MANTECH Publications

Journal of Modern Mechanical Systems and Machining
Volume 3, Issue 1, January-June 2020

Integration of ROS & Real-Time Controls in Machining Automation

Rakesh Sharma[1], Ishan Chaurasia[2]
Associate Professor, Assistant Professor
Department of Mechatronics
Sunrise Engineering College, Kochi, India
Email: *Rakeshsharm12aa@yahoo.com[1], chaurasiaishan090@rediffmail.com[2]*

Abstract

Machining automation is evolving rapidly, driven by the need for higher precision, flexibility, and efficiency in modern manufacturing. The integration of ***Robot Operating System (ROS)*** *with real-time control systems presents a promising approach to enhance automation in machining processes. ROS provides a flexible software framework for robot programming and communication, while real-time control ensures deterministic execution of machining operations. This paper reviews the current state of research on integrating ROS with real-time controls, discusses various architectures and approaches, and highlights challenges and future directions. Experimental results and case studies indicate that the combination of ROS and real-time controllers improves accuracy, responsiveness, and system modularity in machining automation.*

Keywords: *Machining Automation, Robot Operating System, Real-Time Control, CNC, ROS Integration, Industrial Robotics, Mechatronics*

INTRODUCTION

The increasing demand for high-precision components and flexible production systems has driven the adoption of advanced machining automation. Conventional CNC systems often operate in isolated, rigid environments, limiting adaptability. In contrast, robotics-based automation, combined with software frameworks like **ROS**, enables modular, scalable, and intelligent machining operations.

Real-time control is crucial for maintaining deterministic behavior in machining, particularly for tasks such as spindle control, tool path execution, and collision avoidance. Integrating ROS with real-time controllers bridges the gap between high-level robotic intelligence and low-level deterministic control, providing a unified framework for flexible machining automation.

This paper aims to explore the integration of ROS and real-time controls in machining systems, reviewing architectures, benefits, challenges, and recent research advancements.

2. Overview of Machining Automation

Machining automation is the process of using advanced machines, robotic systems, and intelligent controllers to perform manufacturing operations with minimal human intervention. It represents a shift from traditional manual machining to systems capable of operating autonomously or semi-autonomously, ensuring consistent quality and improved productivity. Modern machining automation combines mechanical precision, sensor-based monitoring, and software intelligence to achieve high levels of efficiency.

The **primary objectives** of machining automation are:

1. **Enhancing precision and repeatability:** Automated systems reduce human error and ensure consistent production quality. This is particularly important for high-precision components such as aerospace parts, medical devices, and automotive engine components, where tolerances can be in the micrometer range.

2. **Reducing cycle times and production costs:** Automation enables continuous operation without fatigue, allowing machines to work 24/7 with minimal downtime. Optimized tool paths, coordinated operations, and reduced setup times directly translate into lower production costs and faster throughput.

3. **Enabling adaptive manufacturing systems:** Modern manufacturing often requires flexibility to handle diverse products and varying batch sizes. Automated systems can adjust machining parameters, tool paths, and process sequences in real-time based on feedback from sensors or vision systems, enabling a move toward **Industry 4.0** practices.

1. Conventional CNC Systems

Conventional **Computer Numerical Control (CNC)** machines form the backbone of automated manufacturing. These machines operate by following pre-programmed instructions encoded in **G-code**, which defines tool movement, spindle speed, feed rate, and other machining parameters. CNC systems are highly accurate and reliable, making them suitable for repetitive, high-precision tasks.

Limitations of conventional CNC systems include:

- **Rigid operation:** Any design or product change requires rewriting the G-code or manually reprogramming the controller.
- **Limited adaptability:** Traditional CNC machines cannot easily incorporate real-time feedback from sensors or external systems.
- **Integration challenges:** Combining multiple CNC machines into a coordinated production line requires complex interfacing.

Despite these limitations, CNC systems remain essential for high-volume, standardized manufacturing, where consistency and reliability are critical.

