Quality Festival 2025
ISBN 978-86-6335-121-9

Gligorije Mirkov [1]
Miladin Stefanović

Research paper
DOI – – 10.24874/QF.25.079

OPC UA IN INDUSTRIAL AUTOMATION: INTEGRATION WITH CNC SYSTEMS AND COMPARATIVE ANALYSIS WITH DNC

Abstract: *OPC UA (Open Platform Communications Unified Architecture) is a modern industrial communication standard designed for secure, reliable, and interoperable data exchange between different devices, applications, and systems within Industry 4.0. Compared to traditional protocols, OPC UA offers several significant advantages, with its main benefit being platform independence, enabling communication between devices from different manufacturers without requiring specific hardware interfaces or custom drivers. OPC UA provides bidirectional communication, allowing not only the transmission of NC programs but also real-time monitoring of machine status and process data. Additionally, OPC UA is scalable, meaning it can be used not only for connecting CNC machines but also for integrating entire manufacturing systems, including SCADA, MES, and ERP solutions.*

The advantages of OPC UA over DNC and older protocols make it an ideal choice for industrial environments that require flexibility, security, and integration with modern technologies. Its capability to operate in distributed systems, compatibility with cloud and IoT (Internet of Things) technologies, and ability to automatically adapt to complex network infrastructures make it a key element of future industrial systems.

Keywords: *OPC UA, DNC, Industrial Automation, CNC Communication, Industry 4.0, IIoT, SCADA, Data Security.*

1. Introduction

OPC UA (Open Platform Communications Unified Architecture) is an industrial communication standard developed to ensure interoperability between various devices and systems in automation. It was developed by the OPC Foundation as a successor to the classic OPC standard. According to (SIEMENS, 2025), the open, platform-neutral, and flexibly scalable OPC UA communication standard has a unique information model. The literature used in the development of the OPC UA system can be categorized based on the type of sources, main topics, and practical applications. Key sources include technical specifications, professional books, manuals, and scientific papers, each playing a specific role in the implementation of OPC UA systems.

[1] Corresponding author: Gligorije Mirkov
Email: gmirkov@sbb.rs

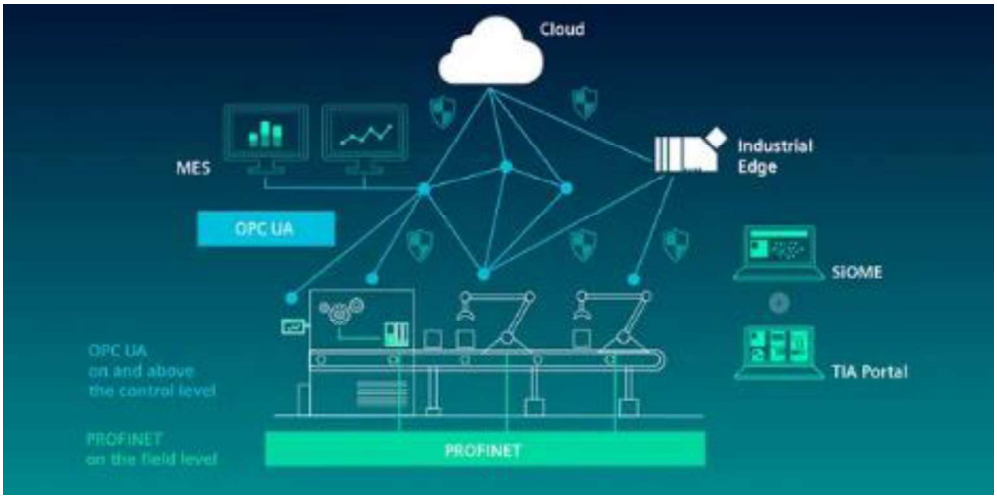

Figure 1. OPC UA (Download from OPC UA SIEMENS Web Site)

1.1. Technical Specifications

One of the most important sources is (OPC Foundation, 2022), which provides a detailed description of the OPC UA architecture through technical documentation divided into fourteen sections. This specification covers key aspects, including communication models, security mechanisms, data models, and interoperability with other systems. This document serves as the foundation for developing OPC UA servers and clients and is used in industrial systems worldwide.

1.2. Professional Books and Manuals

To understand the OPC UA standard and its application in the industry, books and manuals play a significant role. The work (Mahugnon & Dubois, 2021) provides a basic introduction to OPC UA, focusing on the concept of data models, security, and practical implementation in industrial automation. This work is particularly useful for engineers and beginners aiming to understand how OPC UA functions in real systems.

