Industry Science and Engineering Vol. 1 No. 11, 2024

Development of the Digital Twin System for Four-Axis Suction and Circular Conveyor Transport

Min Song[1], Tiejun Pan[1,*], Huijie Huang[1], Junyi Chai[1], Enhui Hu[1], Samuel Ken-En Gan[2], Leina Zheng[3]

[1]*College of Information Engineering, College of Science & Technology Ningbo University, Ningbo, Zhejiang, China*

[2]*Ouhai College of Science, Wenzhou-Kean University Wenzhou, Zhejiang, China*

[3]*Yinzhou business school, Zhejiang Wanli University, Ningbo, Zhejiang, China*

**Corresponding Author.*

Abstract: With the rapid development of 5G technology, digital twins and Industry 5.0 have become hot focal points in the manufacturing sector. A digital twin is a technology that fully utilizes models, data, and intelligence, integrating multidisciplinary knowledge. It serves the entire product lifecycle and acts as a bridge between the physical and information worlds, providing more real-time, efficient, and intelligent services. This study focuses on the implementation pathways and construction framework of digital twin-controlled robotic arms. Relying on an intelligent factory assembly line experimental device, we have built an experimental system for digital twin robotic arms in intelligent assembly, which includes the physical robotic arm device and its virtual counterpart. The study also elaborates on the significant role of digital twin robotic arms in advancing the manufacturing industry. Through the digital twin platform, we can achieve visual management of the entire production line operation, thereby gaining deeper insights into the actual conditions of the production line and quickly identifying and resolving issues.

Keywords: MQTT; Robotic Arm Control; Digital Twin; Virtual Simulation

1. Introduction

Digital Twin technology can be characterized by three aspects: full lifecycle, real-time, and bidirectional [1]. The full lifecycle aspect emphasizes that Digital Twins can span the entire lifecycle of a product, from design, development, and manufacturing to service, maintenance, and end-of-life recycling. It helps developers design products and also enhances users' experience with the products. Real-time/near real-time refers to the ability to establish a comprehensive real-time or near real-time connection between the physical entity and its virtual counterpart. The two are not completely independent entities, and the mapping relationship between them also has a certain degree of real-time nature. Bidirectional means that the data flow between the physical entity and the virtual twin is two-way. It is not limited to the physical entity sending data to the virtual twin; the virtual twin can also provide feedback to the physical entity. Developers can take further actions and interventions on the physical entity based on the feedback from the virtual twin, ultimately achieving control over the virtual robotic arm.

The structure of robotic arms is relatively simple and lacks effective management methods. Although robotic arms are widely used, their control modes and remote monitoring capabilities have significant limitations. Specifically, the control mode of robotic arms is usually quite limited, only allowing for the execution of simple, repetitive motions according to preset programs [2]. This makes it difficult for them to adapt flexibly to complex and changing environments and task requirements, which greatly restricts their potential for application in broader fields and their work efficiency. At the same time, traditional robotic arms also have obvious shortcomings in remote monitoring and fault diagnosis. Due to the lack of efficient remote communication and virtual management methods, it is difficult to achieve real-time and accurate remote monitoring and precise fault diagnosis, thereby affecting the maintenance

efficiency and reliability of robotic arms.

In summary, our contributions are as follows:

To achieve precise target detection and positioning, we introduced the Cyclic Coordinate Descent Inverse Kinematics (CCDIK) technology. By accurately solving the kinematic equations of the robotic arm, this technology can significantly optimize its motion trajectory, ensuring precise positioning and efficient grasping operations, thereby greatly enhancing the working accuracy of the robotic arm. CCDIK also possesses excellent environmental adaptability, allowing it to dynamically adjust control parameters based on real-time perception of the working environment and task requirements, demonstrating remarkable flexibility.

In terms of data transmission and communication, we selected the MQTT protocol. As a lightweight, publish/subscribe-based messaging protocol, MQTT is highly suitable for use in low-bandwidth, unstable network environments. By simulating the motion trajectory and working status of the robotic arm, the MQTT protocol provides stable, low-latency communication services, offering opportunities for validation and optimization of the design.