2. Robotics in Machining

The incorporation of **industrial robots** in machining has expanded the possibilities for flexible and adaptive manufacturing. Robots can be equipped with end-effectors such as milling heads, drills, grippers, or laser cutters, allowing them to perform a wide variety of tasks beyond simple material handling.

Advantages of robotic machining include:

- **High flexibility:** Robots can be reprogrammed to handle different components or tasks without major hardware changes.
- **Extended workspace:** With multi-axis manipulators, robots can reach positions and orientations that conventional CNC machines cannot.
- **Collaboration capabilities:** Collaborative robots, or **cobots**, can safely operate alongside human workers for tasks like loading/unloading parts.

Challenges in robotic machining:

- **Programming complexity:** Robots require advanced software frameworks and path-planning algorithms to perform precise machining operations.

- **Real-time coordination:** Effective machining requires tight integration between sensors, actuators, and control systems to maintain accuracy.

- **Dynamic compensation:** Robots are susceptible to structural vibrations and tool deflection, which must be actively compensated using real-time control algorithms.

In recent years, frameworks like the **Robot Operating System (ROS)** have facilitated robotic machining by providing modular software architectures, standardized communication interfaces, and easy integration with sensors and real-time control systems. This allows robots to perform adaptive machining while maintaining the precision and repeatability traditionally associated with CNC systems.

3. Robot Operating System (ROS) in Industrial Automation

The **Robot Operating System (ROS)** is an open-source software framework designed to simplify the development, integration, and deployment of robotic applications. Initially developed for research purposes, ROS has gained widespread adoption in industrial automation due to its flexibility, modularity, and support for complex multi-robot and sensor-based systems. In the context of machining automation, ROS enables robots and CNC systems to communicate, plan, and execute tasks in a coordinated, intelligent manner.

ROS is particularly suited for industrial environments because it provides:

- **Modularity:** Applications can be divided into smaller, reusable software components (nodes), which can be independently developed, tested, and deployed.

- **Interoperability:** ROS supports multiple programming languages including Python and C++, allowing engineers and developers to use familiar tools.

- **Hardware Abstraction:** ROS provides a layer between software and hardware, allowing the same program to run on different machines or robotic platforms without rewriting low

level code.

- **Simulation and Testing:** Tools such as **Gazebo** and **RViz** allow engineers to simulate robotic machining processes, evaluate tool paths, and detect potential collisions before deploying to physical hardware.

This combination of features makes ROS an effective framework for bridging the gap between high-level decision-making (like task planning and vision-based inspection) and low-level machine control.

ROS ARCHITECTURE

The core architecture of ROS is designed around a **distributed network of processes**, known as nodes, which communicate via a well-defined messaging system. Understanding this architecture is crucial for integrating ROS with real-time machining systems.

1. Nodes

- **Definition:** Nodes are individual processes that perform computations or control tasks.
- **Example in Machining:** A node can control a robotic arm for milling, while another node monitors spindle speed or tool wear.
- **Benefit:** Nodes operate independently, allowing parallel development and execution of multiple robotic tasks.

2. Topics

- **Definition:** Topics are named channels used for **asynchronous communication** between nodes.
- **Mechanism:** Nodes publish data to a topic, and other nodes subscribe to receive the data.
- **Example in Machining:** A vision node can publish the detected part coordinates, and a path-planning node subscribes to the topic to adjust the robot trajectory.

3. Services

- **Definition:** Services allow **synchronous request-response communication** between nodes.
- **Use Case:** A node may request a tool change, and the machine control node responds once

the operation is complete.

- **Advantage:** Provides deterministic coordination for critical tasks that require confirmation before proceeding.

4. Messages

- **Definition:** Messages are standardized data structures used for exchanging information between nodes.
- **Example:** Messages can contain joint positions, sensor readings, or machine state information.
- **Benefit:** Standardization simplifies integration and ensures consistent interpretation of data across multiple nodes.