On the other hand, the book (Hoppe & Mihoc, 2020) offers practical guidelines for developers creating OPC UA applications. It thoroughly explains the development of OPC UA servers and clients, data handling, and working with methods within the OPC UA model. The book (Falk & Niemann, 2018) goes a step further, covering advanced industrial communication concepts, data standardization, and the application of OPC UA in SCADA[1] and IIoT systems.

For developers and technical engineers, the (Unified Automation GmbH, 2023) manual is an invaluable source of information. It provides detailed instructions on implementing OPC UA applications using programming languages such as Python (Rossum,2009) and C++, along with code examples. This manual is particularly useful for those looking to directly implement OPC UA into their systems.

1.3. Scientific Papers and Industrial Application

[1] SCADA (Supervisory Control and Data Acquisition) is a system for monitoring and controlling industrial processes in real time. It is used for collecting data from sensors and devices, visualizing it, and remotely controlling production processes. The main functions of a SCADA system include: real-time data collection and processing, remote control of devices (valves, pumps, motors), alarming in case of system malfunctions, and archiving data for later analysis.

Scientific papers provide an in-depth analysis of the OPC UA standard and comparisons with other industrial communication protocols. The paper (Zhou, Diedrich & Wisniewski, 2019) examines the use of OPC UA within cyber-physical systems (CPS), highlighting its role in Industry 4.0 and smart factories. It focuses on the performance and integration capabilities of OPC UA with advanced production management systems.

The paper (Jirkovsky et al., 2017) compares OPC UA and MQTT in the context of industrial automation and IIoT systems. The authors analyze the advantages and disadvantages of both protocols, particularly regarding efficiency, latency, and security. This paper helps in selecting the appropriate communication protocol for specific industrial applications.

1.4. Practical Application in Industry

Based on the literature mentioned, OPC UA is used in various industrial sectors, including:

- **Industry 4.0 –** enables automation and digitization of manufacturing processes in smart factories.
- **SCADA systems –** facilitates secure communication between controllers, sensors, and monitoring systems.
- **IIoT (Industrial Internet of Things) –** integrating OPC UA with IoT technologies allows real-time monitoring and optimization of production processes.

The referenced literature covers all key aspects of the OPC UA standard, from fundamental architecture and programming to industrial applications and academic analyses. Technical specifications serve as the foundation for understanding the standard, while specialized books and manuals assist in practical implementation. Scientific papers provide deeper analysis and comparisons of OPC UA with other protocols, whereas industrial applications highlight its crucial role in modern automated systems.

2. OPC and OPC UA Standard

During the 1970s, the first flexible manufacturing systems and specific communication protocols for file-related operations, data collection, and remote control of NC machine tools emerged. Initially, these protocols were based on LSV2 or DK3964R procedures for serial point-to-point communication (HEIDENHAIN, 2023). Later, communication evolved to be based on Ethernet and the TCP/IP protocol, such as the MCIS RPC Sinumerik Computer Link developed by Siemens (SIEMENS, 2005). The following section presents the history and development path of OPC UA.

2.1. Beginning of the OPC Standard

- **1996**: OPC Foundation was established to address communication challenges between industrial devices from different manufacturers. At that time, the classical OPC standard (known as OLE for Process Control) was developed, based on Microsoft technologies such as COM and DCOM.
- The goal of classical OPC was to enable communication between SCADA systems, PLCs, and other industrial controllers. However, its dependence on the Windows platform and related technologies became a limiting factor.

2.2. The Need for a More Modern Approach

The classical OPC standard had several issues, including:

- Dependence on Microsoft technologies (COM/DCOM).

- Security and scalability challenges.
- Difficulties in heterogeneous network environments (e.g., working across multiple operating systems).

Due to these limitations, the OPC Foundation (OPC Foundation, 2022) decided to develop a new standard that would be platform-independent, flexible, and more secure.

2.3. Development of OPC UA

2006: The first version of the OPC UA specification was published:

- OPC UA represents a significant improvement over classical OPC because it is platform-independent (it can run on Windows, Linux, and other operating systems).
- It uses modern protocols (such as TCP/IP and HTTPS) and supports communication through SOAP and binary encoding.
- Implements advanced security mechanisms, including authentication, encryption, and access control.

2.4. Evolution and Standardization

- **2008-2010**: Adoption of the OPC UA standard grows among industrial manufacturers, particularly in the fields of automation and IoT.
- **2013**: OPC UA becomes part of the IEC 62541 standard, further solidifying its importance as a global industrial standard.
- **2015-2020**: OPC UA plays a crucial role in concepts such as Industry 4.0 and smart factories, providing interoperability between devices, sensors, and cloud technologies (Zhou, Diedrich & Wisniewski, 2019).
- **2021 and beyond**: The standard continues to evolve to support increasingly complex requirements, including edge computing, artificial intelligence, and an expanded IoT ecosystem.