Furthermore, we combined the advantages of a four-axis pick-and-place robotic arm and a six-axis ABB robotic arm to expand logistics transportation capabilities. The four-axis pick-and-place robotic arm has precise placing and picking capabilities, enabling rapid and accurate material handling and transportation. The six-axis ABB robotic arm, on the other hand, demonstrates exceptional flexibility and assembly processing capabilities, allowing it to perform more complex assembly tasks. This combination not only promotes the application expansion of robotic arms in fields such as healthcare and logistics but also enhances the accuracy and safety of logistics transportation. By integrating these advanced technologies, we have laid a solid foundation for the intelligent and efficient application of robotic arms.

2. Related Works

2.1 Virtual Simulation

Unity3D is a professional game development engine that can be used for game development, animation production, engineering simulation, VR design, and more. For example, Unity3D has been employed to develop an industrial robot simulation and teaching experimental system for educational purposes. Additionally, a hybrid virtual and physical industrial robot experimental platform has been designed using Unity3D. Research has also been conducted on the virtual and physical twin control simulation of industrial robots using Unity3D. Furthermore, a virtual teaching system for robot disassembly and assembly has been designed with Unity3D to facilitate learning in this area. These examples demonstrate the feasibility of developing a virtual simulation training and teaching system using Unity3D.

In another application, the UI components of the Unity3D engine have been utilized to develop a human-computer interaction interface. The main interface includes four functions: communication connection, joint control, command control, and trajectory control. In the system monitoring experiment, the industrial robot is activated through a set program to observe whether the movements of the virtual and physical robots are synchronized. In the control experiment, commands are generated to test the control of joints, end-point positions, and end trajectories.

2.2 Data Interaction and Communication

In data communication, there exists a multitude of communication protocols that stipulate the methods of data transmission, formats, synchronization mechanisms, error detection, and correction. For example, Wen Guojun et al. adopted a Socket network communication model based on the TCP/IP protocol. In the Unity platform, a C# script is used as the Socket client, while the physical robot controller serves as the server. Once the physical and virtual models are ready for communication, a connection is established based on the server's IP address and port number. After a successful handshake, a sub-thread is created to transmit and receive data. Commands are sent in the format of Turing robot network command packets to achieve bidirectional transmission. The system also communicates with the robot's PLC to control the robot's status by setting IO signal switches. The external PLC used for the industrial robot is the Mitsubishi FX5U series. During development, the

HslCommunication open-source library is employed, and its API functions are used to efficiently read and write IO data from the PLC.

3. Proposed Method

To ensure the standardization and orderliness of the system development process, thereby guaranteeing the stable operation of the system and its future maintainability, it is necessary to divide software development into modules based on guidelines. Developing the system by modules not only accelerates the development process but also clarifies the system requirements more effectively, thus ensuring the stability of the system. Under the editor module, there are mainly three major modules [3], and below the module level are more refined system functions, as shown in Figure 1.

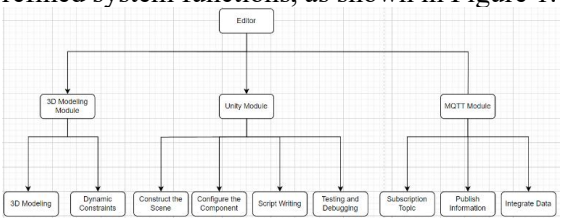

Figure 1. System Function Module Diagram

3.1 Construction of Virtual Models

In this project, we first created digital models of the existing mechanical equipment using 3ds Max software and then integrated them into the 3D simulation space provided by the Unity engine. Subsequently, on the Unity platform, we designed a UI interface and configured the model components in a parent-child hierarchy to achieve the desired motion constraints. By utilizing Unity's 3D editing tools, we directly dragged and dropped the models into the Scene view for layout. Through scene layout settings, we precisely positioned, rotated, and scaled these 3D models. By assigning materials and setting up lighting, we covered the models with material spheres to achieve rich visual characteristics and lighting effects [4]. The construction of the scene is shown in Figure 2.

