دانشگاه سمنان

Semnan University
Faculty of Mechanical Engineering

دانشکده مهندسی مکانیک

دانشکده مهندسی مکانیک

درس رباتیک

ROBOTICS

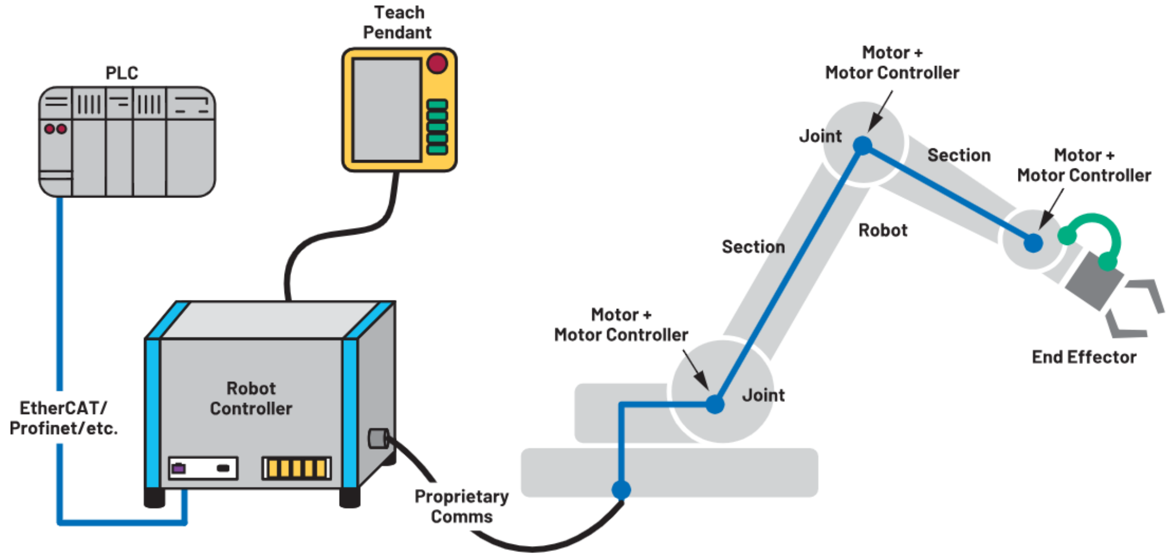
Chapter 6 – Control Architecture
Class Lecture

CONTENTS:

- Chapter 1: Introduction
- Chapter 2: Kinematics
- Chapter 3: Differential Kinematics and Statics
- Chapter 4: Trajectory Planning
- Chapter 5: Actuators and Sensors
- Chapter 6: **Control Architecture**

6. Control Architecture

- A reference functional architecture of an industrial robot control system
- Emphasizing its hierarchical modular structure
  - Clarifies how robot functions are organized and executed
  - Defines programming requirements, such as abstraction levels and interfaces
  - Guides the hardware architecture by specifying control responsibilities at each level