2.5. Characteristics of OPC UA

OPC UA has several key features that make it a powerful standard for industrial communication:

- **Platform Independence –** It can operate on different operating systems and hardware platforms.
- **Scalability –** Applicable to resource-constrained sensors as well as powerful server solutions.
- **Interoperability –** Facilitates integration of various devices and systems.
- **Security –** Advanced security protocols provide data confidentiality, integrity, and protection.
- **Data Model –** A flexible model for describing complex relationships between data.

3. OPC UA Communication Between Client and Server

3.1. The Importance of OPC UA Today

OPC UA is considered one of the key technologies in industrial automation, supporting initiatives such as Industry 4.0 and digital transformation. Its ability to integrate various technologies and standards makes it the foundation of modern industrial solutions (Jirkovsky et al., 2017).

In the OPC UA architecture, communication between the client and server plays a crucial role in data exchange and control operations in industrial systems. Below is an overview of what OPC represents on the server side and the client side.

3.2. OPC on the Server Side

The OPC server is the central element of the system, with its characteristics and tasks summarized in Table 1.

Table 1. Overview of OPC Server Characteristics

OPC on the Server Side	Description
Data Access	The server is responsible for collecting, processing, and providing data from industrial devices (such as PLCs, sensors, actuators) according to the OPC UA standard.
Data Model Implementation	- The server uses the OPC UA Information Model standard to structure and organize data. - Contains nodes representing data, functionalities, or relationships in the system.
Handling Client Requests	- The server responds to client requests for reading, writing, data subscriptions, or method execution. - Provides information on the current device state, alarms, and events.
Security	Ensures secure authentication, authorization, and encryption of communication with clients.
Communication Protocols	Uses OPC UA protocols such as TCP/IP, HTTPS, or MQTT.
Example Tasks of an OPC Server: - Reading the current temperature from a sensor. - Forwarding overheat alarms.- Providing access to historical data (e.g., log values).	

3.3. OPC on the Client Side

The OPC client is an application on the client side, with its characteristics and tasks listed in Table 2.

Table 2. Overview of OPC Client Characteristics

OPC on the Client Side	Description
Initiates Communication	The client connects to the OPC server to access data and functions provided by the server.
Performs Operations	- Reads data (e.g., current temperature value). - Writes data to the server (e.g., setting a target value for a regulator). - Subscribes to data changes (subscription) for real-time updates.
Visualization and Analysis	Clients are often part of SCADA systems or HMIs (Human-Machine Interfaces), where data is visualized and analyzed.
User Interaction	Allows operators to manage the system, view alarms, or adjust parameters.
Adaptability	Can communicate with multiple servers simultaneously and integrate data from various sources.
Example Tasks of an OPC Client: - Displaying the current machine status on a control panel. - Sending commands to start or stop a production line. - Analyzing historical data for process optimization.	

Client-Server Relationship:

- **Server:** Passively waits for client requests and provides data or functionalities.
- **Client:** Actively sends requests to the server and uses data or functionalities for application needs.

3.4. Example Of Interaction

1. The client connects to the server using the OPC UA protocol.

2. The client sends a request to read the current sensor value.
3. The server processes the request and returns the current value.
4. The client visualizes the data on the user interface or uses the value for further analysis.

This interaction enables flexibility and modularity in industrial systems.

3.5. Comparative Analysis of DNC and OPC UA Characteristics

A comparison of the characteristics of DNC (Distributed Numerical Control) and OPC UA (Open Platform Communications Unified Architecture) is presented by examining their purpose, application, and technological features. Table 3 highlights the key characteristics.

If a CNC machine supports the TCP/IP protocol and allows access to the DNC communication status, there is some similarity with OPC UA in terms of the basic principle of data exchange over a network. DNC can use TCP/IP to transfer NC programs and monitor the basic status of the machine. OPC UA also uses TCP/IP for communication but enables the transfer of a much broader range of data. If DNC allows the presentation of machine status, that aspect is similar to the core OPC UA functionality for monitoring. However, there are several key differences between DNC communication over TCP/IP and the OPC UA protocol, as shown in Table 4.