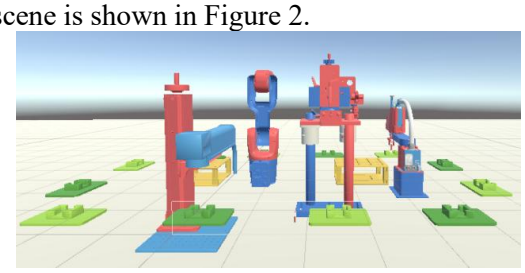

Figure 2. 3D Model Rendering

3.2 Dynamic Constraints of the Model

To ensure that the movement of objects in the virtual environment is both logical and coherent, at the software design level, we utilize parent-child relationship chains to bind the dynamic components of objects. This not only ensures that individual components do not become detached from the overall model during movement but also organizes the various parts of the object through the hierarchical structure of parent-child relationships. Specifically, we establish a hierarchical relationship in the software based on the operational process of the object, placing components affected by the active parts above the active parts in the hierarchy, thus forming a clear subordinate relationship. When we import such models into the Unity3D engine, we can visually see these parent-child relationships, and by simply calling the child object's name, we can easily manage and control the moving parts of the object and their affected areas. For example, if the base is the parent, then all parts of the mechanical arm connected to it by a hinge must be its child objects. If the parent moves, all its child objects will follow and move synchronously. Conversely, if a child object moves, the parent does not necessarily follow and move synchronously. Figure 3 shows the simulation of the master-slave movement relationship between the parts of the mechanical arm's hinges completed through detailed parent-child relationships.

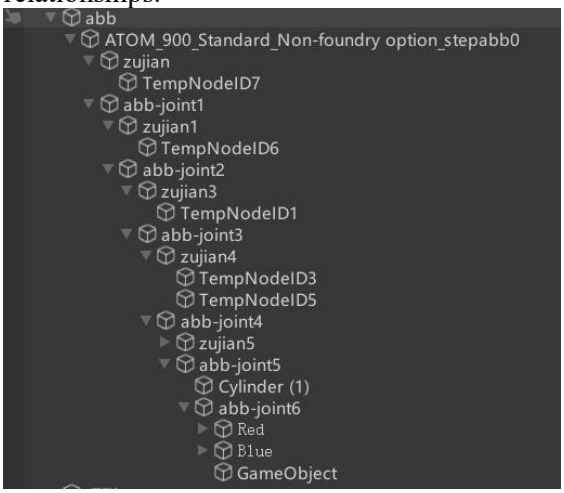

Figure 3. Parent-Child Configuration

3.3 Component Configuration

In the Unity module, users are allowed to design, simulate, and test the movements and

functions of the robotic arm. Developers can leverage Unity's physics engine and animation system to add various components to the robotic arm, such as joints, actuators, and sensors, thereby achieving precise motion control and interactive functions [5]. Unity also provides powerful integrated editor features that enable developers to easily manage game projects and assets. Through the Unity Editor, developers can create, import, and export various game objects and resources, while using the hierarchy and inspector panels to organize and edit them, as shown in Figure 4.

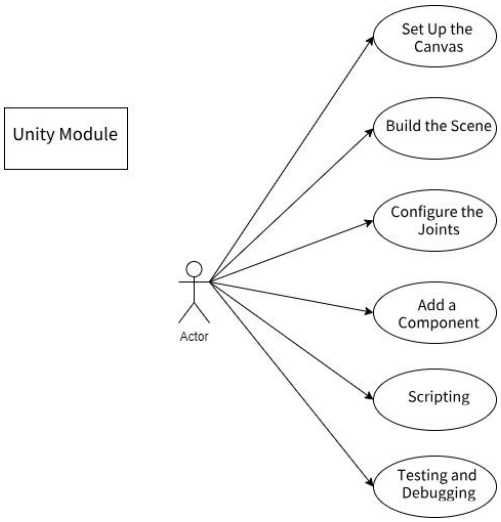

Figure 4. Use Case Diagram of the Unity Module

To ensure that the robotic arm model has flexible movement capabilities in Unity, it is necessary to design and configure multiple joints. In Unity's Inspector panel, we need to meticulously configure the properties of each joint. This includes determining the connection point of the joint, which is the specific location between the two objects connected by the joint; setting the rotation axis of the joint, which determines the direction around which the robotic arm segment can rotate; and configuring the joint's limits and spring properties, which will affect the range of motion and dynamic behavior of the robotic arm, as shown in Figure 5.

