ДОДАТОК А

Апробація результатів

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

Comparative analysis of modern SCADA packages for production automation

Maksym Danylenko1, Svitlana Sotnik2

1Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
e-mail: maksym.danylenko@nure.ua
2Department of Computer-Integrated Technologies, Automation and Mechatronics, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
e-mail: svetlana.sotnik@nure.ua

Abstract: *The work reveals the main capabilities and conceptual aspects of modern SCADA packages, which are widely used to automate production processes. A detailed study of the functional characteristics of three wired systems has been carried out: InTouch, iFix and WinCC. A comparison of their technical capabilities, architecture, supported protocols, visualization tools, security mechanisms and economic aspects of implementation is carried out. Particular attention is paid to the analysis of potential risks and threats associated with the implementation of SCADA systems in production. Technical integration challenges, cybersecurity issues, organizational risks and compatibility problems with existing equipment are considered. On the basis of the study, strategies for minimizing risks are proposed, and practical recommendations for choosing the optimal system depending on the specifics of the enterprise are developed. The results of the study are of practical value for enterprises planning to modernize production or introduce new automation systems. The presented materials can be used to make a reasonable choice of an SCADA system and develop an effective strategy for its implementation, taking into account the potential risks and features of a particular production.*

Keywords—system; SCADA; review; advantages; disadvantages

1. Introduction

Automation and robotization provide faster data collection and accurate processing, which allows reducing the human factor and increasing the efficiency of modern industries in any industry [1-5].

SCADA systems (Supervisory Control and Data Acquisition) are an integral part of modern automation technologies that allow you to effectively control and manage production, energy, transport and other complex technical processes. Their relevance in the modern world is due to the rapid development of technologies, the growing demand for effective resource management, and the increase in the level of requirements for the reliability and safety of technological processes.

One of the key factors in the relevance of SCADA systems is their ability to provide continuous monitoring and control of technological processes in real time. This is especially important for enterprises in the energy sector, water supply, oil and gas industry, and other industries, where even minor failures can lead to significant economic losses or dangerous situations. SCADA allows you to quickly identify and eliminate deviations from normal functioning, thereby minimizing risks and increasing the efficiency of systems.

Another important aspect is the integration of SCADA systems with modern Internet of Things (IoT) and cloud computing technologies. This provides the ability to analyze large amounts of data, predict the operation of systems, and optimize processes. For example, in industry, SCADA systems help analyze energy consumption, identify weaknesses in production cycles, and reduce costs by automating processes.

Security is another important factor in the relevance of SCADA systems. With the increasing number of cyberattacks on critical infrastructure, SCADA systems are constantly being improved to ensure a high level of data protection and communications reliability. Developers are implementing new methods of authentication, data encryption, and intrusion protection, making these systems more resilient to external threats.

It is also worth noting the role of SCADA in promoting sustainable development. With the ability to accurately track the consumption of energy, water and other resources, these systems help reduce their use and reduce their negative impact on the environment. For example, in water supply, SCADA allows you to reduce water losses, and in the energy sector, it can increase the efficiency of energy generation and distribution.

Therefore, the introduction into production really opens up wide opportunities for automation, increasing efficiency and optimizing processes. However, this process has its own challenges that require attention.

Thus, the problem of studying the opportunities and risks of integrating SCADA packages into production is extremely relevant, since it includes a comprehensive analysis of the advantages and challenges that arise when implementing the latest ones.

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

The purpose of the study is to study and analyze the main features of using SCADA packages.

Therefore, to achieve this goal, the following tasks are provided:

- overview of the main functionality of SCADA packages;
- evaluate the advantages of SCADA packages;
- analyze SCADA packages with each other;
- investigate the main risks and threats associated with SCADA packages for production;
- strategies for minimizing the risks of implementing SCADA packages;
- recommendations for choosing a system.

2. Related Work

Research on the analysis and comparison of SCADA packages is an important part of modern approaches to automation of production processes. This part of the work discusses existing scientific works that highlight the features, advantages and disadvantages of various SCADA systems, as well as their application in industrial conditions.

Considerable attention is paid to comparing functionality, reliability, scalability, compatibility with industry standards, and the cost of implementing such systems. In particular, the literature analyzes popular SCADA packages, such as WinCC, InTouch, Master Scada, Trace Mode.

In [6], we are talking about SCADA systems operating on the basis of web-based SCADA technologies. Such systems allow you to remotely monitor and manage production processes through web browsers on computers or mobile devices. The article summarizes the research of the platform software (PS) for SCADA and offers ideas for its development. The main emphasis is on studying current trends and technologies for creating platforms that ensure data security, simplify the maintenance and expansion of systems, and contribute to the integration of key technologies of the fourth industrial revolution into industry. This, in turn, helps companies in digital transformation and optimization of production processes.

The work [7] is devoted to the analysis of modern SCADA systems, which are key for the automated management of critical infrastructures. The authors focus on studying the vulnerabilities of these systems that arise due to their connection to the Internet and integration with cloud technologies and the Internet of Things (IoT). The work covers an overview of existing SCADA architectures, analyzes real-world attacks on these systems, and evaluates threat detection methods and test environments to investigate them. At the end of the article, the main problems that need to be solved to improve the security of SCADA systems in the future are considered.

The paper [8] analyzes the vulnerabilities of SCADA systems, which are the basis of critical infrastructures such as water supply, energy, and transport. The study covers SCADA architecture, attack types, intrusion detection techniques (IDS), and test environments. The authors proposed a classification of vulnerabilities, threats, and IDS and pointed out key challenges and open issues in the field of SCADA security for further research.

Many scientific papers emphasize the advantages of using SCADA systems to increase the efficiency of production [9-14]. For example, a study [9] highlights the use of SCADA systems on wrapping machines using PLC and Wonderware InTouch in car manufacturing. The work demonstrates the integration of an SCADA system with a database for data collection and processing, which is an important step for the automation of wrapping machines in industry.

[10] describes the development and implementation of the InTouch SCADA system to automate the production process of tires for two-wheeled vehicles in the context of the development of the Industry 4.0 concept. The article presents an SCADA system designed to monitor and control 6 vulcanization machines using HMI Wonderware InTouch for control and visualization. The control system uses a Mitsubishi PLC with an Ethernet module QJ71E71-100 for communication, and the database was created in Microsoft Access.

[11] describes the development and implementation of an automated production line control system for sorting and adjusting parts using WinCC software.

[12] also describes the effective implementation of WinCC to control the production line for sorting and adjusting parts.

As for the effective implementation of iFIX in production, then, in the work [13], the authors successfully describe the development of a warning signal system for an automated cigarette production line based on PLC and iFIX software.

In the work [14], the iFIX 6.5 software was used to monitor and control the operation of solar panels in real time. It was used as part of a SCADA system to automate the process, where the condition of the panels was monitored, potential malfunctions were detected using predictive diagnostics, and the panels were automatically shut down when symptoms of future malfunctions were detected. iFIX 6.5 allows you to provide reliable control and management of solar panels, as well as an interface for users that allows you to effectively monitor the status of the system.

In [15], an SCADA system was used to effectively control and monitor the entire production line for the production of stainless pipes. It provides automation of data collection, processing and analysis of information, which allows you to increase production efficiency and the level of management. The SCADA system was developed in the C# language.

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

There are also works that investigate specific aspects of SCADA systems in critical infrastructure. For example, in the work [16], the SCADA system was used in the context of IoT infrastructures for critical industries such as energy and water supply, where devices such as smart meters and water valves are used. The article focuses on the use of SCADA to control and monitor these infrastructures, which are becoming increasingly complex due to the vast amount of data from sensors and control devices.