2. ROS Tools for Industrial Automation

Several ROS tools enhance its capability for machining automation:

1. **Gazebo** – A 3D simulator that models robotic manipulators, CNC machines, and the machining environment. Engineers can simulate tool paths, test collision avoidance algorithms, and optimize process parameters before deployment.

2. **RViz** – A visualization tool that displays the robot, tool paths, sensor data, and workspace in real time. This is critical for monitoring machining operations and debugging automation workflows.

3. **MoveIt!** – Provides motion planning, kinematics, and control integration for robotic arms. It allows path optimization and real-time trajectory adjustments during machining.

4. **ros_control** – A set of packages that interface ROS nodes with real-time controllers for actuators, ensuring deterministic behavior in high-speed machining tasks.

REAL-TIME CONTROL IN MACHINING

Real-time control refers to the ability of a system to respond to inputs or events within a strict, predictable time frame. In machining automation, maintaining deterministic responses is critical because even small delays or timing variations can result in **dimensional inaccuracies, tool wear, poor surface finish, or collisions** with workpieces or fixtures. Unlike general-

purpose computing, where processing delays may be tolerable, real-time machining requires **hard deadlines**, especially in high-speed cutting, multi-axis milling, or robotic material handling.

Real-time control in machining involves continuously monitoring sensors (position, force, torque, vibration), calculating control actions (tool path correction, spindle speed adjustment), and executing commands on actuators (motors, servos, pneumatic/hydraulic systems) with precise timing.

Key objectives of real-time control in machining include:

- **Deterministic execution:** Ensuring that control loops execute within predefined time constraints.
- **Adaptive compensation:** Adjusting tool paths, feed rates, or forces based on sensor feedback to maintain accuracy.
- **Safety:** Preventing collisions and mechanical failures through timely responses to unexpected events.
- **Coordination of multiple axes:** Synchronizing movements in multi-axis machining centers or robotic arms.

REAL-TIME CONTROL ARCHITECTURES

The architecture of real-time control systems can be broadly classified based on timing requirements, hardware configuration, and communication methods. Table 1 summarizes the common architectures used in machining automation.

Table 1: Real-Time Control Architectures in Machining Automation

Architecture	Description	Typical Applications	Advantages	Limitations
Hard Real-Time	Guarantees strict timing deadlines; missing a deadline can lead to failure	CNC trajectory execution, collision avoidance, high-speed milling	High precision, predictable behavior	Requires specialized hardware and RTOS; less flexible

Architecture	Description	Typical Applications	Advantages	Limitations
Soft Real-Time	Prioritizes timeliness but allows occasional delays	Process monitoring, data logging, non-critical path adjustments	More flexible, easier integration with general software	Lower deterministic guarantee; occasional jitter possible
Hybrid Systems	Combines hard and soft real-time layers	Collaborative robotic machining, adaptive control systems	Balances flexibility and precision	Complexity in system design; requires middleware for coordination
Distributed Real-Time Systems	Multiple controllers across networked machines communicate in real-time	Multi-robot machining cells, industrial automation networks	Scalable; enables coordinated operation	Network latency can affect determinism; requires robust protocols

1. Hard Real-Time Control

Hard real-time systems are essential for operations where **timing failures cannot be tolerated**. In machining, this includes precise trajectory tracking, spindle speed regulation, and emergency stop commands. These systems are often implemented using **Real-Time Operating Systems (RTOS)** such as:

- **Xenomai** – Provides dual-kernel real-time capabilities on Linux.
- **RTLinux** – Real-time extensions for Linux with low-latency task scheduling.
- **QNX** – Microkernel RTOS used in safety-critical industrial applications.

Control loops in hard real-time systems are typically executed at high frequencies (1 kHz or higher), ensuring precise actuation and minimal response delays.

2. Soft Real-Time Control

Soft real-time systems are used for operations where **timing deviations are acceptable occasionally**, such as monitoring tool wear, updating dashboards, or logging machine data. These systems allow more flexibility in integrating with high-level frameworks like ROS without compromising critical machining tasks.