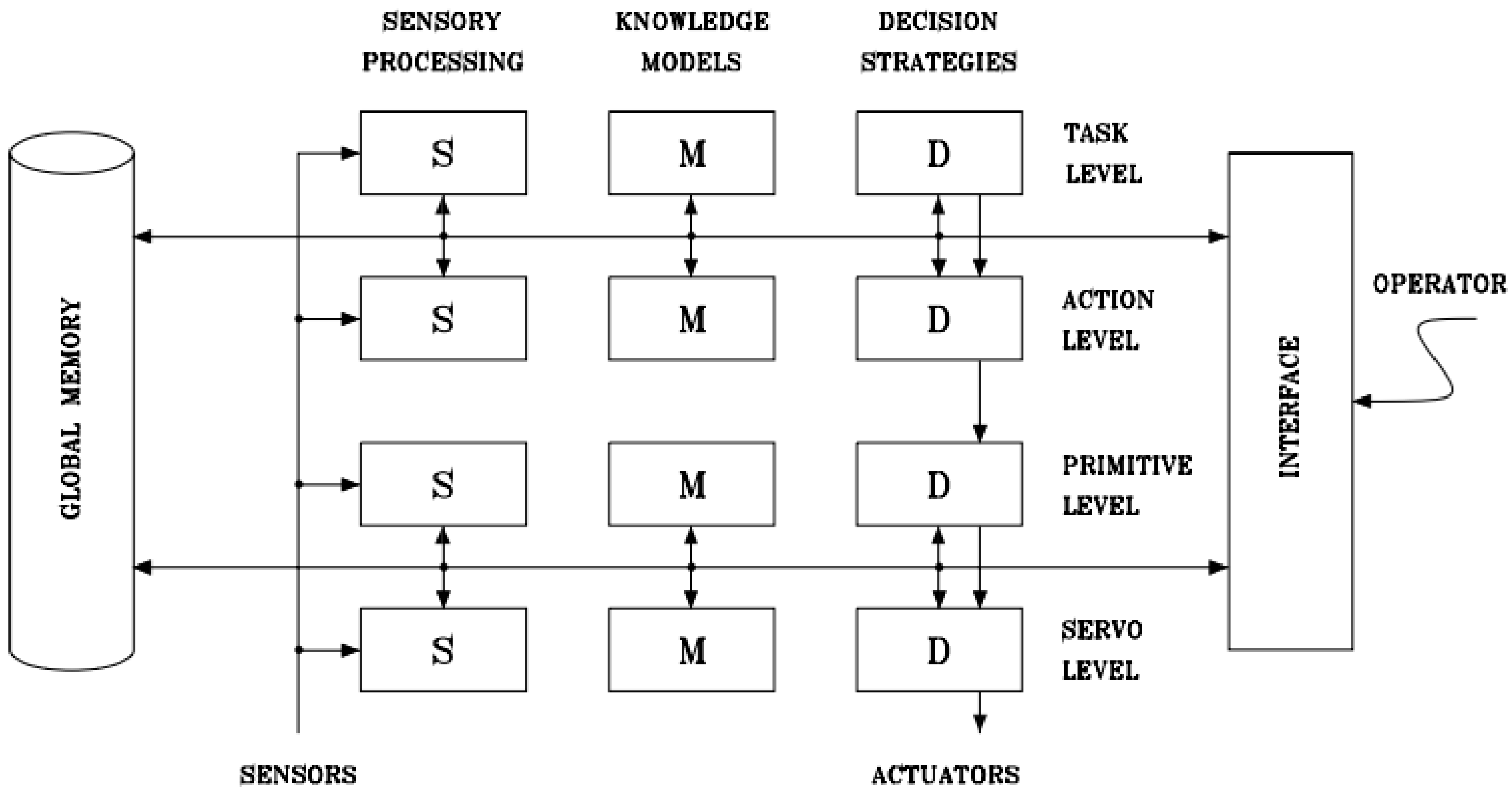


6.1 Functional Architecture

- The control system to supervise the activities of a robotic system should be endowed with a number of tools providing the following functions:
  - Capability of moving physical objects in the working environment, i.e., manipulation ability
  - Capability of obtaining information on the state of the system and working environment, i.e., sensory ability
  - Capability of exploiting information to modify system behaviour in a preprogrammed manner, i.e., intelligence ability
  - Capability of storing, elaborating and providing data on system activity, i.e., data processing ability.

6.1 Functional Architecture

- Reference model for a control system functional architecture



6.1 Functional Architecture

- Task Level – High-Level Planning
  - User-defined goal, specified in a symbolic or abstract form (e.g., "assemble part A to part B")
  - The system analyzes and decomposes this task into elementary actions.
  - Relies on a knowledge base (e.g., tools, parts, procedures) and sensor data (e.g., object locations via cameras or proximity sensors).
  - Outputs: symbolic actions (like "move to part", "pick", "assemble")

6.1 Functional Architecture

- Action Level – Path Planning
  - Translates symbolic actions into specific motion paths or intermediate configurations
  - Decides:
    - ✓ Coordinate system (e.g., joint space vs operational space)
    - ✓ Separation of translation and rotation
    - ✓ Path planning (e.g., via points, interpolation)
  - Checks for feasibility
    - ✓ Collision avoidance, joint limits, singularities, and use of redundancy
  - Uses a geometric/environmental model and receives updates from range or low-level vision sensors.