Protection of critical infrastructure networks for smart grids, SCADA and other industrial control systems is presented in [17].

Many studies mainly consider certain aspects of SCADA systems, their functionality and applications in various fields, in particular in the automation of production processes, critical infrastructure monitoring and security. However, a detailed comparison of SCADA systems is lacking. In addition, not enough attention has been paid to the main risks associated with the introduction of SCADA packages into production.

Thus, the literature review indicates the need for further research aimed at a comprehensive comparison of SCADA systems, taking into account both the possibilities and risks of their implementation.

3. Analysis of the Main Capabilities of SCADA Packages

In this section, we will review and analyze the main features of SCADA packages that are used.

Thus, let's consider the 3 most influential SCADA systems.

Let's start with the InTouch SCADA system from Wonderware (AVEVA), which is one of the most common and popular systems for controlling, monitoring and visualizing production processes at enterprises. It is designed to create human-machine control (HMI) interfaces and is used in various industries such as energy, food processing, oil and gas, water supply, chemical production, and many others. InTouch provides a convenient and reliable way to track processes, collect and analyze data, and manage equipment in real time (Fig. 1).

The InTouch system stands out for its scalability, flexibility and ease of integration with other automation systems. It supports a wide range of industrial communication protocols and standards, such as OPC, Modbus, DDE, Ethernet/IP, and others, making it easy to interact with PLCs (Programmable Logic Controllers), sensors, actuators, and other devices. The software has a user-friendly graphical interface for creating control panels and mnemonic diagrams, which can include a variety of visualization elements, such as graphs, charts, animated objects, and status indicators.

Fig. 1. *Settings windows in the InTouch system*

One of the main advantages of InTouch is its ease of use thanks to its intuitive development environment. Even users with minimal coding skills can quickly create graphical screens and scripts to automate processes. At the same time, the system supports complex scenarios for developers who require a high level of customization and integration. To do this, the QuickScript scripting language is used, which allows you to create management logic, interact with data, and implement custom functions.

InTouch provides powerful tools for collecting, processing, and archiving data. It can work with historical data, recording all events, parameter changes, and emergencies for further analysis and reporting. This makes the system indispensable for analyzing trends, finding anomalies, and optimizing processes. Additionally, the system supports integration with databases such as Microsoft SQL Server, allowing you to store large amounts of data and use it for business intelligence.

One of the key features of InTouch is the ability to work with remote clients. Thanks to web technologies and mobile applications, operators can access the system in real time from anywhere in the world. This allows flexibility and mobility in the management of production processes, which is especially important for large enterprises with distributed production sites. The system also supports alarm and alert functions, which allows you to quickly respond to emergencies and minimize downtime or loss.

Security in SCADA systems plays a critical role, and InTouch provides a wide range of data protection and access control. It supports user authentication, differentiation of access rights, as well as the ability to keep audit logs, which allows you to track all the actions of operators and administrators. This ensures compliance with modern cybersecurity standards and reduces the risks of unauthorized access to the system.

Due to its reliability, scalability, and ease of configuration, InTouch remains one of the leaders in the SCADA solution market. It allows enterprises to optimize production processes, reduce operating costs, increase resource efficiency, and improve product quality. This solution is suitable for both small enterprises and large corporations that

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

need comprehensive monitoring and management of their production assets.

Next, let's take a closer look at iFIX because it is a powerful and flexible SCADA system developed by GE Digital (General Electric), which is used to monitor, control and automate production and technological processes. It is widely used in various industries, such as energy, chemical production, the food industry, water supply, oil and gas processing, as well as in pharmaceuticals and transport. The main tasks of iFIX are process visualization, data collection, processing and analysis, real-time equipment management, as well as optimization of technological processes to increase their efficiency and safety (Fig. 2).

Fig. 2. *Settings windows in the iFIX system*

One of the key features of iFIX is its modular architecture, which allows you to scale the system from small local solutions to large distributed network systems. It supports integration with a large number of hardware through standardized communication protocols such as OPC, Modbus, DNP3 and BACnet, making it easy to connect PLCs, process controllers, sensors and other automation devices. In addition, the system can work with databases such as Microsoft SQL Server to store historical data and generate reports.

iFIX offers a user-friendly graphical environment for creating human-machine control (HMI) interfaces. Development tools allow you to create intuitive mnemonic diagrams that reflect the state of equipment, process parameters, graphs, diagrams, and other visual elements. With support for animation and dynamic objects, operators can monitor equipment operation in real time, track parameter changes, and detect deviations. In addition, iFIX provides functionality for creating scripts in the VBScript language, which makes it possible to implement complex control and automation logic.

The iFIX system has powerful capabilities for data collection and analysis. It supports working with real and historical data, which allows you not only to monitor the current state of processes, but also to analyze trends, identify anomalies, and predict potential problems. Reporting functionality and trend graphs provide users with detailed information to make informed decisions. In addition, the system can integrate with other GE Digital solutions, such as Historian, to efficiently store and analyze large amounts of data.

Separately, the alert and alarm system in iFIX should be noted. It allows you to configure flexible algorithms for alerts about emergency situations and deviations of parameters, as well as keep a log of events and user actions. This provides a high level of control over critical processes and minimizes response time to problems.

From a security point of view, iFIX supports modern mechanisms for user authentication and authorization, which allows you to differentiate access to the system depending on roles and powers. It also supports data encryption and keeps audit logs to ensure compliance with cybersecurity standards. This makes the system suitable for use in critical industries where data protection issues are a priority.

An important advantage of iFIX is its integration with modern technologies such as cloud services and mobile devices. This allows operators to access data and monitor processes from anywhere in the world through web browsers or mobile applications. Thanks to this, enterprises get more opportunities for remote management, increasing the efficiency of their work.

Thus, iFIX is a universal solution for monitoring, managing and automating production processes, which combines high functionality, reliability and ease of use. Its modularity, scalability, and support for modern technologies make this system the best choice for businesses of various sizes and industries looking to increase their efficiency, reduce costs, and ensure high product quality.

And another most common system is WinCC (Windows Control Center) - this is a powerful and flexible SCADA system developed by Siemens that is designed to monitor, control and automate industrial and technological processes. It is widely used in various industries, including mechanical engineering, energy, chemical manufacturing, food processing, water supply, and transportation and construction. WinCC combines functionality for data collection, visualization, analysis, and archiving, providing effective real-time management of production processes.

The main features of WinCC are:

- WinCC supports both small on-premises automation systems and large distributed networks with thousands of data collection points. It can work as a standalone application on one PC or as a client-server system for several operator stations. Thanks to the modular architecture, the system is easily expandable and customizable to specific user requirements;
- WinCC is compatible with a large number of industrial communication protocols such as OPC, Modbus, Profibus, Profinet, DNP3, and BACnet, allowing for easy integration with PLCs, controllers, sensors, and other automation devices. In addition, it supports integration with databases

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

(SQL Server) and ERP systems (e.g., SAP) for data exchange at the enterprise management level;

• WinCC offers an intuitive graphical environment for creating human-machine control (HMI) interfaces (Fig. 3). Operators can use mnemonic diagrams, graphs, diagrams, and animated objects to visualize production processes. With support for animations and dynamic elements, WinCC provides convenient control and monitoring of the state of equipment and processes;

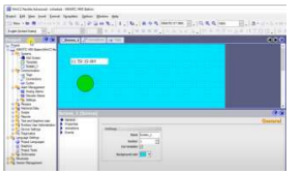

Fig. 3. *Configuration window in the WinCC system*

• WinCC has built-in tools for collecting, archiving, and analyzing historical data. It supports working with large amounts of information and provides detailed reports, graphs, and trends. This allows you to analyze production parameters, identify anomalies and predict possible equipment malfunctions. Archives can be stored in standard databases, which facilitates their further analysis;

• WinCC includes a powerful alarm management system. It allows you to set up notifications about parameter deviations or emergencies, as well as keep logs of events and actions of operators. Messages can be received via SMS, email or mobile applications, which allows you to quickly respond to critical situations;

• WinCC supports work through web interfaces and mobile devices, which allows operators and engineers to control processes from anywhere in the world. This is especially useful for businesses with geographically distributed facilities where centralized control and management are required;

• the system meets modern cybersecurity requirements. It provides access rights differentiation, user authentication, data encryption and audit logging. This allows you to protect the system from unauthorized access and increase the level of security in critical industries such as energy and transportation;

• WinCC supports the use of scripts based on VBScript and ANSI C, which makes it possible to configure complex control and automation algorithms. This makes the system flexible to solve non-standard tasks and adapt to specific processes.

