, Nov. 2025, pp. 393–399. doi: 10.1109/ICEAMST67459.2025.11336099.

[13] G. Manoharan, V. Kumar, A. Karthik, V. Asha, C. R. Mohan, and G. Nijhawan, "IoT-Based Smart Inventory Management System Using Machine Learning

Techniques," in *Proceedings - 3rd International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2024*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/ACCAI61061.2024.10601903.

[14] S. Omoleye, "INTEGRATION OF COMPUTER VISION AND IOT SENSORS FOR REAL-TIME PERFORMANCE MONITORING IN SMART MANUFACTURING SYSTEMS."

[15] R. Lin, "Multi-sensor fusion for inventory monitoring."

[16] J. Trojanowska, J. Husár, S. Hrehova, and L. Knapčíková, "Poka Yoke in Smart Production Systems with Pick-to-Light Implementation to Increase Efficiency," *Applied Sciences (Switzerland)*, vol. 13, no. 21, Nov. 2023, doi: 10.3390/app132111715.

[17] C. Stockinger, T. Steinebach, D. Petrat, R. Bruns, and I. Zöller, "The effect of pick-by-light-systems on situation awareness in order picking activities," in *Procedia Manufacturing*, Elsevier B.V., 2020, pp. 96–101. doi: 10.1016/j.promfg.2020.04.078.

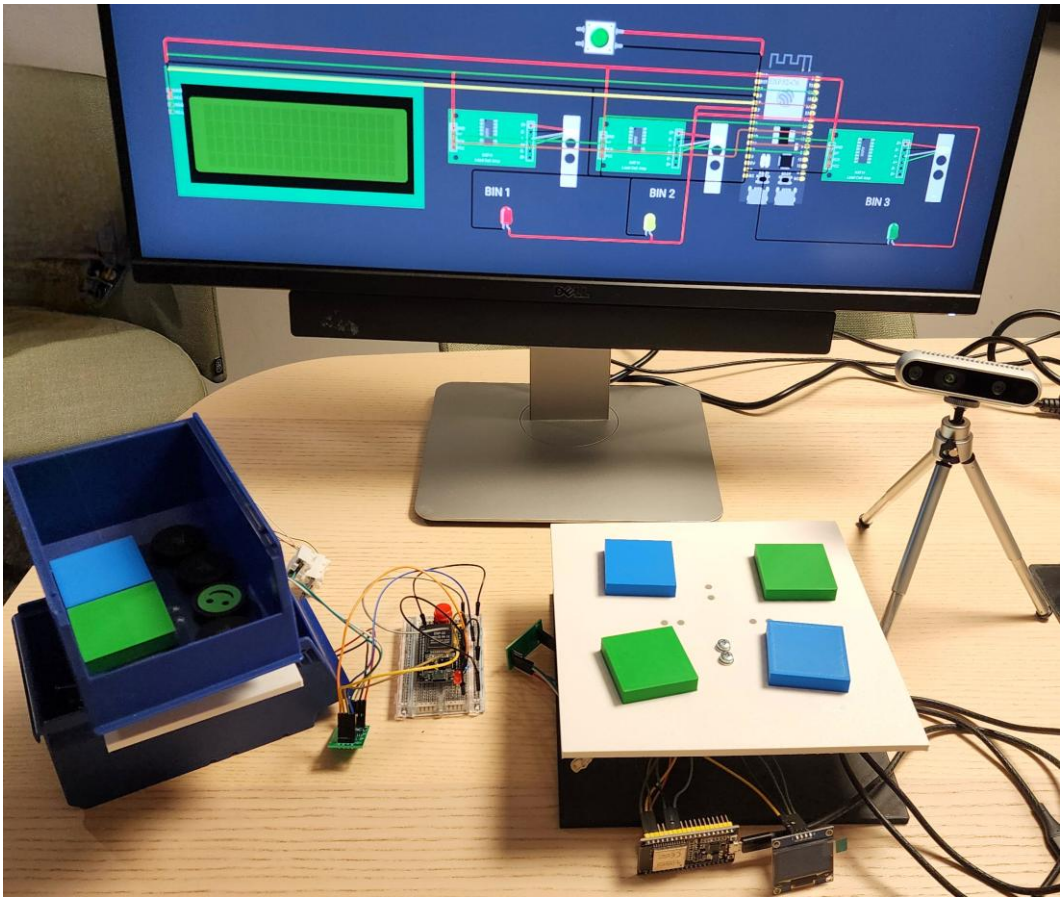
[18] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents." [Online]. Available: https://github.com/YuchenXia/GPT4IndustrialAutomation

[19] Y. Xia, N. Jazdi, J. Zhang, C. Shah, and M. Weyrich, "Pre-print accepted at 30th IEEE ETFA 2025 Control Industrial Automation System with Large Language Model Agents." [Online]. Available: https://github.com/YuchenXia/LLM4IAS.

[20] M. Wang *et al.*, "Function Calling in Large Language Models: Industrial Practices, Challenges, and Future Directions," *ACM Comput. Surv.*, vol. 58, no. 9, Jul. 2026, doi: 10.1145/3788284.

[21] A. R. Hevner, S. T. March, J. Park, and S. Ram, "DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH 1," 2004.

A. Physical Prototype - Laboratory Hardware and Multi-Bin Simulation



FigureA.1 Phase 1 and Phase 2 laboratory hardware alongside the Phase 3 multi-bin simulation

B. Physical Prototype - Industrial Installation at Binar Solutions AB

FigureB.1 The system integrated into Binar Solutions AB's existing Poka-Yoke pick-to-light installation (Phase 2).