Among the added components, the Transform component defines the position, rotation, and scale of the game object in 3D space. The Animator component is the core animation controller, used to manage the animation state machine of the game object. It allows developers to play, blend, and transition between different animations based on game logic, thereby achieving rich control effects. In addition, there are the Rigidbody component, which controls physical behavior, and the Camera component, which is responsible for rendering the game view and capturing and displaying elements in the scene. To configure the Rotation Limit of a joint in Unity, first ensure that the joint component is correctly attached to the corresponding Rigidbody and adjust the property parameters to achieve the desired physical effects. By properly setting these parameters, complex robotic arm control can be realized. Specifically, for Hinge joints, you can find the Angular Limits section in the Inspector panel. Here, you can set the minimum (Min) and maximum (Max) angles of joint rotation to restrict the range of motion. You can also adjust parameters such as Bounce Min and Bounce Max to simulate the elastic behavior of the joint when it reaches its rotation limits. The joint configuration is shown in Figure 6.

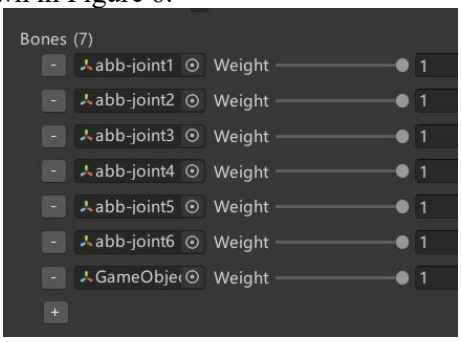

Figure 5. Joint Configuration Effect Diagram

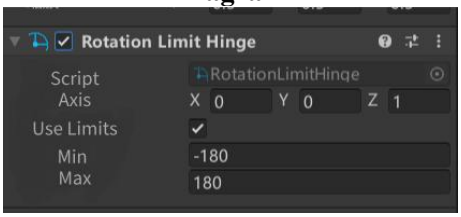

Figure 6. Component Configuration Effect Diagram

4. Experiments

4.1 Controlling Robotic Arm Movement with CCD-IK Plugin

In Unity, the combination of the CCD-IK (Cyclic Coordinate Descent Inverse Kinematics) plugin and Hinge Joint can achieve complex robotic arm functions. By calculating the movements of child bones and deriving the positions of parent bones in the bone chain, it determines the motion state of

the entire bone chain, enabling precise control of the end-effector's position. This is further enhanced by integrating path planning techniques to control the robotic arm's operation [6].

Firstly, in Unity, the typical approach is to create empty GameObjects as joints and connect them in a chain structure. For each joint of the robotic arm, a GameObject is created and equipped with appropriate physical components (Hinge Joint) to simulate joint movement. Using the CCD-IK plugin, the bone chain of the robotic arm is set up, and a target position is specified, allowing the end-effector to accurately reach the designated point.

To enable the robotic arm to move along a specific path, path planning techniques must be introduced. This can be based on inverse kinematics (IK) algorithms, which calculate the target angles each joint needs to achieve, ensuring the robotic arm moves smoothly and accurately along the preset path. Alternatively, other path planning algorithms can also be employed for the same purpose. Meanwhile, the Hinge Joint simulates the rotational behavior of the robotic arm's joints. By setting its Anchor, Axis, and Motor properties, the rotation axis, range, and power of the joint can be defined [7]. Combining these two elements creates a robotic arm that is both flexible and precise. CCD-IK is responsible for overall path planning and end-effector position control, while the Hinge Joint ensures that each joint's rotational movement conforms to physical laws. By writing scripts, the target position can be dynamically adjusted to achieve real-time control of the robotic arm. Additionally, collision detection and feedback mechanisms can be incorporated to further enhance the robotic arm's practicality and stability, enabling it to perform well in various application scenarios. The bone chain configurations are shown in Figures 7 and 8.