WinCC is part of the Totally Integrated Automation (TIA) Portal ecosystem, enabling seamless integration with other Siemens products such as SIMATIC S7 series PLCs, operator panels, drive systems and remote access tools. This allows you to create a single environment for managing and monitoring all aspects of production, from the lower level of automation to the upper level of control.

Types of WinCC solutions:

• WinCC Runtime Advanced – a solution for small and medium-sized HMI systems running on a single PC or operator panel;

• WinCC Runtime Professional – for large-scale SCADA systems with the ability to work in client-server mode;

• WinCC V7 is a classic version for building large distributed SCADA systems;

• WinCC Unified is a modern cloud solution with the ability to integrate with IoT and big data analytics.

4. Comparison of SCADA Packages

This section compares the most common SCADA packages used to automate production processes [18, 19]. The goal is to analyze their functionality, compatibility and cost-effectiveness in order to provide recommendations for choosing the optimal solution for specific conditions. The comparison is based on key criteria such as functionality, cost of ownership, support and technical assistance, as well as the level of security, etc.

Let's analyze the characteristics and functionality of SCADA packages (Table 1).

Table 1: Analysis of characteristics and functionality of SCADA packages

#	*Criterion*	*InTouch*	*iFix*	*WinCC*
1	Developer	AVEVA	GE Digital	Siemens
2	Scalability	Small and large systems	Local and distributed	Single to large networks
3	Supported protocols	OPC, Modbus	OPC, Modbus, BACnet	OPC, Profibus
4	Interface and visualization	Graphically intense	Powerful	Intuitive
5	Alarms and notifications	SMS, email	SMS, email	SMS, email
6	Remote access	Web access, mobile	Web access	Web access
7	User authentication	Supports multi-level authentication; integration with Windows Active Directory (AD).	Multi-level authentication; integration with AD.	AD support; the ability to customize roles.
8	Access control (role-based model)	Flexible distribution of access rights; separation of actions for different roles.	Advanced role model for users.	High granularity of access rights for objects.
9	Data encryption	You can set up a secure channel via Windows Communication Foundation (WCF) that supports TLS/SSL.	Data encryption in communication channels.	Encryption for communication via OPC UA and other protocols.
10	Protection against cyber attacks	Built-in security features, compatible with AVEVA cybersecurity products.	Support for cybersecurity standards; integration with antiviruses.	Comprehensive protection against attacks; IEC 62443 certification.
11	Compatibility with cybersecurity standards	Compliance with ISA/IEC 62443 standards.	Compliance with ISA/IEC 62443 standards.	Compliance with ISA/IEC 62443 standards.
12	Programming	QuickScript (own scripting language), support for VBS, .NET	Visual Basic for Applications (VBA), pidtrymka C# via .NET	VBS, ANSI C, C++ через ODK (Open Developer Kit), VBA, ScriptsImprove
13	Integration	ERP, MES, SQL, OPC UA/DA, Historian, .NET, REST API, databases (Oracle, MS SQL), SAP	ERP, MES, OPC UA/DA, Historian, REST API, databases (Oracle, MS SQL), SAP, integration with other GE products	TIA Portal, ERP, MES, OPC UA/DA, databases (Oracle, MS SQL), SAP, REST API, Plant Intelligence, integration with S7
14	Cloud solutions and IoT	Partial support	Partial support	Full support
15	Cost	Medium-high	Medium	High

As for the cost, this is a subjective assessment, which may vary depending on: region, specific configuration, number of licenses and special conditions from suppliers.

Analysis of Comparative Table 1 of SCADA packages shows that all three solutions reviewed – InTouch by AVEVA, iFix by GE Digital and WinCC by Siemens – are professional and reliable high-level systems.

In terms of basic functionality, these systems are quite similar to each other. They support major industrial protocols, including OPC, have built-in SMS and email notification capabilities, provide web access, and offer robust security systems with data encryption.

However, there are certain differences between them. WinCC from Siemens stands out for the most comprehensive support for cloud solutions and IoT technologies, but at the same time has the highest cost. InTouch is characterized by powerful graphics capabilities and is offered at a medium-high price. iFix has good integration with other GE products and is positioned in the middle price range.

The choice of a specific system largely depends on the existing infrastructure of the enterprise, the available budget, the scale of the project, and the specific functionality requirements. For example, if a business already uses Siemens equipment, then WinCC may be the most logical choice, despite the higher cost. For projects with a limited budget or without the need for advanced functionality, iFix may prove to be the optimal solution.

Each of these SCADA systems has its own strengths and may be the best choice in a particular situation, depending on individual needs and implementation conditions.

5. Main Risks and Threats Associated With the Implementation of SCADA Packages in Production

When working with InTouch, it may be difficult to integrate legacy hardware and optimize the graphical interface for large systems.

iFix exhibits certain limitations when scaling and may require additional resources to ensure fault tolerance.

WinCC, despite its reliability, can cause problems when integrating with third-party hardware.

Among the key challenges, cybersecurity can be highlighted. Despite the built-in protection mechanisms, each system has its own vulnerabilities. InTouch needs regular updates to protect against new threats and carefully configure WCF channels. iFix may have security difficulties when integrating with older systems. WinCC, although IEC 62443 certified, requires constant monitoring and updating of security policies.

Also, we will highlight organizational challenges, namely, the human factor. The implementation of any SCADA system requires proper training of personnel. Employees must master specific programming languages: QuickScript for InTouch, VBA for iFix, or ANSI C for WinCC. Insufficient qualifications can lead to configuration errors and system vulnerabilities.

We will also include economic aspects as important challenges. Every system has hidden costs. InTouch and iFix may require additional integration and scaling costs. WinCC, having the highest initial cost, also requires a significant investment in staff training and support.

It is important to pay attention to operational risks such as reliability and fault tolerance. All three systems have different approaches to ensuring fault tolerance. InTouch can have problems when working with large amounts of data. iFix sometimes shows instability in complex configurations. WinCC can put additional strain on the network infrastructure.

And, let's pay attention to compatibility and integration in the same way that each system has its own integration features:

• InTouch may have difficulties with some protocols and databases;

• iFix sometimes shows problems when integrating with cloud services;

• WinCC can be difficult when working with non-Siemens equipment.

So, let's offer strategies for minimizing the risks of implementing SCADA packages:

1. Development of a comprehensive plan for personnel training.

2. Conducting a detailed audit of the existing infrastructure before choosing a system.

3. Creation of backup systems and disaster recovery plans.

4. Regular security audits and system updates.

5. Phased implementation with testing at each stage.

When choosing a system, it is worth considering:

• InTouch is optimal for medium-sized enterprises with the need for powerful visualization;

• iFix is suitable for enterprises with limited budgets and existing GE infrastructure;

• WinCC is best suited for large enterprises, especially with Siemens equipment.