Table 3. Characteristics of DNC and OPC UA

Characteristics	DNC	OPC UA
Purpose and Application	- DNC is a technology specifically designed for managing numerically controlled machines (CNC) in manufacturing plants.- Its main purpose is to send, receive, and manage NC programs (codes) between a computer and CNC machines.- Focuses on managing production operations, often related to machine tools.	- OPC UA is a universal standard for industrial communication and interoperability, designed for data exchange between various devices, systems, and applications.- Covers a wide range of industrial systems, including production lines, SCADA, HMI, IoT, and cloud technologies.- Provides flexibility for data management, alarms, historical records, and real-time control.
Specific vs. Universal	- Focused on narrowly defined tasks related to CNC machine management.- Has a specific role in transferring G-codes and other NC programs.- Does not provide broader functionalities such as alarms, security, or data modeling.	- A universal standard with broad applications in various industries.- Provides a flexible data model and the ability to integrate with various devices, including CNC machines.- Can serve as infrastructure for communication between DNC systems and other components within a broader industrial system.
Technology and Architecture	- Typically uses serial connections (RS-232) or network protocols for communication with CNC machines.- Has a simple communication model: file transfer or command exchange between a central computer and machines.	- A more modern technology supporting various network protocols, including TCP/IP, HTTP(S), MQTT, and others.- Supports complex data models, security features (authentication, authorization, encryption), and real-time communication.
Security	- Limited security features, often relying on physical system isolation or basic network security protocols.	- Built-in advanced security features, including: * Encryption of communication. * User and device authentication. * Access control to data.

Characteristics	DNC	OPC UA
Interoperabil.	- Focused on specific tasks within a single system (CNC machines).- Limited interoperability with other industrial systems.	- Designed for high interoperability, enabling the integration of different devices and systems within industrial and IoT networks.- Can communicate with CNC machines, SCADA systems, cloud applications, and other technologies.
Examples of Application: DNC: -Sending G-codes from CAD/CAM software to CNC machines. -Managing versions of NC programs and their storage. OPC UA: -Connecting CNC machines to the broader manufacturing system (SCADA, MES, ERP). -Monitoring machine status, alarms, and data analysis for production optimization.		

Table 4. Differences Between DNC Over TCP/IP and OPC UA

Characteristic	DNC over TCP/IP	OPC UA
Standardization	Specific to CNC devices, not standardized.	Industrial standard with interoperability.
Data Types	Focuses on NC programs and basic statuses.	Wide range of data: status, parameters, history.
Data Structure	Text-based data or specific format.	Structured data, node hierarchy.
Security	Limited, depends on implementation.	Built-in authentication and encryption.
Communication Model	Request-based protocol (pull model).	Supports requests, event-driven transfers, and subscriptions.
Interoperability	Focused on DNC; difficult to expand.	Supports a wide range of devices and software.
Real-time Support	Depends on implementation.	OPC UA often employs real-time monitoring.

4. CNC Machine Simulation in Industry 4.0 via TCP/IP Communication

4.1. Context and Issues

In modern industry, digitalization of manufacturing processes is crucial for increasing efficiency, reducing costs, and improving quality. Traditional CNC systems often operate in isolation, without the possibility of remote communication and integration with advanced industrial solutions like MES[2], ERP, SCADA, and IoT platforms. This lack of connectivity complicates machine status monitoring, process management, and production optimization.

4.2. Objective and Purpose of the Project

This program simulates a connected CNC machine in Industry 4.0 using the TCP/IP

[2] MES (Manufacturing Execution System) – The Manufacturing Execution System is a software system that monitors, tracks, and controls manufacturing processes in real time. Its main function is to connect the operational level of production with higher business systems (e.g., ERP) to improve efficiency and optimize production. The main functions of the MES system include: real-time monitoring of production processes, resource management – tools, workers, machines, data collection and analysis from production lines, quality management – monitoring deviations in production, and tracking OEE (Overall Equipment Effectiveness) – the efficiency of manufacturing equipment.

protocol for bidirectional communication between the machine and supervisory systems. By implementing the JSON[3] format, easy integration with IoT and IIoT technologies is enabled, and the system can serve as a foundation for advanced data analytics and predictive maintenance.

4.3. Key Features of the Program

- **CNC Machine Simulation –** Allows testing of network communication without physical equipment.
- **Bidirectional Communication –** The client can read the machine status and send commands.
- **JSON Data Format –** Facilitates integration with other industrial systems.
- **Easy Integration with OPC UA, MES, and ERP[4] Systems –** Can serve as a basis for further digitalization.
- **Foundation for Industry 4.0 –** The system can be expanded by adding sensor data, analytics, and AI solutions.

4.4. System Structure

The CNC program and its status are crucial for the server:

- The server simulates or communicates with the real CNC system, meaning it must know which program is currently loaded and its state (Running, Stopped, Reset).
- The CNC machine program can be located directly on the CNC machine (if it has its own computer) or on a computer acting as the server.
- The server uses this information to respond to client requests.