6.1 Functional Architecture

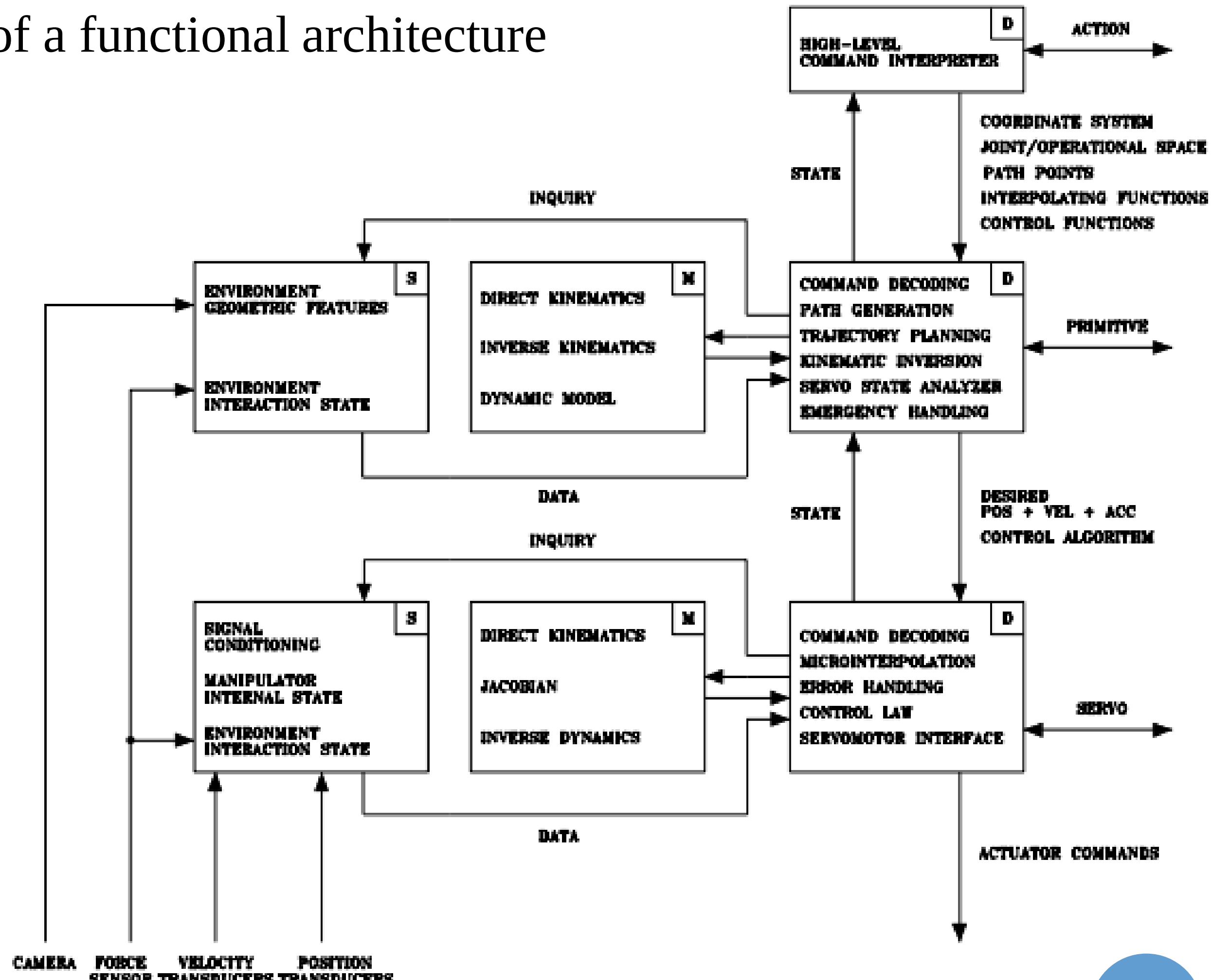
- Primitive Level – Trajectory and Control Preparation
  - Computes detailed motion trajectories from paths provided by the action level
  - Determines:
    - ✓ Trajectory interpolation (for smooth execution)
    - ✓ Control strategy (e.g., centralized, decentralized, impedance control)
    - ✓ Gains and transformations (like inverse kinematics)
  - Uses the dynamic model of the manipulator
  - Sensory feedback (like force sensors) is used to detect and handle conflicts between plan and execution.