6. Conclusion

In this study, a comprehensive analysis of three leading SCADA packages was carried out: InTouch, iFix and WinCC. The main results of the study are:

1. A detailed analysis of the functionality of each system was carried out: InTouch by AVEVA with an emphasis on graphical visualization and scalability; iFix by GE Digital, with its modular architecture and integration capabilities; WinCC from Siemens with full support for cloud technologies and IoT.

A comparative analysis of the systems was performed according to 15 key criteria, including scalability, supported protocols, security, programming and cost. This made it possible to determine the strengths and characteristics of each system.

2. The main challenges (risks) of the implementation of SCADA packages have been identified and analyzed, including: technical challenges in integration and scaling; cybersecurity issues; organizational risks associated with the human factor; economic aspects of implementation; operational risks and compatibility issues.

Practical recommendations have been developed on: strategies for minimizing risks during implementation; selection of the optimal system depending on the specifics of the enterprise; planning the implementation process and staff training.

The practical value of the study lies in the fact that its results can be used by enterprises in the selection and implementation of SCADA systems, which will reduce risks and optimize the cost of automation of production processes.

As a result, the key result of the study was the creation of a comprehensive comparative characteristic of three leading SCADA packages (InTouch, iFix and WinCC), which allows a reasonable approach to the choice of the optimal automation system depending on the specifics of the enterprise. The analysis covers not only the technical characteristics of the systems, but also the economic aspects of implementation, potential risks and ways to minimize them. The results of the study are of practical value for enterprises planning to modernize or implement new systems for automation of production processes.

7. References

[1] Andreiev, A.S., et al. (2024). Analysis of robotics platforms for educational and research purposes. Kompiuterni ihry ta multymedia yak innovatsiinyi pidkhid do komunikatsii - 2024 / Materialy IV Vseukrainskoi naukovo-tekhnichnoi konferentsii molodykh vchenykh, aspirantiv i studentiv, Odesa, 26-27 veresnia 2024 r., 25-27

[2] Polikanov, K., et al. (2024). Smart home with house module: overview of automation technologies. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», 20-21

[3] Khalimonov, Y., et al. (2024). Approaches to ensuring proper working conditions using sensor technologies IoT. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», 24-25

[4] Sotnik, S. V. (2024). Development of automated control system and registration of metal in continuous casting// Radio Electronics, Computer Science, Control, 3, 197-211

[5] Marunich, R., et al. (2024). Approaches to ensuring the effective implementation of iot technologies in various industries. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», pp. 22-23

[6] Chi, C. P., et al. (2024). Platform Software for Building Web SCADA Applications. Creative Approaches Towards Development of Computing and Multidisciplinary IT Solutions for Society, 529-548

[7] Yadav, G., Paul, K. (2021). Architecture and security of SCADA systems: A review. International Journal of Critical Infrastructure Protection, 34, 100433

International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 9 Issue 2 February - 2025, Pages: 26-34

[8] Alanazi, M., et al. (2023). SCADA vulnerabilities and attacks: A review of the state-of-the-art and open issues. Computers & security, 125, 103028

[9] Ardi, S., et al. (2024). The application of scada systems on wrapping machines using plc and wonderware intouch in the automotive manufacturing industry. Jurnal Pendidikan Teknologi dan Kejuruan, 30.2, 297-309

[10] Ardi, S., et al. (2020). Design of Monitoring and Control of SCADA Systems on Curing Machine using PLC and HMI Wonderware InTouch. FoITIC, 200-206

[11] Cao, Yi, et al. (2024). Parts sorting and adjustment automatic production line control system based on PLC and WinCC. Proceedings of The 4th International Conference on Computer, Internet of Things and Control Engineering, 97-101

[12] Ershov, V. A., et al. (2021). Process control system for producing a silica-based modifying additive. Metallurgis, 65, 446-453

[13] Si, X., et al. (2024). Design and application of quality early warning system based on PLC and iFIX learning. Fourth International Conference on Sensors and Information Technology (ICSI 2024), Vol, 13107, 1-8

[14] García, E., et al. (2022). Solar panels string predictive and parametric fault diagnosis using low-cost sensors. Sensors, 22.1, 332, 1-29

[15] Chen, F., et al. (2020). Design of SCADA System for Stainless Steel Pipe Production Line. 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). IEEE, 1-6

[16] Baker, T., et al. (2020). A secure fog-based platform for SCADA-based IoT critical infrastructure. Software: Practice and Experience, 50.5, 503-518

[17] Knapp, E. D. (2024). Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems. Elsevier, 483

[18] Sotnik, S. V. (2024). Development of automated control system for continuous casting. Radio Electronics, Computer Science, Control, 2, 181-189

[19] Lashyn, Z. V., et al. (2024). Automation capabilities of equipment with built-in robot for manufacture of microelectronics products. Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024», 283-286