The server functions as the communication center:

- The server processes incoming requests and provides information about the CNC machine's status.
- If the server runs on the CNC machine's computer, it can directly access data (via DNC, OPC UA, or internal software like SINUMERIK, FANUC, HEIDENHAIN).
- If the server runs on a separate computer, it must have a communication method with the CNC machine (e.g., OPC UA, Modbus, DNC, TCP/IP).

The client is a separate entity that requests data from the server:

- The client does not communicate directly with the CNC machine but requests specific data from the server.
- The client can be a computer, industrial PC, MES/SCADA system, or mobile application.
- **Server –** Simulates the CNC machine, processes incoming requests, and sends data about the current status.
- **Client –** Can be an industrial PC, MES system, or mobile app that communicates with the server.

---

[3] JSON (JavaScript Object Notation) is a human-readable format for storing and exchanging data. JSON enables efficient and standardized data exchange between applications, making it an ideal format for Industry 4.0, IoT systems, and network communication.

[4] ERP (Enterprise Resource Planning) – Enterprise Resource Planning is integrated business software that enables the management of key business processes within a company, including production, logistics, finance, human resources, and procurement.

- **Data –** Is transmitted in JSON format, ensuring compatibility with various platforms.

The program, by using a simple TCP/IP simulation, can serve as a model for Industry 4.0, enabling advanced automation, monitoring, and control of CNC machines. With further system expansion, security protocols, cloud analytics, and artificial intelligence can be added to optimize production.

4.5. CNC Remote Monitor & Control Program on the Server Side

In Figure 2, the program and the processing of commands that the client can send are presented. The program implements a simple TCP server that simulates the CNC machine status and allows clients to: retrieve the current machine status, change override values (feed and spindle), and reset the NC program status. Communication between the client and the server occurs via JSON messages. Communication between the client and the server occurs via JSON messages.

Server Testing

- Starting the server:

- Run the server Python script (python server.py).
- The server will display the message: Server is listening on port 9999...

- Connecting the client via Telnet:

- Open the terminal and enter: telnet 127.0.0.1 9999
- After connecting, you can manually send JSON requests.

Connecting the Client via Python Script

```
import socket
import json

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("127.0.0.1", 9999))

# Submitting a status request
request = {"action": "get_status"}
client.send(json.dumps(request).encode("utf-8"))

# View server response
                                    response = client.recv(1024)
print(json.loads(response.decode("utf-8")))

client.close()
```

FIgure 2. Python Script

- The client connects to the server.
- Sends a JSON request for the machine status.
- Receives and displays the response.

The server simulates the CNC machine and enables bidirectional communication with clients via JSON messages over TCP/IP. The advantages of this server setup include:

- **Flexibility –** It can easily be extended by adding new commands.
- **Interoperability –** Uses the JSON format, making integration with other systems easier.
- **Easy integration with OPC UA –** This server can be connected to an OPC UA server to allow access to machine status data from industrial applications.

