Introduction to Robotic Systems Integration

DASE7503 Robotic Systems Integration

Contents

- What is Robotic Systems Integration
- The Importances of Robotic Systems Integration
- Key Components of a Robotic System
- The Integration Process
- Key Challenges in Integration

What is Robotic Systems Integration?

- The process of combining various robotic components such as sensors, actuators, controllers and software into a comprehensive and efficient robotic system that performs desired tasks

- The integration of software platforms, middleware, communication protocols, and human-machine interfaces (HMIS) to ensure all parts work harmoniously



What is Robotic Systems Integration?

Key elements of robotic systems integration :

- **Sensor Fusion:** Combining data from multiple sensors (e.g., LIDAR, cameras, IMUs) for improved decision-making and system control.
- **Actuator Control:** Seamlessly managing actuators to perform movements with precision and responsiveness.
- **Middleware Solutions:** Using frameworks like ROS (Robot Operating System) to facilitate communication between various hardware and software components.
- **Human-Machine Interfaces (HMI):** Creating interfaces that allow operators to monitor, control, and adjust robotic behavior with ease.

The Importances of Robotic Systems Integration

- **Functionality**: Ensures different subsystems (sensors, actuators, controllers) operate harmoniously.
- **Reliability**: Reduces system failures by verifying compatibility and communication.
- **Efficiency**: Optimizes performance through coordinated control and data flow.
- **Customization**: Allows tailoring robots to meet specific application needs.
- **Improved Safety**: Can be more efficiently implemented and maintained with appropriate safety system such as emergency stops and real-time monitoring.

What are Robotic Systems?

- engineered assemblies that combine hardware and software components to perform tasks autonomously or semi-autonomously.
- designed to sense, process, and act within their environment to achieve specific objectives, often replacing or assisting human effort.
- expanding their capabilities and applications with the continuous advancements in AI, perception, and mobility.



Key Components of a Robotic System

- **Hardware:**
  - Mechanical parts (arms, wheels, chassis)
  - Sensors (cameras, LIDAR, force sensors)
  - Actuators (motors, servos)
  - Controllers (embedded boards, PLCs)
- **Software:**
  - Operating systems (ROS2, RTOS)
  - Middleware for communication
  - Algorithms for perception, planning, and control
- **Connectivity:**
  - Communication protocols (Ethernet, CAN, Wi-Fi)
  - Data interfaces and APIs

Types of Robotic Systems

- **Industrial Robots:** Automated machines used in manufacturing (e.g., robotic arms in assembly lines).
- **Mobile Robots:** Robots that can move through their environment (e.g., autonomous vacuum cleaners, delivery robots).
- **Humanoid Robots:** Robots that mimic human appearance and behaviors.
- **Service Robots:** Assist humans in tasks like healthcare, hospitality, or security.
- **Research Robots:** Used for scientific exploration and development.