6.1 Functional Architecture

- Servo Level – Low-Level Control
  - Executes real-time control laws to drive the robot’s actuators (motors)
  - Performs:
    - ✓ Microinterpolation for precise motion,
    - ✓ Computation of control signals (e.g., voltage, current)
    - ✓ Error correction using sensor feedback (from proprioceptive sensors)
  - Ensures smooth and accurate trajectory following based on the kinematic/dynamic models.

6.1 Functional Architecture

- Hierarchical levels of a functional architecture for industrial robots

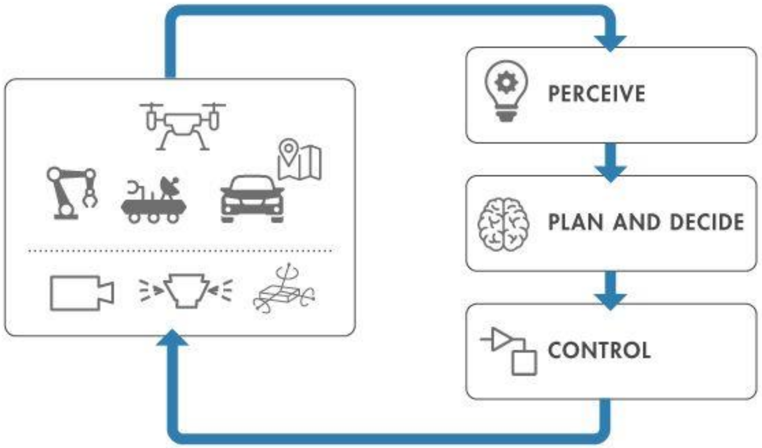


6.2 Programming Environment

- Programming Environment: Provides tools and languages for task definition
- Task Instructions: Operators use the environment to specify robot actions
- Translation Function: Converts high-level commands into executable instructions
- Monitoring Function: Checks and verifies correct task execution
- Unique Challenges: Impacts the physical world, not just virtual outcomes
- Unexpected situations may occur despite accurate models
- Key Difference from Traditional Programming: Must handle real-world unpredictability

6.2 Programming Environment

- A robot programming environment features:
  - Real-time operating system
  - World modelling
  - Motion control
  - Sensory data reading
  - Interaction with physical system
  - Error detection capability
  - Recovery of correct operational functions
  - Specific language structure



6.2 Programming Environment

- 6.2.1 Teaching-by-Showing
  - Basic Concept: Operator manually guides the robot or uses a teach pendant
  - Motion Storage: Joint positions are recorded for later playback
  - No Logic Handling: Lacks capabilities for logic or sequence control
  - Low Technical Barrier: Plant technicians can program without special skills
  - Robot Downtime: Robot must be offline during teaching
  - Common Uses: Spot welding, spray painting, simple palletizing
  - Can be improved by advanced control algorithms

6.2 Programming Environment

- 6.2.2 Robot-oriented Programming
  - Improved by Low-Cost Computing: Enabled development of structured, robot-specific languages.
  - Integration of Functions: Combines high-level language features (e.g., BASIC, PASCAL) with robotics-specific needs.
  - Still Supports Teaching-by-Showing: Retains compatibility with early methods.
  - Requires Skilled Programmers: Operator must be fluent in structured languages.
  - Supports Offline Programming: Programs can be developed without the robot, ideal for structured environments.
  - Enables Complex Applications: Suitable for tasks like assembly in work cells with other machines.
  - Functional Access Level: Operator works at the action level.