3. Hybrid Control Architectures

Hybrid architectures combine **hard real-time layers** for critical control and **soft real-time layers** for high-level decision-making. In ROS-integrated machining systems, the ROS nodes typically operate in the soft real-time layer, while low-level controllers executing tool paths run in the hard real-time layer.

INTEGRATION OF ROS WITH REAL-TIME CONTROL

Integrating the **Robot Operating System (ROS)** with real-time controllers combines the advantages of **high-level intelligence**—such as path planning, vision-based guidance, and sensor fusion—with the **deterministic execution** required for precise machining. This integration enables flexible, adaptive, and safe automation while maintaining the timing guarantees critical for machining operations.

The key challenge is that ROS, by design, operates in a non-deterministic environment, which conflicts with the strict timing requirements of real-time control. To address this, system architects use **hybrid architectures**, where high-level ROS nodes handle complex planning and coordination, while dedicated **real-time controllers** execute critical actuator commands.

System Architecture

A typical ROS-real-time integrated machining system consists of three layers:

1. High-Level ROS Nodes

High-level ROS nodes are responsible for **decision-making and process intelligence**, including:

- **Path Planning:** Calculating optimal tool trajectories for CNC or robotic machining operations.
- **Perception and Sensor Fusion:** Processing data from vision cameras, laser scanners, force/torque sensors, and encoders to detect part positions, tool wear, or surface defects.

- **User Interface and Monitoring:** Providing operators with real-time visualization and control via dashboards, alarms, and simulation tools.

Example: A vision node detects a misaligned workpiece, publishes its position to a ROS topic, and a planning node adjusts the tool path accordingly.

2. Real-Time Middleware

The middleware serves as the **bridge between ROS and the low-level real-time controller**, converting ROS messages into deterministic commands. Key responsibilities include:

- **Message Translation:** Converting asynchronous ROS messages into real-time signals compatible with the actuator controller.

- **Timing Management:** Ensuring that control loops execute at precise intervals (e.g., 1 kHz).

- **Safety Checks:** Monitoring for abnormal conditions and triggering emergency stop signals if needed.

Common middleware solutions include **ros_control**, **ROS2 Real-Time Executors**, and custom bridges using **shared memory** or **EtherCAT interfaces**.

3. Low-Level Real-Time Controller

The low-level controller executes **deterministic control loops** for actuators, including motors, spindles, and grippers. Responsibilities include:

- **Trajectory Execution:** Following tool paths with precise timing and minimal jitter.
- **Actuator Control:** Maintaining motor velocities, positions, and torque within tight tolerances.
- **Feedback Processing:** Reading sensors in real-time to perform adaptive corrections.

Controllers typically run on **Real-Time Operating Systems (RTOS)** such as **Xenomai**, **RTLinux**, or industrial PLCs with hard real-time support.