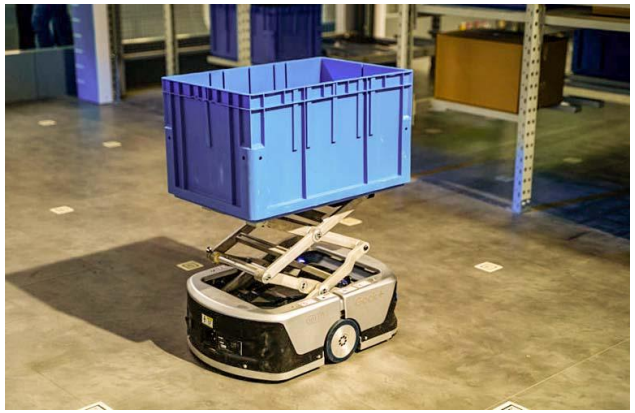




https://www.youtube.com/watch?v=HOoRnv3lA0k&list=RDHOoRnv3lA0k&start_radio=1

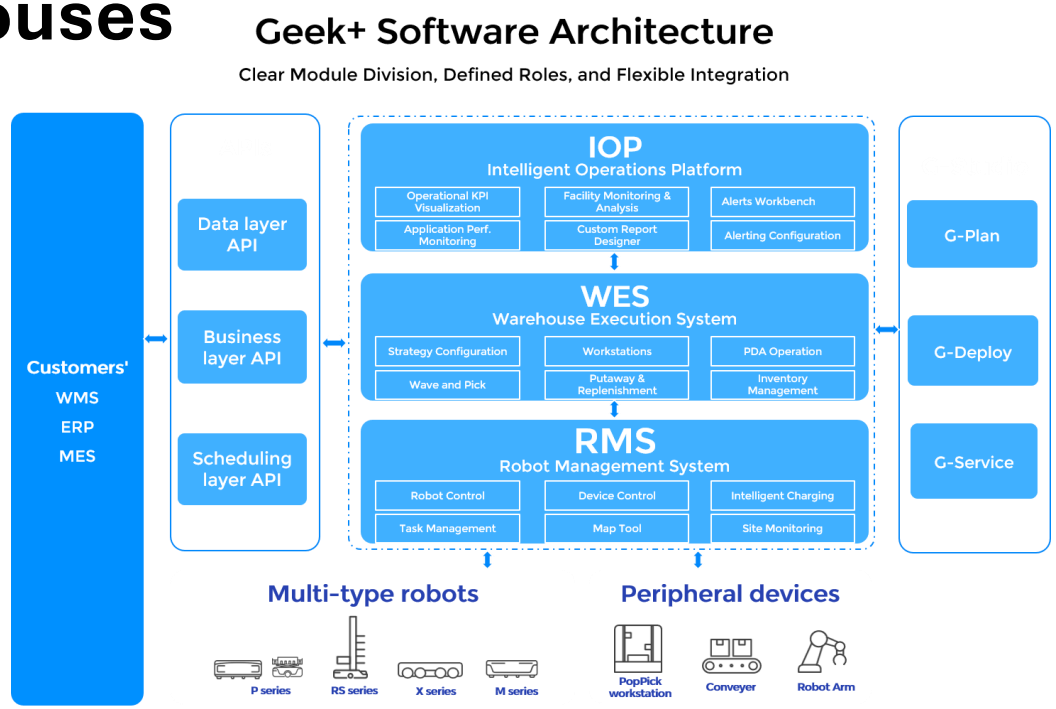
Examples of Real-world Robotic Systems Applications

Intelligent robotics for smart warehouses









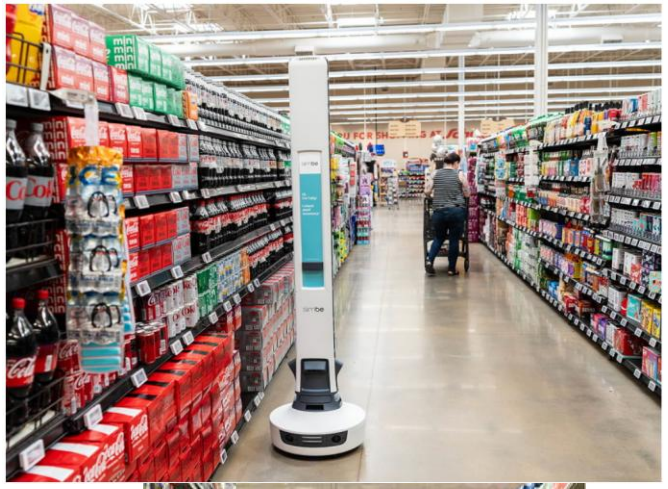
https://www.youtube.com/watch?v=1h64BANqC_g