ДОДАТОК Б

Код програми conveyor.py

```
…
import bpy
from mathutils import Vector
import blf

# Глобальні налаштування коефіцієнтів
TEXTURE_SPEED_MULTIPLIER = 0.11  # Текстура рухається повільніше для синхронізації
KRISHKA_SPEED_MULTIPLIER = 1.0  # Швидкість руху банки
warning_text = ""
show_warning = False

#Перевіряє, чи знаходиться точка point всередині габаритного куба (AABB — Axis-Aligned Bounding Box) об’єкта obj.
def is_point_in_aabb(point, obj):
    bbox = get_world_bbox_corners(obj)
#Отримується список координат всіх 8 кутів прямокутного габаритного об'єму (bounding box) об'єкта у світових координатах.
    min_corner = Vector((min(v.x for v in bbox), min(v.y for v in bbox), min(v.z for v in bbox)))
    max_corner = Vector((max(v.x for v in bbox), max(v.y for v in bbox), max(v.z for v in bbox)))
#Обчислюються мінімальні та максимальні координати по X, Y, Z — тобто кути "ящика" bounding box, в якому знаходиться об’єкт.
    return (
        min_corner.x <= point.x <= max_corner.x and
        min_corner.y <= point.y <= max_corner.y and
        min_corner.z <= point.z <= max_corner.z
    )
#Повертає True, якщо точка point лежить всередині bounding box об'єкта. Інакше — False.

def is_colliding(obj1, obj2, tx=0.3, ty=0.3, tz=0.3):
    #Перевіряє, чи знаходяться два об’єкти досить близько один до одного — умовна "колізія".
    return (abs(obj1.location.x - obj2.location.x) < tx and
            abs(obj1.location.y - obj2.location.y) < ty and
            abs(obj1.location.z - obj2.location.z) < tz)
#Якщо відстань між об’єктами по кожній осі менша за заданий поріг (tx, ty, tz) — вважається, що об’єкти "зіштовхнулися".

def get_world_bbox_corners(obj):
    return [obj.matrix_world @ Vector(corner) for corner in obj.bound_box]
#Отримує координати всіх 8 кутів bounding box у світових координатах.
```

```
def is_aabb_overlap(obj1, obj2):
    bbox1 = get_world_bbox_corners(obj1)
    bbox2 = get_world_bbox_corners(obj2)
#Перевіряє, чи перетинаються два габаритних прямокутника (bounding boxes)

    min1 = Vector((min(v.x for v in bbox1), min(v.y for v in bbox1), min(v.z for v in bbox1)))
    max1 = Vector((max(v.x for v in bbox1), max(v.y for v in bbox1), max(v.z for v in bbox1)))
 #Обчислення мінімального та максимального кута bounding box першого об’єкта.

    min2 = Vector((min(v.x for v in bbox2), min(v.y for v in bbox2), min(v.z for v in bbox2)))
    max2 = Vector((max(v.x for v in bbox2), max(v.y for v in bbox2), max(v.z for v in bbox2)))
#Обчислення мінімального та максимального кута bounding box другого об’єкта.

    return (
        min1.x <= max2.x and max1.x >= min2.x and
        min1.y <= max2.y and max1.y >= min2.y and
        min1.z <= max2.z and max1.z >= min2.z
    )



def ensure_properties():
    if not hasattr(bpy.types.Scene, "conveyor_speed"):
        bpy.types.Scene.conveyor_speed = bpy.props.FloatProperty(
            name="Швидкість",
            description="Швидкість руху конвеєра",
            default=0.1, min=0.01, max=5.0
        )
    if not hasattr(bpy.types.Scene, "conveyor_running"):
        bpy.types.Scene.conveyor_running = bpy.props.BoolProperty(
            name="Працює",
            description="Чи увімкнено режим керування",
            default=False
        )
…
class ConveyorBase:
    @staticmethod
    def get_objects():
        krishki = [
            obj for obj in bpy.data.objects
            if obj.name.startswith("krishka") and obj.parent is None
        ]
```

```
    rubber = bpy.data.materials.get("Rubber")
    mapping_node = None
    if rubber and rubber.use_nodes:
        for node in rubber.node_tree.nodes:
            if node.type == 'MAPPING':
                mapping_node = node
                break
    return krishki, mapping_node

@staticmethod
def move(distance, direction):
    krishki, mapping_node = ConveyorBase.get_objects()
    grabbers = [obj for obj in bpy.data.objects if obj.name.startswith("grabber_collider")]
    ends = [obj for obj in bpy.data.objects if obj.name.startswith("end")]

    wm = bpy.context.window_manager
    if "obstacle_shown" not in wm:
        wm["obstacle_shown"] = False
    if "last_blocked_direction" not in wm:
        wm["last_blocked_direction"] = None

    for krishka in krishki:
        probe = krishka.location.copy()
        probe.y -= 0.6 if direction == 'forward' else -0.6

        #Вивід повідомлення якщо перешкода та обмеження, та комбінації кнопок щоб вийти з перешкоди
        for obj in grabbers + ends:
            if is_point_in_aabb(probe, obj):
                if not wm["obstacle_shown"] or wm["last_blocked_direction"] != direction:
                    wm["obstacle_shown"] = True
                    wm["last_blocked_direction"] = direction
                    global warning_text

                    if obj.name.startswith("grabber_collider"):
                        warning_text = (
                            "Попереду маніпулятор!\n"
                            "Якщо рух вперед — натисніть □□□;\n"
                            "Якщо рух назад — натисніть □□□"
                        )

                    elif obj.name.startswith("end"):
                        warning_text = (
```

```
                    "Обмеження по руху!\n"
                    "Якщо рух вперед — натисніть □□□;\n"
                    "Якщо рух назад — натисніть □□□"
                )
            else:
                warning_text = "Перешкода!"

            bpy.ops.wm.call_menu(name="WM_MT_obstacle_warning")

        # Блокуємо рух тільки якщо знову у ту сторону
        if wm["last_blocked_direction"] == direction:
            return True  # стоп тільки в цю сторону

    # Рухаємо банку
    krishka.location.y -= distance * KRISHKA_SPEED_MULTIPLIER

# Рухаємо текстуру
if mapping_node:
    offset = mapping_node.inputs[1].default_value
    offset[1] += distance * TEXTURE_SPEED_MULTIPLIER
    mapping_node.inputs[1].default_value = offset

wm["obstacle_shown"] = False
wm["last_blocked_direction"] = None

return False
```

…

ДОДАТОК В

Код програми liquid_filler.py

```
…
import bpy

# Налаштування
MAX_FILL_HEIGHT = 2.5
# Межі руху
NOZZLE_MIN_Z = -4.5
NOZZLE_MAX_Z = -2.45
NOZZLE_SUPPORT_MIN_X = -1.75
NOZZLE_SUPPORT_MAX_X = 2.5

def find_attached_honey_cutter():
    nozzle_support = bpy.data.objects.get("nozzle_support")
    if not nozzle_support:
        return None

    for krishka in bpy.data.objects:
        if not krishka.name.startswith("krishka"):
            continue

        dx = abs(nozzle_support.location.x - krishka.location.x)
        dy = abs(nozzle_support.location.y - krishka.location.y)
        dz = nozzle_support.location.z - krishka.location.z

        if dx < 0.14 and dy < 0.14 and 0.35 < dz < 0.6:
            # Є банка під соплом - шукаємо її дитини з honey_cutter
            for child in krishka.children:
                if child.name.startswith("honey_cutter"):
                    return child
    return None

def ensure_properties():
    props = bpy.types.Scene

    if not hasattr(props, "manipulator_honey_speed"):
        props.manipulator_honey_speed = bpy.props.FloatProperty(
            name="Швидкість Маніпулятора",
            description="Швидкість руху сопла і балки",
            default=0.05,
```

```
            min=0.001,
            max=1.0
        )

    if not hasattr(props, "keyboard_control_running"):
        props.keyboard_control_running = bpy.props.BoolProperty(
            name="Керування Клавіатурою Активне",
            description="Чи працює режим керування клавішами",
            default=False
        )

    if not hasattr(props, "fill_speed"):
        props.fill_speed = bpy.props.FloatProperty(
            name="Швидкість Наливання Рідини",
            description="Регулювання швидкості заливки Рідини",
            default=0.5,
            min=0.1,
            max=1.0
        )

    if not hasattr(props, "cutter_obj_name"):
        props.cutter_obj_name = bpy.props.StringProperty(
            name="Ім’я Об’єкта Рідини",
            description="Технічне ім’я об’єкта honey_cutter.xxx",
            default=""
        )
    if not hasattr(props, "fill_progress"):
        props.fill_progress = bpy.props.IntProperty(
            name="Прогрес Заливки",
            description="Відсоток налитого рідини",
            default=0,
            min=0,
            max=100
        )

def is_krishka_under_nozzle_support():
    nozzle_support = bpy.data.objects.get("nozzle_support") # Отримуємо об'єкт сопла
    if not nozzle_support:
        return False  # Якщо об'єкт не знайдено — повертаємо помилкове значення

    # Перебираємо всі об'єкти, що починаються з "krishka"
    for obj in bpy.data.objects:
```

```
        if obj.name.startswith("krishka"):

            # Обчислюємо відстань між об'єктом та соплом по кожній осі
            dx = abs(nozzle_support.location.x - obj.location.x)
            dy = abs(nozzle_support.location.y - obj.location.y)
            dz = nozzle_support.location.z - obj.location.z

            # Якщо відстані менші за допустимі межі — вважається, що банка під соплом
            if dx < 0.14 and dy < 0.14 and 0.35 < dz < 0.6:
                return True
    return False # Якщо жодна банка не під соплом — повертаємо False

def get_krishka_under_nozzle():
    nozzle_support = bpy.data.objects.get("nozzle_support")
    if not nozzle_support:
        return None

    for obj in bpy.data.objects:
        if obj.name.startswith("krishka"):
            dx = abs(nozzle_support.location.x - obj.location.x)
            dy = abs(nozzle_support.location.y - obj.location.y)
            dz = nozzle_support.location.z - obj.location.z
            if dx < 0.14 and dy < 0.14 and 0.35 < dz < 0.6:
                return obj
    return None
…
# Оператор заливки рідини
class WM_OT_FillHoney(bpy.types.Operator):
    bl_idname = "wm.fill_honey"
    bl_label = "Залити Рідини"

    _timer = None
    _cutter = None
    _stream = None
    _stream_start_z = 0
    _stream_returning = False

    def execute(self, context):
        if not is_krishka_under_nozzle_support():
            self.report({'ERROR'}, "Немає банки під соплом!")
            return {'CANCELLED'}
```

```
        cutter = find_attached_honey_cutter()
        if not cutter:
            self.report({'ERROR'}, "Об'єкт 'honey_cutter' не знайдено!")
            return {'CANCELLED'}
        if cutter.location.z > 2.0:
            self.report({'WARNING'}, "У цій банці вже є рідина")
            return {'CANCELLED'}

        self._cutter = cutter  # зберігаємо об’єкт
        stream = bpy.data.objects.get("honey_stream")
        if stream:
            self._stream = stream
            self._stream_start_z = stream.location.z
        context.scene.fill_progress = 1
        self._timer = context.window_manager.event_timer_add(0.02, window=context.window)
        context.window_manager.modal_handler_add(self)
        return {'RUNNING_MODAL'}

    def modal(self, context, event):
        if event.type == 'TIMER':
            if self._cutter and self._cutter.location.z < MAX_FILL_HEIGHT:
                # Двигаємо потік вниз
                if self._stream:
                    target_z = self._stream_start_z - 4
                    if self._stream.location.z > target_z:
                        self._stream.location.z -= context.scene.fill_speed / 5
                # Поднимаємо рідину
                self._cutter.location.z += context.scene.fill_speed / 10
                # Оновлюємо прогресс
                percent = (self._cutter.location.z / MAX_FILL_HEIGHT) * 100
                context.scene.fill_progress = int(min(percent, 100))
            else:
                # Завершення: збросити потік та прогресс
                if self._stream:
                    self._stream.location.z = self._stream_start_z
                context.window_manager.event_timer_remove(self._timer)
                context.scene.fill_progress = 0
                return {'FINISHED'}

            return {'RUNNING_MODAL'}
        return {'PASS_THROUGH'}
…
```