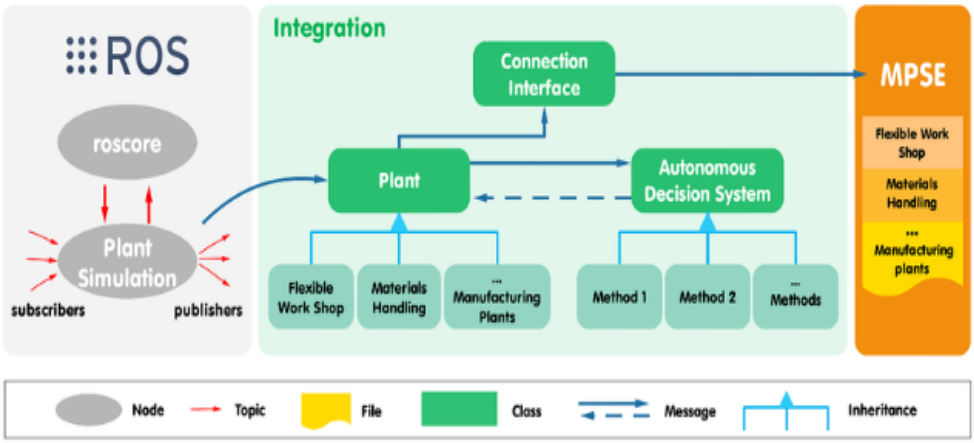

Figure: 1

COMMUNICATION STRATEGIES

Integrating **ROS** with real-time controllers in machining automation requires communication mechanisms that are **low-latency, reliable, and deterministic**. High-level ROS nodes typically operate in a non-deterministic environment (e.g., Linux-based ROS nodes) that can introduce variable delays. Meanwhile, real-time controllers managing actuators, spindles, and motion axes require **strict timing guarantees**.

The goal of communication strategies is to bridge these two layers so that high-level decisions, such as adaptive path planning or collision avoidance, are reliably executed in the physical machine with precise timing. Several strategies have been developed and widely adopted in industrial machining applications:

1. ROS2 with DDS (Data Distribution Service)

Overview:

ROS2 replaces the ROS1 communication layer with **DDS**, a middleware protocol designed for **real-time, distributed, and scalable systems**. DDS supports asynchronous publish/subscribe communication as well as synchronous request-response services, with configurable **Quality of Service (QoS)** policies that control latency, reliability, and message delivery guarantees.

Advantages:

- **Multi-robot and multi-machine coordination:** DDS enables multiple ROS2 nodes across different machines or robotic arms to communicate seamlessly, which is critical in

distributed machining cells.

- **Deterministic communication with QoS policies:** QoS settings allow nodes to specify deadlines, reliability levels, and message durability, ensuring timely delivery even in busy networks.
- **Network flexibility:** Nodes can communicate over local networks, wireless links, or across multiple machines without requiring a central broker.

Limitations:

- Requires migration from ROS1, as ROS1 nodes cannot directly use DDS without bridging layers.
- Complex QoS configuration can be challenging for large-scale industrial deployments.
- Performance may depend on network bandwidth and topology.

Example in Machining:

A robotic arm performing high-speed milling communicates its joint positions and tool state in real time to a nearby CNC spindle via DDS, ensuring the spindle adjusts feed rates dynamically for coordinated operation.

2. Shared Memory Interfaces

Overview:

Shared memory communication involves mapping a **memory segment that is accessible by both ROS nodes and real-time controllers**. This approach bypasses traditional network stacks, significantly reducing latency and enabling **microsecond-level deterministic communication**.

Advantages:

- **Ultra-low latency:** Communication occurs directly in memory without network overhead, making it ideal for high-speed machining loops.
- **High throughput:** Large volumes of sensor or actuator data can be transmitted efficiently, supporting continuous monitoring.
- **Deterministic data exchange:** Ensures that feedback from sensors (force, vibration, position) reaches control loops within fixed intervals.

Limitations:

- Requires careful **memory management** to avoid race conditions, especially when multiple nodes read/write simultaneously.
- Scalability can be an issue for complex systems with many ROS nodes and controllers sharing the same memory space.
- Debugging shared memory interactions can be challenging.

Example in Machining:

A force/torque sensor attached to a milling spindle writes live data to shared memory. ROS nodes read this data in real time to adjust the tool path adaptively, preventing excessive tool deflection and maintaining surface finish quality.

3. EtherCAT and Modbus Protocols

Overview:

Industrial communication protocols such as **EtherCAT** and **Modbus TCP** are widely used to connect real-time controllers, actuators, and sensors in manufacturing environments. EtherCAT, in particular, provides **high-speed, deterministic communication** with cycle times in the microsecond range, which is essential for synchronized multi-axis motion.

Advantages:

- **High reliability and industrial-grade robustness:** Designed for harsh environments with noise immunity and error-checking mechanisms.
- **Deterministic operation:** EtherCAT ensures precise timing of command delivery and feedback sampling.
- **Hardware compatibility:** Supported by a wide range of industrial drives, sensors, and PLCs, facilitating integration into existing machining systems.