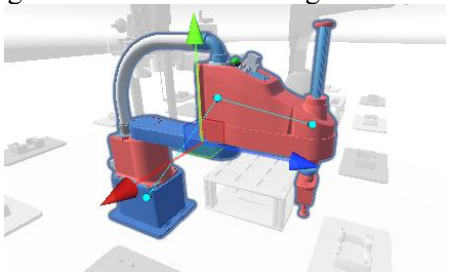

Figure 7. Bone Chain Diagram of a Four-Axis Robotic Arm

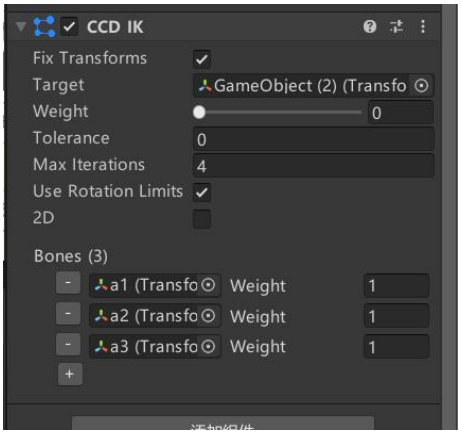

Figure 8. Configuration Diagram of the Four-Axis CCDIK Plugin

4.2 Writing Circular Transport Control Scripts

In Unity, to achieve smooth control of object positions and rotations on a circular conveyor belt using a robotic arm, scripts need to be written to read data and convert it into control commands. By iterating through the items list, the next object's position is calculated as the target position (targetPositions) for each object, and the current rotation of each object is saved as the target rotation (targetRotations). Here, the target positions and rotations are set based on the positions and rotations of the next object in the list, creating a looping movement effect.

First, the current positions and rotations of each items object (referred to as startPositions and startRotations) need to be recorded. Then, using an internal loop, linear interpolation (Vector3.Lerp and Quaternion. Slerp) is employed to smoothly update each object's position and rotation until they reach the target positions and rotations. Using Vector3.Lerp and Quaternion. Slerp for linear interpolation ensures that the objects' positions and rotations transition smoothly as they move from one point to another. In the script, the start and target positions and rotations are first defined, and then the current positions and rotations of the objects are calculated using interpolation functions based on a time parameter t. Binding the time t to an animation or button event allows dynamic adjustment of t, thereby controlling the movement and rotation of the objects. To achieve continuous transport, the objects' positions and rotations can be immediately reset to the start points once they

reach the target points, and the interpolation calculation can be restarted. Additionally, binding button controls to the robotic arm's operation allows users to start or stop the transport process by clicking buttons.

Through this approach, an intuitive and easily controllable robotic arm system can be created for smoothly transporting objects in a counterclockwise direction on a circular conveyor belt. The circular structure is shown in Figure 9.

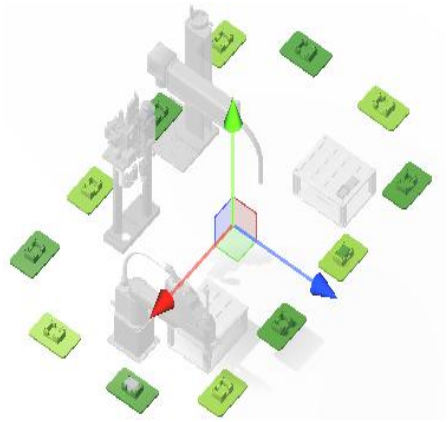

Figure 9. Circular Structure Diagram

4.3 Configuration of MQTT Communication and Testing and Adjustment

MQTT is a lightweight Internet of Things (IoT) communication protocol, whose core functions include message publishing and subscribing, device management, data storage, and analysis. Based on the publish/subscribe model, MQTT enables real-time communication and data exchange between devices, enhancing the flexibility and scalability of the system. It also supports comprehensive device management, ensuring the security of data transmission and storage, and provides remote control capabilities for convenient device monitoring and control. Additionally, MQTT is equipped with a rule engine that enables customized data processing, offering users a more convenient data processing solution, as shown in Figure 10.

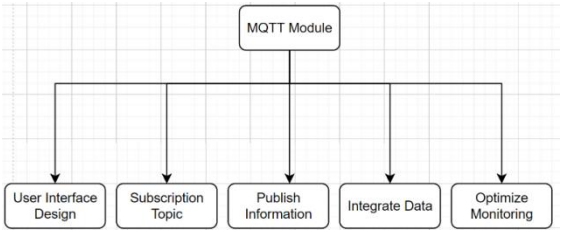

Figure 10. MQTT Module Diagram

When designing the MQTT interface, it is essential to first consider its lightweight and efficient characteristics, ensuring that the interface is simple and easy to operate. A publish/subscribe model can be adopted to allow users to easily publish and subscribe to messages. The interface should include a connection settings area, where users can input necessary information such as the MQTT server address, port, and client ID. Additionally, a status display area should be provided to show real-time connection status and message transmission and reception. To enhance user experience, a message history area can also be included, allowing users to conveniently view past messages. During the design process, attention should be paid to color coordination and layout rationality, ensuring that users can quickly get started and use the MQTT interface efficiently, as shown in Figure 11.