ДОДАТОК Г

Код програми manipulator.py

```
…
import bpy
import math
import mathutils
from mathutils import Vector

# Налаштування
CATCH_DISTANCE = 3.0
PLACE_DISTANCE = 4.0
BOX_PLACE_DISTANCE = 5.0
ALIGN_OFFSET_Z = 2.20
MIN_DIS = 0.25
MIN_X, MAX_X = -6.5, 13.5
MIN_Y, MAX_Y = -25, -2.2
MIN_Z, MAX_Z = 0.6, 9

def is_colliding_with_obstacles(obj):
    bbox1 = [obj.matrix_world @ Vector(corner) for corner in obj.bound_box]
    min1 = Vector((min(v.x for v in bbox1), min(v.y for v in bbox1), min(v.z for v in bbox1)))
    max1 = Vector((max(v.x for v in bbox1), max(v.y for v in bbox1), max(v.z for v in bbox1)))

    for ob in bpy.data.objects:
        name = ob.name.lower()
        if (name.startswith("obstacle") or name.startswith("krishka")) and ob.type == 'MESH':
            bbox2 = [ob.matrix_world @ Vector(corner) for corner in ob.bound_box]
            min2 = Vector((min(v.x for v in bbox2), min(v.y for v in bbox2), min(v.z for v in bbox2)))
            max2 = Vector((max(v.x for v in bbox2), max(v.y for v in bbox2), max(v.z for v in bbox2)))

            if (min1.x <= max2.x and max1.x >= min2.x and
                min1.y <= max2.y and max1.y >= min2.y and
                min1.z <= max2.z and max1.z >= min2.z):
                return True

    return False
…
def find_nearest_free_box():
    controller = bpy.context.scene.manipulator_controller
    nearest_box = None
    min_dist = float('inf')
```

```
    if not controller:
        return None
    for obj in bpy.data.objects:
        if obj.name.startswith("box"):
            has_banka = any(child.name.lower().startswith("krishka") for child in obj.children)
            if not has_banka:
                dist = (obj.location - controller.location).length
                if dist < min_dist:
                    min_dist = dist
                    nearest_box = obj
    return nearest_box

def align_to_object(controller, target, offset_z):
    controller.location.x = target.location.x
    controller.location.y = target.location.y - 0.10
    controller.location.z = target.location.z + offset_z

def get_attached_banka():
    ctrl = bpy.context.scene.manipulator_controller
    if not ctrl:
        return None
    for obj in bpy.data.objects:
        name = obj.name.lower()
        if name.startswith("banka") and obj.parent == ctrl:
            return obj
    return None

def get_attached_krishka():
    ctrl = bpy.context.scene.manipulator_controller
    if not ctrl:
        return None
    for obj in bpy.data.objects:
        name = obj.name.lower()
        if name.startswith("krishka") and obj.parent == ctrl:
            return obj
    return None

def align_to_free_box_point(ctrl, offset_z):
    scene = bpy.context.scene
    for point in bpy.data.objects:
        if point.name.startswith("box"):
            has_banka = any(child.name.lower().startswith("krishka") for child in point.children)
```

```
            if not has_banka:
                align_to_object(ctrl, point, offset_z)  # контролер вирівнюється к обєкту
                scene.aligned_to_box = True
                return True
    return False

def align_to_object(controller, target, offset_z):
    controller.location.x = target.location.x
    controller.location.y = target.location.y - 0.10
    controller.location.z = target.location.z + offset_z
…
# кришка в руці — підводимо до банки
        elif attached_kryshka:
            if krishka:
                dist = (krishka.location - ctrl.location).length
                if dist < PLACE_DISTANCE:
                    layout.operator("wm.align_to_jar", text="Вирівняти по Банці")

                    dist = (krishka.location - ctrl_offset).length
                    if dist < MIN_DIS:
                        # перевірка діапазану між катерром і банкою
                        has_honey = False
                        if krishka:
                            base_z = krishka.matrix_world.translation.z
                            for child in krishka.children:
                                if child.name.lower().startswith("honey_cutter"):
                                    z_world = child.matrix_world.translation.z
                                    delta_z = z_world - base_z
                                    if 0.9 <= delta_z <= 1.1:
                                        has_honey = True
                                        break
                        if has_honey:
                            layout.operator("wm.rotate_banka", text="Закрутити Кришку")
                        else:
                            layout.label(text=" У цієї банки нема рідина")
                    else:
                        layout.label(text="Підведіть граббер ближче до Банки!", icon='INFO')
                else:
                    layout.label(text="Банка не знайдена! Додайте нову.")

        # кришка ще не захвачена
        else:
```

```
        kryshka = find_nearest_object("banka")
        if kryshka:
            dist = (kryshka.location - ctrl_offset).length
            if dist < CATCH_DISTANCE:
                layout.operator("wm.align_to_banka", text="Вирівняти до Кришки")

                dist = (kryshka.location - ctrl_offset).length
                if dist < MIN_DIS:
                    layout.operator("wm.grab_banka", text="Захопити Кришку")
            else:
                layout.label(text=" Підведіть граббер ближче до Кришки!", icon='INFO')
        else:
            layout.label(text="Кришка не знайдена!")
…
# функція перевірки столкнення
    def is_inside_obstacle(obj):
        bbox1 = [obj.matrix_world @ Vector(corner) for corner in obj.bound_box]
        min1 = Vector((min(v.x for v in bbox1), min(v.y for v in bbox1), min(v.z for v in bbox1)))
        max1 = Vector((max(v.x for v in bbox1), max(v.y for v in bbox1), max(v.z for v in bbox1)))

        for ob in bpy.data.objects:
            name = ob.name.lower()
            if (name.startswith("obstacle") or name.startswith("krishka")) and ob.type == 'MESH':

                if name.startswith("krishka") and ob.parent == ctrl:
                    continue

                bbox2 = [ob.matrix_world @ Vector(corner) for corner in ob.bound_box]
                min2 = Vector((min(v.x for v in bbox2), min(v.y for v in bbox2), min(v.z for v in bbox2)))
                max2 = Vector((max(v.x for v in bbox2), max(v.y for v in bbox2), max(v.z for v in bbox2)))

                if (min1.x <= max2.x and max1.x >= min2.x and
                    min1.y <= max2.y and max1.y >= min2.y and
                    min1.z <= max2.z and max1.z >= min2.z):
                    return True

        return False

    #Если застрягли — дозвалити тільки по осі Z уверх
    if is_inside_obstacle(collider):
        if axis != 'Z' or direction <= 0:
            bpy.ops.wm.call_menu(name="WM_MT_manipulator_obstacle")
```

```
            return {'CANCELLED'}

    #Стандартний рух з обмеженням
    if axis == 'X':
        ctrl.location.x = min(max(ctrl.location.x + direction * speed, MIN_X), MAX_X)
    elif axis == 'Y':
        ctrl.location.y = min(max(ctrl.location.y + direction * speed, MIN_Y), MAX_Y)
    elif axis == 'Z':
        ctrl.location.z = min(max(ctrl.location.z + direction * speed, MIN_Z), MAX_Z)

    return {'FINISHED'}
…
```