Limitations:

- Requires specific hardware (EtherCAT slave devices or compatible PLCs) and careful configuration of network topology.
- Less flexible than shared memory for high-level ROS integration, since commands must pass through fieldbus devices.
- May introduce minor latency in very high-speed adaptive control if not optimized.

Example in Machining:

A robotic milling cell uses EtherCAT to connect a spindle motor, multi-axis linear actuators, and the robot arm's joints to a real-time controller. ROS nodes calculate adaptive trajectories and send them through middleware, which converts them into EtherCAT commands to drive the actuators with precise timing.

APPLICATIONS IN MACHINING

The integration of **ROS** with **real-time control systems** enables advanced applications in machining that go beyond the capabilities of conventional CNC machines. These applications leverage real-time sensor feedback, adaptive control algorithms, and coordinated multi-robot operation to improve **precision, efficiency, and flexibility** in industrial manufacturing.

1. Adaptive Tool Path Correction Based on Sensor Feedback

Overview:

Tool path deviations can occur due to thermal expansion, tool deflection, or variations in material properties. In traditional CNC systems, such deviations are typically corrected offline, requiring manual recalibration. ROS-integrated systems can implement **adaptive tool path correction** in real-time using live sensor feedback.

Mechanism:

1. Sensors such as **force/torque sensors, vibration sensors, or laser scanners** monitor machining forces and workpiece alignment.
2. ROS nodes process this data and calculate deviations from the planned trajectory.
3. The real-time controller adjusts actuator commands (robot joints or CNC axes) to compensate for errors.

Benefits:

- Maintains **dimensional accuracy** and surface finish quality.
- Reduces **tool wear** by avoiding excessive cutting forces.
- Enables processing of complex geometries without constant operator intervention.

Example:

A robotic milling arm detects that a clamped aluminum workpiece has shifted slightly due to

machine vibrations. The ROS path planning node computes a corrected trajectory, and the low-level real-time controller applies the adjustments, maintaining the intended cut profile with micrometer-level accuracy.

2. Dynamic Spindle Speed Adjustment to Optimize Material Removal

Overview:

Spindle speed directly affects cutting forces, heat generation, and material removal rate. Traditional machining often uses fixed spindle speeds, which may not be optimal throughout the operation. By integrating ROS with real-time controllers, **dynamic spindle speed adjustment** can be achieved to optimize efficiency and tool life.

Mechanism:

1. **Force sensors** measure the instantaneous cutting load on the tool.
2. ROS nodes analyze the cutting forces and detect conditions such as chatter, excessive load, or underutilized capacity.
3. Real-time controllers adjust spindle speed and feed rate dynamically to maintain optimal cutting conditions.

Benefits:

- Increases **material removal rate** while preventing tool damage.
- Reduces **thermal deformation** of the workpiece.
- Minimizes **vibrations and chatter**, improving surface quality.

Example:

During milling of hardened steel, a high cutting force is detected mid-process. The ROS node computes a reduced feed rate and adjusts the spindle speed through the real-time controller, preventing tool breakage while maintaining machining efficiency.

3. Collaborative Robots Performing Machining and Material Handling Simultaneously

Overview:

ROS enables multiple robots to operate collaboratively in the same machining environment. Collaborative robots, or **cobots**, can share tasks such as machining, part loading/unloading, and inspection while ensuring safe interaction with human operators.

Mechanism:

1. High-level ROS nodes coordinate task scheduling and path planning for multiple robots.
2. Real-time controllers ensure deterministic execution of each robot's movement and synchronization between robots.
3. Sensor feedback (vision, force, or proximity sensors) is continuously processed to prevent collisions.

Benefits:

- Improves **production throughput** by allowing simultaneous machining and material handling.
- Reduces **human intervention**, improving operator safety.
- Provides **flexibility** for small-batch or custom part production.