https://www.youtube.com/watch?v=U2AGLeJBFNg

Examples of Real-world Robotic Systems Applications

Simbe robotics (autonomous mobile robot) with computer vision for retail

- **Low-stock on-shelf detection**: Combines volumetric detections and advanced depth sensing to identify products running low before they become out-of-stock, enabling proactive replenishment and preventing lost sales. Simbe said its volumetric-based low-on-shelf detection provides 1.4x more early signals about shelf condition compared to traditional methods.
- **Comprehensive shelf intelligence**: Detects product spreads, plugs, misplaced items and missing price tags using advanced AI technology, improving planogram accuracy and reducing inventory distortion.
- **Expanded inventory monitoring**: Adapts to dynamic retail environments by combining mobile and fixed sensors. Simbe's advanced computer vision fuses data from autonomous robots and fixed cameras, eliminating planogram maintenance and capturing real-world shelf conditions. Experience reduced complexity with higher accuracy and insights that evolve with a changing store, no matter how often layout or merchandising strategy shifts.
- **Product recognition & similarity detection**: Employs AI-powered visual search for precise product identification across shelves, distinguishing even nearly identical products (such as flavor or size variants) to ensure correct placement and prevent mis-merchandising.
- **Shelf-to-stock comparison**: Cross-checks on-shelf inventory with backroom stock data to uncover hidden "plugs" that appear fully stocked at a glance, preventing phantom out-of-stocks and improving replenishment accuracy.
- **Automated shelf tag verification**: Utilizes barcode scanning, optical character recognition (OCR) and proprietary machine learning to detect mismatches between price tags and products, eliminating costly errors.





Key Integration Steps

1. **System Design:**
    1. Define the robot’s tasks and requirements.
    2. Select compatible hardware and software components.
2. **Component Selection:**
    1. Choose sensors, actuators, and controllers suited for the application.
3. **Hardware Assembly:**
    1. Mechanical construction and wiring.
4. **Software Development:**
    1. Program control algorithms.
    2. Implement communication protocols.
5. **Testing & Calibration:**
    1. Verify hardware functionality.
    2. Calibrate sensors and actuators.
6. **Deployment & Maintenance:**
    1. Deploy the robot in its operational environment.
    2. Perform regular updates and troubleshooting.

Challenges in Integration

1. Compatibility of Components
   - Issue: Ensuring that hardware (sensors, actuators, controllers) and software (drivers, middleware) are compatible.
   - Challenge: Different manufacturers may use proprietary interfaces or standards, making seamless communication difficult.
   - Impact: May require custom interfaces, adapters, or extensive testing to achieve reliable operation.
2. Interoperability and Standardization
   - Issue: Integrating components from multiple vendors with varying protocols.
   - Challenge: Lack of universal standards can cause communication issues or data incompatibility.
   - Solution: Adoption of common protocols (e.g., ROS, OPC UA) and standards to facilitate smoother integration.

Challenges in Integration

3.Real-Time Data Processing and Control

- Issue: Many robotic applications require processing sensor data and controlling actuators within strict time constraints.
- Challenge: Ensuring low latency, deterministic responses, and synchronization across subsystems.
- Impact: Poor timing can lead to unstable or unsafe robot behavior.

4.System Complexity

- Issue: Modern robots often contain numerous subsystems (perception, planning, control).
- Challenge: Managing the complexity, ensuring correct communication flow, and avoiding conflicts.
- Solution: Modular design, hierarchical control architectures, and thorough testing.

Challenges in Integration

5.Calibration and Alignment

- Issue: Sensors and actuators need to be precisely calibrated for accurate operation.
- Challenge: Achieving and maintaining calibration over time, especially in changing environments.
- Impact: Misalignment can cause errors in perception and control, reducing system accuracy.

6.Safety and Reliability

- Issue: Ensuring the robotic system operates safely around humans and in unpredictable environments.
- Challenge: Detecting faults, handling exceptions, and implementing fail-safes.
- Solution: Robust software, hardware redundancies, and safety standards (e.g., ISO 10218).

Challenges in Integration

7. Scalability and Flexibility
   - Issue: Systems need to adapt to new tasks or expanded functionalities.
   - Challenge: Designing systems that are modular and easily extendable without extensive rework.
   - Impact: Difficulties in scaling can limit future upgrades or system reuse.
8. Environmental Variability
   - Issue: Changes in lighting, temperature, or physical surroundings can affect sensors and system performance.
   - Challenge: Designing robust perception and control algorithms that can handle variability.

Challenges in Integration

9. Cost and Time Constraints
   - Issue: Integration can be resource-intensive, requiring significant time and budget.
   - Challenge: Balancing thorough testing with project deadlines and budgets.
10. Human-Machine Interaction
   - Issue: Ensuring that operators can effectively monitor and control complex robotic systems.
   - Challenge: Designing intuitive interfaces and feedback mechanisms.