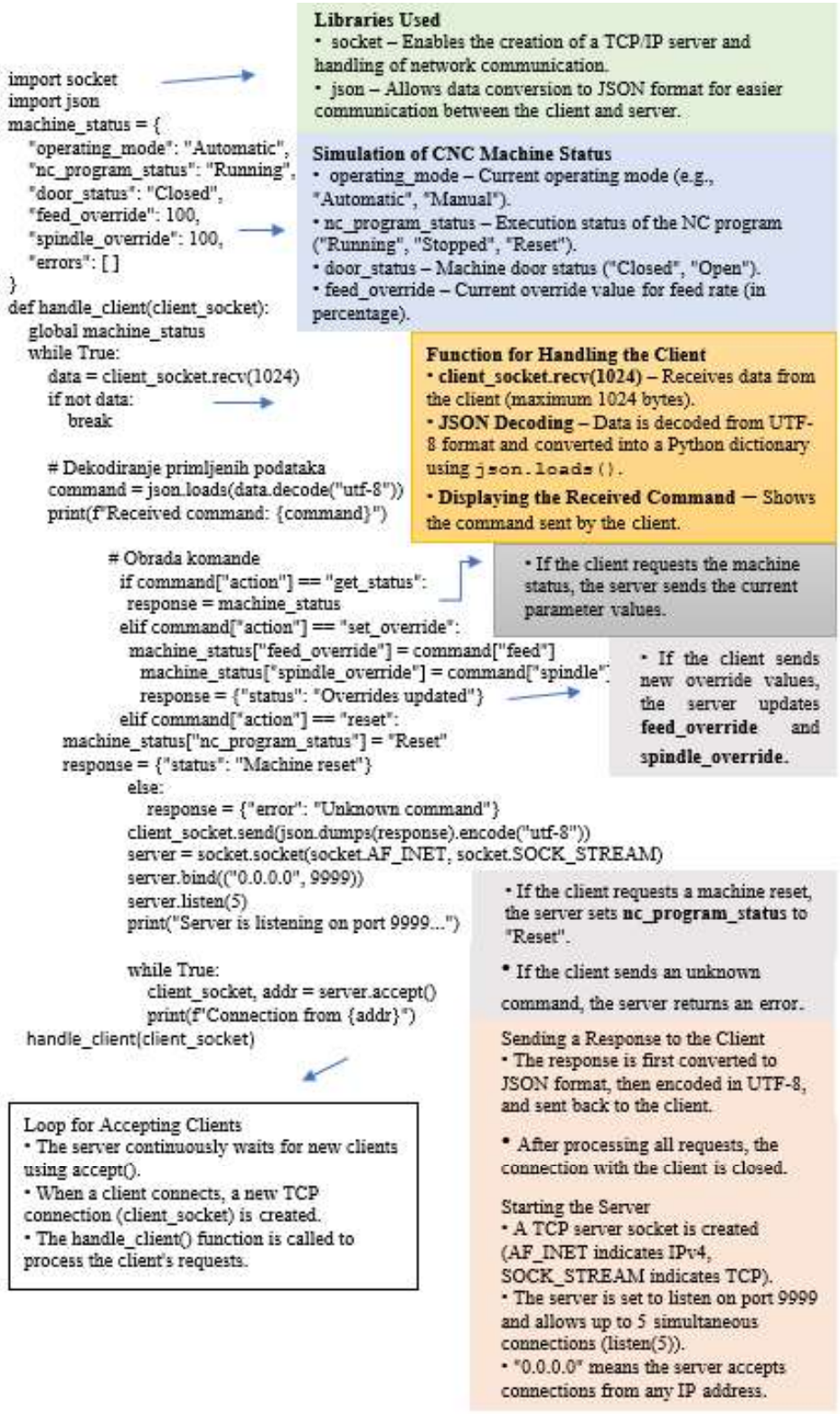

Figure 3. Program capabilities and command handling for client requests

One key question is: Is the simulation of this server aligned with Industry 4.0 (I4.0) principles? The answer is YES because it enables:

- **Digitalization and connection of industrial systems** – The CNC machine sends data over the network, enabling remote monitoring and control.
- **Bidirectional communication** – Clients can not only read data but also control the machine (e.g., change override values, reset).
- **Integration with advanced technologies** – This system can connect to OPC UA, MES, ERP, IoT, and IIoT platforms.
- **Predictive maintenance** – By adding sensor data, it is possible to analyze machine performance and detect potential failures before they occur.
- **Connection to cloud solutions** – The server can send data to the cloud for advanced analytics, AI processing, and production process optimization.
- **Industry 4.0 requirements** – These types of simulations are needed to test virtual machine copies (Digital Twins) and improve their functionality before deployment in real-world environments.

4.6. CNC Remote Monitor & Control Program on the Client Side

The client application will communicate with the previously written server program via TCP/IP and JSON format. The client program enables: receiving the machine status from the server, adjusting override values for feedrate and spindle, and resetting the machine by sending commands to the server (Figure 3). The program flow in phases is presented in Table 5. A command-line interface (CLI) is foreseen for the client-side program as described in the test procedure subsection.

Client Program (Python)

```
import socket
import json

# Function to send request to the server
def send_request(request):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(("127.0.0.1", 9999))  # Connect to server (local machine, port 9999)

    # Send data in JSON format
    client.send(json.dumps(request).encode("utf-8"))

    # Receive and decode server response
    response = client.recv(4096)
    client.close()

    return json.loads(response.decode("utf-8"))

# Function to display menu to user
def show_menu():
    while True:
        print("\n=== CNC Machine Client ===")
        print("1. Get machine status")
        print("2. Set override values (Feed & Spindle)")
        print("3. Reset machine")
```

```
        print("4. Exit")
        choice = input("Choose an option: ")

        if choice == "1":
            request = {"action": "get_status"}
            response = send_request(request)
            print("\nMachine Status:")
            for key, value in response.items():
                print(f"{key}: {value}")
        elif choice == "2":
            feed = int(input("Enter new feed override (0-150): "))
            spindle = int(input("Enter new spindle override (0-150): "))
            request = {"action": "set_override", "feed": feed, "spindle": spindle}
            response = send_request(request)
            print("\nServer Response:", response["status"])
        elif choice == "3":
            request = {"action": "reset"}
            response = send_request(request)
            print("\nServer Response:", response["status"])
        elif choice == "4":
            print("Exiting client...")
            break
        else:
            print("Invalid option. Please try again.")

# Run client
if __name__ == "__main__":
    show_menu()
```