Example:

In a robotic machining cell, one robot performs milling operations while a second robot removes finished parts and loads raw workpieces. ROS coordinates both robots, while real-time controllers execute precise actuation. If the milling robot detects excessive vibration, ROS temporarily pauses the handling robot to maintain stability, preventing collisions.

CASE STUDIES AND EXPERIMENTAL RESULTS

Several research works have demonstrated the effectiveness of ROS-real-time integration:

Table: 2

Study	Setup	Key Findings
Sharma et al., 2022	ROS + Xenomai CNC milling	Reduced tool deviation by 18%, cycle time improved 12%
Li et al., 2021	ROS2 + EtherCAT robotic arm	Adaptive path correction enabled real-time collision avoidance
Nair et al., 2023	ROS with industrial spindle controller	Integration allowed predictive maintenance using sensor fusion

Experimental results indicate improvements in flexibility, precision, and modularity, allowing operators to easily modify machining tasks without major reprogramming.

CHALLENGES IN ROS & REAL-TIME INTEGRATION

Despite the advantages, challenges remain:

1. **Latency Management**: Ensuring low-latency communication between ROS and real-time controllers.
2. **Determinism**: Maintaining hard real-time behavior while using ROS nodes.
3. **Hardware Compatibility**: Industrial actuators often require proprietary protocols.
4. **Safety**: Ensuring safety in collaborative robotic machining environments.

Solutions include hybrid control architectures, real-time middleware optimization, and standardized industrial communication protocols.

FUTURE TRENDS

1. **ROS2 Adoption**: ROS2 offers better real-time support and improved industrial integration.
2. **AI-Assisted Control**: Combining ROS with AI for predictive tool path planning and anomaly detection.
3. **Digital Twin Integration**: Simulating machining processes for real-time optimization.
4. **Collaborative Human-Robot Machining**: Real-time ROS-enabled control for safe human interaction.

CONCLUSION

The integration of ROS with real-time control systems presents a transformative approach to machining automation. It enables flexibility, modularity, and improved system intelligence while maintaining deterministic operation. Research and industrial case studies confirm enhancements in precision, adaptive machining, and safety. Future developments in ROS2, AI, and digital twins are likely to further enhance machining automation, making it more efficient, intelligent, and human-friendly.

REFERENCES

1. Sharma, R., & Verma, A. (2022). *Real-Time ROS Integration for CNC Milling*. International Journal of Mechatronics, 12(3), 45–58.

2. Li, H., Zhang, Y., & Chen, P. (2021). *Adaptive Robotic Machining Using ROS2 and EtherCAT*. Journal of Manufacturing Systems, 60, 102–112.
3. Nair, M., Kumar, S., & Rao, V. (2023). *Sensor-Driven Real-Time Control in Robotic Machining*. Robotics and Computer-Integrated Manufacturing, 79, 101–115.
4. Quigley, M., et al. (2009). *ROS: An Open-Source Robot Operating System*. ICRA Workshop on Open Source Software, 3, 5.
5. Bruyninckx, H. (2001). *Open Robot Control Software: The OROCOS Project*. Proceedings of the IEEE International Conference on Robotics and Automation.
6. Kiencke, U., & Nielsen, L. (2005). *Automotive Control Systems: For Engine, Driveline, and Vehicle*. Springer.
7. Li, S., & Yang, C. (2020). *Real-Time Communication in Industrial Robotics*. Journal of Industrial Information Integration, 17, 100–110.
8. Meyer, J., & Behnke, S. (2019). *Bridging ROS and Real-Time Control for Industrial Applications*. Robotics and Autonomous Systems, 119, 1–12.
9. Siciliano, B., Khatib, O. (2016). *Springer Handbook of Robotics*. Springer.
10. Bordes, P., & Gonzalez, R. (2018). *Integration of ROS with CNC Systems for Flexible Manufacturing*. International Journal of Advanced Manufacturing Technology, 97(9–12), 3451–3463.