ДОДАТОК Д

Код програми forklift.py

```
…
import bpy
import math
import time
from mathutils import Vector

POSITION_TOLERANCE = 0.20
FORKLIFT_NAME = "Forklift"
KRISHKA_PREFIX = "krishka"
KRISHKA_TOTAL = 30

def is_inside_box(obj, box_obj):
    bbox_corners = [box_obj.matrix_world @ Vector(corner) for corner in box_obj.bound_box]
    min_corner = Vector((min(v[0] for v in bbox_corners),
                         min(v[1] for v in bbox_corners),
                         min(v[2] for v in bbox_corners)))
    max_corner = Vector((max(v[0] for v in bbox_corners),
                         max(v[1] for v in bbox_corners),
                         max(v[2] for v in bbox_corners)))
    obj_loc = obj.matrix_world.translation
    return all(min_corner[i] <= obj_loc[i] <= max_corner[i] for i in range(3))

def round_xy(vec, precision=0.1):
    return (round(vec[0] / precision) * precision, round(vec[1] / precision) * precision)

def fill_stacks(box_name="boox", cap_prefix="banka", z_offset=0.19):
    box = bpy.data.objects.get(box_name)
    if not box:
        print(f"[!] Box '{box_name}' не знайдено")
        return

    cap_template = None
    for obj in bpy.data.objects:
        if obj.name.startswith(cap_prefix):
            cap_template = obj
            break
    if not cap_template:
        print(f"[!] Не знайдено шаблон кришки з префіксом '{cap_prefix}'")
        return
```

```
    # отримуємо колекцію
    collection_name = "Krishki"
    if collection_name in bpy.data.collections:
        target_col = bpy.data.collections[collection_name]
    else:
        target_col = bpy.data.collections.new(collection_name)
        bpy.context.scene.collection.children.link(target_col)

    stacks = {}
    for obj in bpy.data.objects:
        if obj.name.startswith(cap_prefix) and is_inside_box(obj, box):
            xy = round_xy(obj.location)
            stacks.setdefault(xy, []).append(obj)

    for xy, caps in stacks.items():
        caps.sort(key=lambda c: c.location.z)
        count = len(caps)
        missing = 10 - count
        if missing <= 0:
            continue

        top_z = caps[-1].location.z if caps else 0

        for i in range(missing):
            new_cap = cap_template.copy()
            new_cap.data = cap_template.data.copy()
            new_cap.location = Vector((xy[0], xy[1], top_z + z_offset * (i + 1)))
            target_col.objects.link(new_cap)
…
def delete_nested_objects():
    # шукаємо усі box.xxx
    boxes = [obj for obj in bpy.data.objects if obj.name.startswith("box")]

    for box in boxes:
        for krishka in box.children:
            if krishka.name.startswith("krishka"):
                # видаляємо дітей krishka, якщо вони banka або honey
                for child in list(krishka.children):
                    if child.name.startswith("banka") or child.name.startswith("honey"):
                        bpy.data.objects.remove(child, do_unlink=True)
                # видаляємо krishka
                bpy.data.objects.remove(krishka, do_unlink=True)

class OBJECT_OT_process_crates(bpy.types.Operator):
    bl_idname = "object.process_crates"
    bl_label = "Принести нові ящики"
    bl_options = {'REGISTER', 'UNDO'}
```

```
    _timer = None
    _start_location = None
    _step = 0
    _direction = 1
    _start_time = 0
    _deleted = False

    def modal(self, context, event):
        if event.type != 'TIMER':
            return {'PASS_THROUGH'}

        obj = bpy.data.objects.get(FORKLIFT_NAME)
        if not obj:
            self.report({'ERROR'}, f"Объект {FORKLIFT_NAME} не знайдено")
            return {'CANCELLED'}

        # Фаза 1: від’їзд назад
        if self._phase == "depart":
            local_step = obj.matrix_world.to_3x3() @ Vector((0, -0.15, 0))
            obj.location += local_step
            self._step += 1
            if self._step >= 70:
                self._phase = "cleanup"

        # Фаза 2: видалення об’єктів
        elif self._phase == "cleanup":
            delete_nested_objects()
            self._phase = "return"

        # Фаза 3: повернення назад
        elif self._phase == "return":
            local_step = obj.matrix_world.to_3x3() @ Vector((0, 0.15, 0))
            obj.location += local_step
            self._step_return += 1
            if self._step_return >= 70:
                context.window_manager.event_timer_remove(self._timer)
                return {'FINISHED'}

        return {'RUNNING_MODAL'}


    def execute(self, context):
        obj = bpy.data.objects.get(FORKLIFT_NAME)
        if not obj:
            self.report({'ERROR'}, f"Объект '{FORKLIFT_NAME}' не знайдено")
            return {'CANCELLED'}

        self._start_location = obj.location.copy()
        self._step = 0
        self._step_return = 0
        self._phase = "depart"
        self._deleted = False

        wm = context.window_manager
        self._timer = wm.event_timer_add(0.02, window=context.window)
        wm.modal_handler_add(self)
        return {'RUNNING_MODAL'}
…
```