Table 5. Program Flow

Connecting to the server	• The client uses the socket library to establish a TCP connection to the server (127.0.0.1, port 9999). • After connecting, the client sends a request in JSON format and waits for a response.
send_request() Function	• Establishes a connection with the server. • Sends the JSON request (e.g., "action": "get_status"). • Receives the response, decodes it, and returns it to the user.
Main Menu (show_menu())	• Allows the user to choose options: o Retrieve current machine status o Change override values for feedrate and spindle o Reset the machine o Exit the program
Interaction with the server	• If the user selects option 1, the client sends a status request ({"action": "get_status"}). • If the user selects option 2, the client asks for new feedrate and spindle values and sends them to the server ({"action": "set_override", "feed": 120, "spindle": 130}). • If the user selects option 3, the client sends a "reset" command to the server.

Test Procedure

Run the server: `python server.py`
The server will display the message:

Server is listening on port 9999...

Run the client: `python client.py`
Choose an option and follow the server's responses.

Example:

=== CNC Machine Client ===

1. Get machine status

2. Set override values (Feed & Spindle)

3. Reset machine

4.Exit

Choose an option: 1

The server response might be:

Machine Status:

operating_mode: Automatic
nc_program_status: Running
door_status: Closed
feed_override: 100
spindle_override: 100
errors: [ ]

Possible Program Extensions

- Add a graphical user interface (GUI) using Tkinter[5] for easier management.
- Implement security (e.g., user authentication[6]).
- Connect to a real CNC machine via OPC UA or Modbus protocol.
- Integrate with an IoT/MQTT broker for remote status monitoring.

The client program allows direct interaction with the CNC server, simulating a real industrial system. Through TCP/IP communication and JSON format, the client can monitor the machine status, adjust override values, and reset the process. It is easily extendable and can be used in Industry 4.0 applications.

One key question is when and under what conditions the software transitions from simulation to a real system. The program stops being a simulation when: it receives real data from the machine via industrial protocols, controls actual CNC processes (e.g., starting/stopping programs, adjusting parameters), operates in a real industrial environment with safety mechanisms, and integrates with other industrial systems such as MES, ERP, or SCADA systems. The first step towards reality is connecting the program to a real CNC controller using OPC UA or DNC communication.

5. Before the Conclusion

In this paper, standard Python libraries such as socket, (Python Software Foundation, 2023, Socket), json (Python Software Foundation,2023, Json), and tkinter (Python Software Foundation, 2023, Tkinter) were used, along with external libraries like python-opcua (OPC Foundation, 2023) and paho-mqtt, (Eclipse Foundation, 2023) which are documented and enable the development of OPC UA communication solutions in the Python programming environment.

Additionally, for broader context on industrial automation and standards within Industry 4.0, the work by Krivokapić et al. (2023), which addresses quality approaches in modern production systems, was referenced.

6. Conclusion

OPC UA uses advanced security mechanisms such as encryption, authentication, and access control, significantly improving data protection compared to DNC, which often operates without integrated security protocols. Additionally, OPC UA supports a semantically rich data model, allowing better organization and structuring of information,

[5] Tkinter is the standard Python library for developing graphical user interfaces (GUI).
[6] Authentication, according to Wikipedia, is the process of identifying a subject.

while traditional protocols like DNC transmit data in a simple format without additional semantic value. The developed programs demonstrate the basic principles of Industry 4.0, enabling digital communication between the CNC machine and other systems via a TCP/IP network. Through the implementation of a server-client architecture, remote monitoring and control of the CNC machine are enabled without the need for physical operator interaction. The use of JSON format for data exchange ensures easy interoperability with other industrial systems such as MES, SCADA, and ERP. By using UTF-8[7] encoding, compatibility with different languages and symbols is ensured, which is crucial for global industrial applications. The Tkinter GUI interface provides an intuitive way to interact with the system, making machine control accessible and simple. The implementation of override functionality allows users to dynamically adjust feedrate and spindle speed, while the reset option is added to simulate safety and operational procedures.

This system design provides an opportunity for expansion through integration with OPC UA, enabling greater flexibility and security in communication between devices. By implementing predictive maintenance, adding sensor data, and connecting to cloud platforms, the system can become fully automated and intelligent. Compared to traditional DNC protocols, this approach offers greater scalability, enhanced security, and better support for advanced analytics processes. Additionally, the efficiency of data transfer enables fast sharing of information between different levels of production management. This system can serve as the foundation for developing real industrial applications, where data would be collected directly from CNC machines and sent for processing in centralized servers or cloud systems. By using advanced data analysis methods and artificial intelligence, such a system could optimize machine operation and reduce maintenance costs.