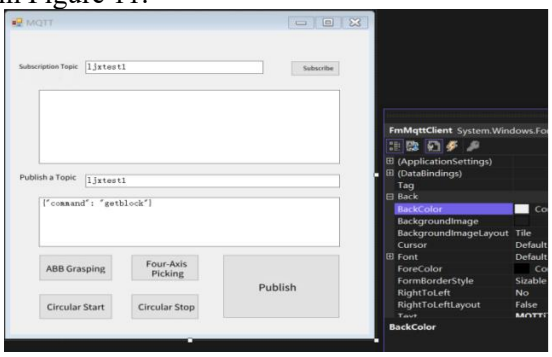

Figure 11. MQTT Interface Design Diagram

When constructing the interaction bridge between the digital twin system and the robotic arm [8], the introduction of the MQTT communication protocol is crucial, as it ensures real-time and reliable data transmission between the two. The core of this step lies in precisely configuring the MQTT communication link, enabling the digital twin system to accurately send control commands to the robotic arm. Subsequently, the control commands sent by the digital twin system are carefully encoded and transmitted through the "high-speed channel" of the MQTT protocol, arriving accurately at the robotic arm's receiving end. Upon receiving these commands, the robotic arm quickly parses them and executes the corresponding actions, such as moving, rotating, or grasping, based on the content of the commands.

Moreover, to meet the needs of user interaction, our scripts also need to be capable of handling signals from various input devices, such as

keyboards, mice, or game controllers. By listening to and parsing these input signals, we can achieve real-time control of the robotic arm by users, further enhancing its practicality and interactivity. The robotic arm also continuously sends its execution status, such as position, speed, load, and other key information, back to the digital twin system through the MQTT protocol.

This feedback mechanism provides the digital twin system with valuable real-time data, enabling it to accurately monitor the robotic arm's operating status and performance. The flowchart is shown in Figure 12.

Based on these real-time data, the digital twin system is able to continuously monitor and optimize the robotic arm. Once a decline in performance or potential malfunction of the robotic arm is detected, the system can quickly respond by adjusting the control strategy or issuing an alert, thereby ensuring the robotic arm operates continuously, stably, and efficiently [9].

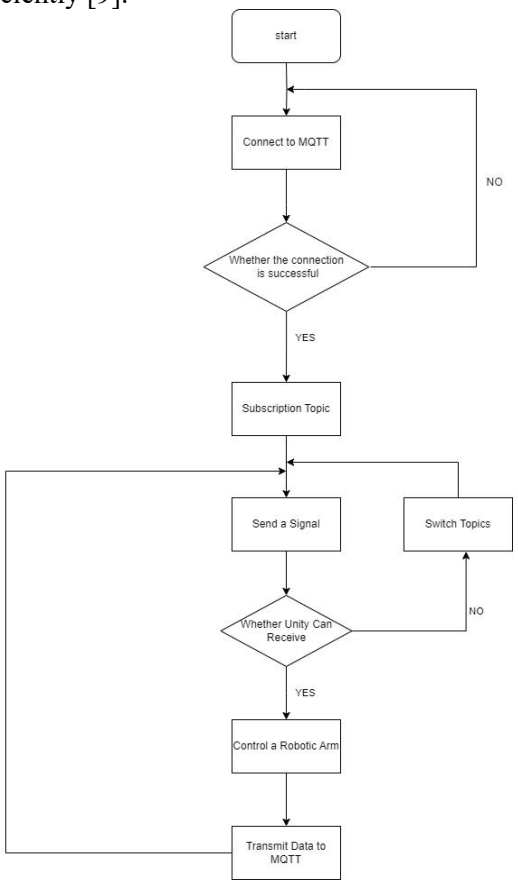

Figure 12. MQTT Signal Transmission Flowchart

4.4 Experimental Data

By receiving real-time signals from the client, the continuous motion of the robotic arm's joints can be initiated, allowing it to pick up the target object from the circular conveyor belt using a pneumatic suction cup and place it smoothly onto the adjacent orange placement platform. The process of achieving the four-axis suction effect in Unity is shown in Figure 13.