ДОДАТОК Е

Код програми automation_controller.py

```
…
import bpy
import time
from bpy.props import IntProperty, BoolProperty
from mathutils import Vector
from  .liquid_filler import find_attached_honey_cutter


def move_object_straight(obj, target_location, speed=0.05):
    if obj is None:
        return True

    direction = target_location - obj.location
    distance = direction.length

    if distance < speed:
        obj.location = target_location.copy()
        return True
    else:
        direction.normalize()
        obj.location += direction * speed
        return False



def can_fill_honey():
    cutter = find_attached_honey_cutter()
    return cutter and cutter.location.z < 2.0


def is_close(a, b, max_distance=0.25):
    return (a - b).length <= max_distance


def has_honey(jar):
    for child in jar.children:
        if child.name.lower().startswith("honey_cutter"):
            jar_z = jar.matrix_world.translation.z
            cutter_z = child.matrix_world.translation.z
            if 0.9 <= (cutter_z - jar_z) <= 1.1:
                return True

    return False


def get_view3d_override():
    for area in bpy.context.screen.areas:
        if area.type == 'VIEW_3D':
            for region in area.regions:
                if region.type == 'WINDOW':
                    return {
                        'window': bpy.context.window,
                        'screen': bpy.context.screen,
                        'area': area,
                        'region': region,
                        'scene': bpy.context.scene
                    }
```

```
    return bpy.context


class AutoOperator(bpy.types.Operator):
    bl_idname = "wm.run_auto_operator"
    bl_label = "Автоматичний Процес"

    _timer = None
    _stage = 0
    _jar_counter = 0
    _jar_total = 0
    _move_count = 0
    _target = None
    _controller = None
    _delay = 0
    _rotated = False
    _stopped_by_user = False

    def modal(self, context, event):
        if context.scene.auto_line_force_cancel:
            self._stopped_by_user = True
            context.scene.auto_line_force_cancel = False
            return self.cancel(context)

        if not context.scene.auto_line_running:
            return self.cancel(context)

        if event.type != 'TIMER':
            return {'PASS_THROUGH'}

        if self._delay > 0:
            self._delay -= 1
            return {'RUNNING_MODAL'}

        try:
            if self._stage == 0:
                if not can_fill_honey():
                    bpy.ops.wm.move_conveyor_forward()
                else:
                    self._stage = 1

            elif self._stage == 1:
                bpy.ops.wm.fill_honey()
                self._stage = 2

            elif self._stage == 2:
                if context.scene.fill_progress == 0:
                    self._move_count = 0
                    self._stage = 3

            elif self._stage == 3:
                if self._move_count < 100:
                    bpy.ops.wm.move_conveyor_forward()
                    self._move_count += 1

                    self._controller = context.scene.manipulator_controller
                    self._target = self._get_target("banka")

                    self._target = bpy.data.objects.get("cap_align_point")
                    if self._controller and self._target:
                        aligned = move_object_straight(self._controller, self._target.location, 0.4)
                        if aligned:
                            bpy.ops.wm.align_to_banka()
```

```
                    self._delay = 5
                    self._stage = 4

        elif self._stage == 4:
            bpy.ops.wm.grab_banka()
            self._target = self._get_target("krishka")
            self._rotated = False
            self._stage = 5


        elif self._stage == 5:
            if self._controller and self._target:
                aligned = move_object_straight(self._controller, self._target.location + Vector((0, -0.1, 2.2)), 0.4)
                if aligned and not self._rotated:
                    bpy.context.view_layer.update()
                    if is_close(self._controller.location, self._target.location + Vector((0, -0.1, 2.2))) and
has_honey(self._target):
                        context.scene.auto_rotation_running = True
                        bpy.ops.wm.rotate_banka()
                        self._rotated = True
                        self._delay = 5
                        self._stage = 6

        elif self._stage == 6:
            self._delay = 140
            self._target = self._get_target("box")
            self._expected_rotation = None
            self._stage = 7


        elif self._stage == 7:
            self._target = bpy.data.objects.get("align_box_point")
            if self._controller and self._target:
                aligned = move_object_straight(self._controller, self._target.location, 0.4)
                if aligned:
                    bpy.ops.wm.align_to_box()
                    self._delay = 5
                    self._stage = 8

        elif self._stage == 8:
            try:
                bpy.ops.wm.place_jar_in_box()
                self._jar_counter += 1
                context.scene.auto_line_jar_done += 1

                if self._jar_counter % 6 == 0:
                    bpy.ops.object.fill_stacks_from_base()

                self._returning_home = True
                self._stage = 9

            except Exception as e:
                self.report({'ERROR'}, f"Помилка при розміщенні банки: {e}")
                return self.cancel(context)


        elif self._stage == 9:
            boxes = [obj for obj in bpy.data.objects if obj.name.startswith("box")]
            occupied = 0
            for box in boxes:
                for child in box.children:
                    if child.name.startswith("krishka"):
                        occupied += 1
```

```
                    break

            if occupied >= len(boxes):
                bpy.ops.object.process_crates()

            if self._jar_counter >= self._jar_total:
                self._returning_home = True
                self._stage = 10

            else:
                self._stage = 0  #початок нового цикла

        elif self._stage == 10:
            if not self._returning_home:
                def delayed_fill():
                    try:
                        bpy.ops.object.fill_stacks_from_base()
                    except Exception as e:
                        print("Помилка при виклику fill_stacks_from_base:", e)
                    return None

                bpy.app.timers.register(delayed_fill, first_interval=0.1)

                self.report({'INFO'}, f"Автомат завершив роботу: {self._jar_total} банок виконано.")
                if not self._stopped_by_user:
                    context.scene.auto_line_last_batch = self._jar_total
                    context.scene.auto_line_total_jars_global = 0
                    context.scene.auto_line_jar_done = 0

                return self.cancel(context)


    except Exception as e:
        self.report({'ERROR'}, f"Автомат зупинено через помилку: {e}")
        return self.cancel(context)

    if self._returning_home:
            done = move_object_straight(self._controller, self._start_position, 0.4)
            if done:
                self._returning_home = False

    context.scene.auto_line_stage_last = self._stage

    return {'RUNNING_MODAL'}

def _get_target(self, prefix):
    nearest = None
    min_dist = float('inf')
    for obj in bpy.data.objects:
        if obj.name.startswith(prefix):
            dist = (obj.location - self._controller.location).length
            if dist < min_dist:
                nearest = obj
                min_dist = dist
    return nearest

def execute(self, context):
    if context.scene.auto_line_running:
        self.report({'WARNING'}, "Автомат вже працює")
        return {'CANCELLED'}

    context.scene.auto_line_total_jars_global = context.scene.auto_line_total_jars
```

```
        self._jar_total = context.scene.auto_line_total_jars_global
        self._jar_counter = context.scene.auto_line_jar_done

        return_point = bpy.data.objects.get("return_point")
        if return_point:
            self._start_position = return_point.location.copy()
        else:
            self.report({'WARNING'}, " Об'єкт return_point не знайдено. Маніпулятор повернеться на стартову позицію.")
            self._start_position = context.scene.manipulator_controller.location.copy()

        self._stage = 0
        self._delay = 0
        self._rotated = False
        context.scene.auto_rotation_running = False
        self._returning_home = False
        context.scene.auto_line_running = True
        wm = context.window_manager
        self._timer = wm.event_timer_add(0.05, window=context.window)
        wm.modal_handler_add(self)
        return {'RUNNING_MODAL'}

    def cancel(self, context):
        wm = context.window_manager
        if self._timer:
            wm.event_timer_remove(self._timer)
        context.scene.auto_line_running = False
        if self._stopped_by_user:
    # якщо зупинили мануально та прошли заливку (stage >= 3), то зарахуємо банку
            if self._stage >= 3:
                context.scene.auto_line_manual_stop_after_fill = True
                self.report({'INFO'}, " Зупинено після заливки — очікується ручне завершення.")
            else:
                self.report({'INFO'}, " Автоматизацію зупинено користувачем.")

        for area in bpy.context.screen.areas:
            if area.type == 'VIEW_3D':
                area.tag_redraw()

        return {'CANCELLED'}
…
```

ДОДАТОК Є

Демонстраційний матеріал

Формат	Позиція.	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кільк.	Примітки
			Текстові документи		
А4	1	ГЮІК ХХХХХХ.043 ПЗ	Пояснювальна записка		А4 с. 92
			Інші документи		
	2		Демонстраційний матеріал		А4. 14 с.

					ГЮИК ХХХХХХ.043			
Змін.	Арк	Номер докум.	Підпис	Дата				
Розроб.		Даниленко			Розроблення додатку–тренажера маніпулятора для захоплення об'єктів на конвеєрі	Літера	Арку	Арку
Перевір.		Близнюк				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Демська						
Затв.		Невлюдов						