The programs written in this project represent a practical example of Industry 4.0, using a combination of network protocols, graphical interfaces, and real-time data processing. This approach enables better control of the production process, greater work efficiency, and a reduction in human errors.

With further improvements, it is possible to add security mechanisms such as user authentication and data encryption to make the system resistant to network attacks and unauthorized changes to machine parameters. Additionally, adding MQTT communication could improve data transfer in IoT and IIoT environments. Finally, this project demonstrates that modern industrial applications can use Python and simple network protocols to enhance production. With further integration into real CNC machines, this system could become a fully operational solution that supports automatic control, optimization of production processes, and increased productivity.

In future versions, machine learning and artificial intelligence could be implemented so that the system could autonomously analyze data and optimize CNC machine operations in real-time. In this sense, this project lays the foundation for the further development of smart industrial systems that will become a key part of automated manufacturing in the future.

[7] UTF-8 (Unicode Transformation Format - 8-bit) is the most commonly used character encoding format, which allows the representation of text on a computer in a standardized form. It is a variable-width encoding, meaning it uses between 1 and 4 bytes to represent a single character.

References:

Eclipse Foundation. (2023). *paho-mqtt – MQTT client library for Python.* Retrieved from https://www.eclipse.org/paho/

Falk, D., & Niemann, K. (2018). Industrial Communication with OPC UA. *Springer.* doi:https://doi.org/10.1007/978-3-319-89947-4

HEIDENHAIN. (2023). *DNC RemoTools SDK virtualTNC - Software Interface and Transfer Components.* Retrieved from https://www.heidenhain.com/fileadmin/pdf/en/01_Products/Prospekte/PR_RemoTools_SDK_ID628968_en.pdf

Hoppe, T., & Mihoc, S. (2020). Practical OPC UA Programming for Engineers and Technicians. *ISA*.

Jirkovsky, V., Obitko, M., & Marik, V. (2017). OPC UA versus MQTT in Industry 4.0 Environments. *Procedia CIRP*, 62,150-155.

Krivokapic, Z., Vujovic, A., Jovanovic, J., Petrovic, S., & Pekovic, S. (2023). Quality approaches in Industry 4.0. *International Journal for Quality Research, 17*(1), 225–232.

Mahugnon, T., & Dubois, C. (2021). OPC UA: The Basics – Introduction to OPC UA for Industrial Automation. *Automation Press*.

OPC Foundation. (2022). OPC Unified Architecture Specification – Part 1 to 14. *OPC Foundation*. doi: https://opcfoundation.org

OPC Foundation. (2023). *python-opcua – OPC UA library for Python.* Retrieved from https://github.com/FreeOpcUa/python-opcua

Python Software Foundation. (2023). *tkinter – Standard Python interface to the Tk GUI toolkit (Python 3 documentation).* Retrieved from https://docs.python.org/3/library/tkinter.html

Python Software Foundation. (2023). *json – JSON encoder and decoder (Python 3 documentation).* Retrieved from https://docs.python.org/3/library/json.html

Python Software Foundation. (2023). *socket – Low-level networking interface (Python 3 documentation).* Retrieved from https://docs.python.org/3/library/socket.html

Rossum, G. V. (2009). *Python programming language (Version 3.x)*. (Python Software Foundation) Retrieved from https://www.python.org

SIEMENS. (2005). *Motion Control Information System – RPC SINUMERIK Computer Link*. Retrieved from Siemens Automation. https://www.automation.siemens.com

SIEMENS. (2025). *OPC UA – Structured data up to the cloud.* Retrieved from https://www.siemens.com/global/en/products/automation/industrial-communication/opc-ua.html

Unified Automation GmbH. (2023). Getting Started with OPC UA – A Developer's Guide. *Unified Automation*. doi:https://www.unified-automation.com

Zhou, L., Diedrich, C., & Wisniewski, L. (2019). OPC UA-Based Communication in Cyber-Physical Production Systems: Analysis, Design, and Implementation. *IEEE Transactions on Industrial Informatics*, 15(3),1482-1494. doi:https://doi.org/10.1109/TII.2019.2896313

Gligorije Mirkov
Belgrade,
Republic of Serbia
gmirkov@sbb.rs
ORCID 0000-0002-1153-0045

Miladin Stefanović
University of Kragujevac,
Faculty of Engineering Sciences
Kragujevac,
Republic of Serbia
miladin@kg.ac.rs
ORCID 0000-0002-2681-0875