By receiving real-time signals from the client, pressing the button can initiate the continuous motion of the robotic arm's joints [10], enabling it to consecutively grasp two target objects from the placement platform using its gripper, combine them, and then place them smoothly onto the circular conveyor belt. The process of implementing the six-axis ABB robotic arm's grasping effect in Unity is shown in Figure 14.

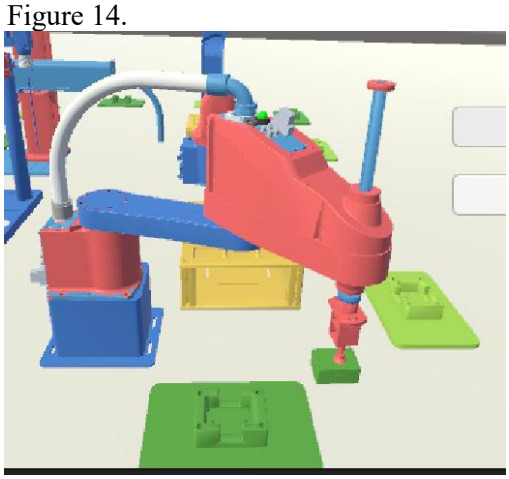

Figure 13. Four-Axis Robotic Arm Suction Process Diagram

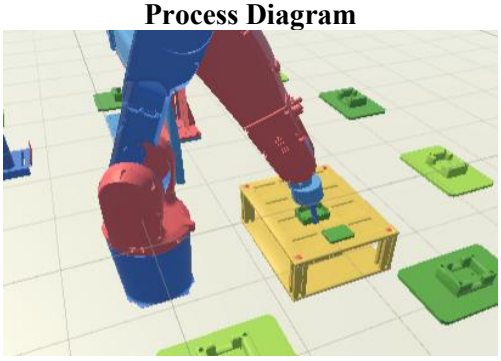

Figure 14. Six-Axis ABB Robotic Arm Grasping Process Diagram

At the initial stage of system design, we predefined corresponding transmission signals for each functional button. When these buttons are pressed, they send specific signals through the MQTT protocol. In the Unity development environment, these MQTT signals are received in real-time and displayed in the console. We have designed corresponding processing functions for each received signal to

implement the functions triggered by the buttons. For example, one signal may control the movement of the circular conveyor belt, while another signal may instruct the robotic arm's suction cup to pick up the target object [11]. Through this design, each operation of the functional buttons receives precise and logically clear responses. The signal transmission information is shown in Figures 15 and 16.

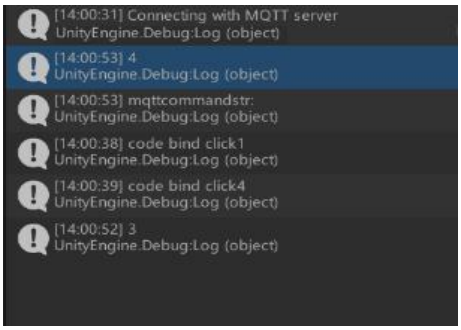

Figure 15. Data Transmission Process

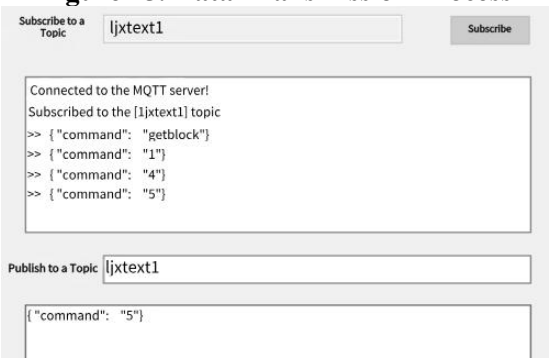

Figure 16. MQTT Subscription Test Diagram

5. Conclusion

Cross-platform communication plays a crucial role in the context of signal transmission and subscription, ensuring the effective flow and interaction of information across different operating systems, various devices, and diverse network environments. In this process, the signal sender encodes the signal into a universal format using a specific encoding method and transmits it via the corresponding communication protocol. Correspondingly, the signal receiver must be capable of decoding the signal in this format and then executing the appropriate operation or response based on the specific content of the signal. This mechanism makes cross-platform communication indispensable in signal transmission and subscription.

Through the MQTT protocol, users can remotely control a six-axis ABB robotic arm to perform precise grasping and assemble models using buttons. Once the model assembly is complete, the circular conveyor belt is activated to transport the items smoothly and efficiently to a designated area for manual labeling and inspection. Here, the items undergo quality inspection and label printing by human operators. After inspection, qualified products are precisely grasped by a four-axis suction robotic arm and placed on a designated placement platform, completing the entire process.

In the future, digital twin systems will focus more on optimizing algorithms and enhancing the level of intelligence to more efficiently handle large-scale, high-complexity data and improve the real-time performance and accuracy of models. At the same time, with the rapid development of technologies such as the Internet of Things (IoT), cloud computing, and big data, digital twin systems will achieve broader data interconnection and sharing, providing more comprehensive information support for intelligent manufacturing.

Acknowledgments

This research was supported by the Zhejiang Provincial. Philosophy and Social Science Planning Project under Grant. 2NDJC127YB, Ningbo City's 'Five Projects' - Research on the Talent Cultivation Model for Industry-Education Integration in the New Generation Information Technology Industry, Ningbo Science and Technology Fund under Grant. (2023Z228, 2023Z213. 2024Z126. 2024Z202, 2024Z119). College Level Research Project of College of Science and Technology, Ningbo University (YK202301). General Research Projects of Zhejiang Provincial Department of Education(Y202456101).

References

[1] Y. Lou, Z. Wang, S. Du, W. Liu and S. Kong, "A Hybrid Data Storage Method for a Robotic Arm Digital Twin Prototype, " 2022 6th International Conference on Automation, Control and Robots (ICACR), Shanghai, China, 2022, pp. 83-87, DOI:10.1109/ ICACR55854.2022.9935549.

[2] Qing Zhao. Research and Implementation of Digital Twin System for a Robot Assembly Line Based on Unity 3D. Xidian University, 2023. DOI: 10.27389/d. cnki. gxadu.2023.001630.

[3] Chengzhi Lin, Hua Huang, Shixiang Zhang. Design of Digital Twin Platform for Automated Production Line Based on Unity. Journal of Tangshan Teachers College, 2023, 36 (03): 38-43+84. DOI: 10.16160/j. cnki. tsxyxb. 2023.03.007.

[4] Jiang Qian. Research on Virtual Teaching Technology and Development of Teaching and Training System of Industrial Robots based on Digital Twin. Zhe Jiang University, 2022. DOI: 10.27461/d.cnki.gzjdx.2022.000129.

[5] J. Nwoke, M. Milanesi, J. Viola, Y. Chen and A. Visioli, "A Reduced-Order Digital Twin FPGA-Based Implementation with Self-Awareness Capabilities for Power Electronics Applications, " in IEEE Journal of Radio Frequency Identification, vol. 8, pp. 493-505, 2024, DOI: 10.1109/JRFID.2024.3404563.

[6] Hepeng Su, Hongbin Miao, Huijun Ji. Research on Collision Detection Method of Grasping System for Digital Twin Manipulator. Machine Tool & Hydraulics, 2024, 52 (08): 159-166.

[7] Zhangwen Lin. Research on Grasping Obstacle Avoidance and Virtual Debugging of Digital Twin Lower Manipulator. San Xia University, 2023. DOI: 10.27270/d.cnki.gsxau.2023.000311.

[8] Zixuan Hui. Development of a Virtual Experimental System for Robotic Arms Based on Digital Twins. North China University of Technology, 2024. DOI: 10.26926/d.cnki.gbfgu.2024.000668.

[9] Qijie Mou, Yang Xiang, Zexin Luo. Automatic Generation Technology of Kinematic Models for Digital Twin Simulation of Robotic Arms. Aviation Precision Manufacturing Technology, 2024, 14(04):123-126. DOI: 10.16667/j.issn.2095-1302.2024.04.032.

[10] Haixiang Guo, Ling Zhu, Jiannan Jiang. Research on the digital twin manipulator grab system based on Unity3D. Journal of Qiqihar University (Natural Science Edition), 2024, 40 (02): 33-39.

[11] Fei Ma, Kun Dai, Weiwei Sun. Design and Implementation of a Digital Twin Virtual Debugging System for Warehouse and Logistics Communication. Machine Tool & Hydraulics, 2023, 51 (16): 95-100.