6.3 Hardware Architecture

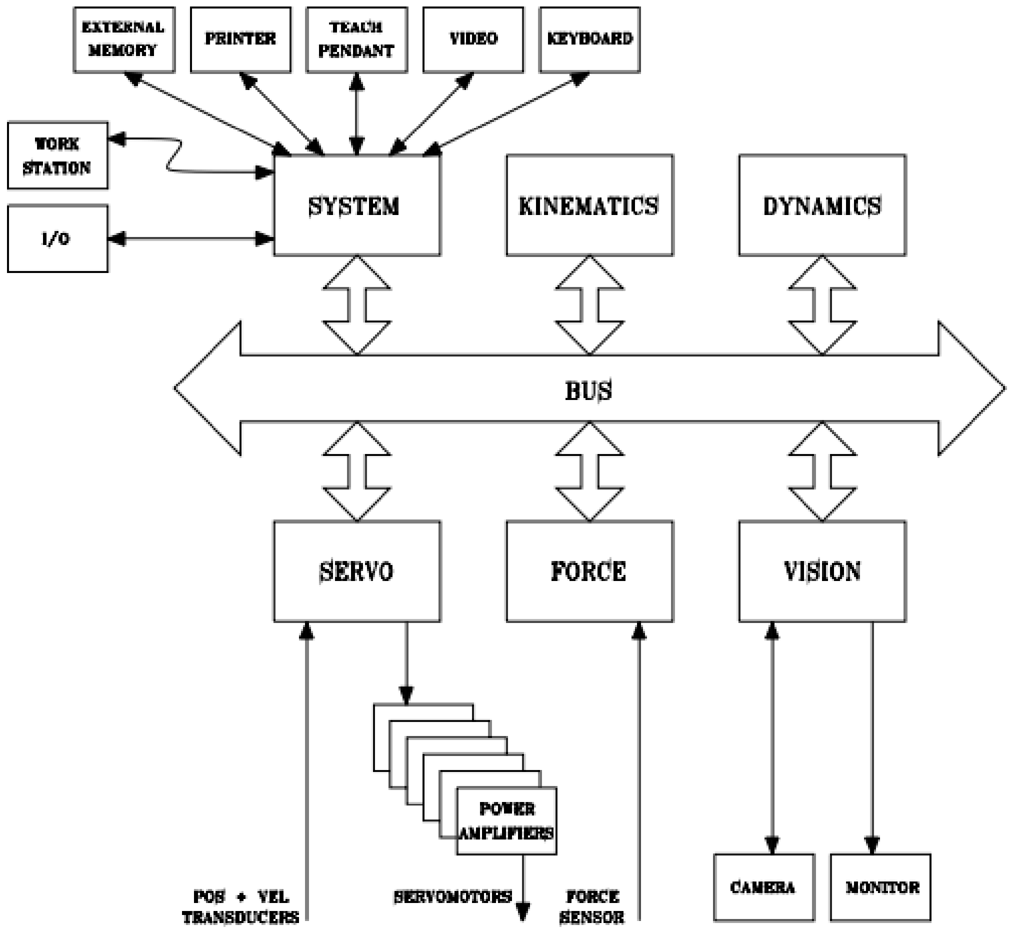
- Hierarchical Functional Structure: Control system follows a layered model
- Distributed Implementation: Functions are mapped to distributed computational boards
- Communication Infrastructure: Boards connected via a high-speed bus system

- Real-Time Performance:
  - Servo and primitive levels require high real-time computing
  - Action level still has limited implementation in most systems

- Bus Bandwidth: Must be sufficient to handle real-time data flow between modules

6.3 Hardware Architecture

- General model of the hardware architecture of an industrial robot's control system



6.3 HARDWARE ARCHITECTURE

- The system board is typically a CPU endowed with:
  - A microprocessor with mathematical coprocessor
  - A bootstrap EPROM (Erasable Programmable Read Only Memory) memory
  - A local RAM memory
  - A RAM memory shared with the other boards through the bus
  - A number of serial and parallel ports interfacing the bus and the external world
  - Counters, registers and timers

6.3 Hardware Architecture

- Additional Processing Power:
  - Boards may include extra processors
  - Purpose: To handle computationally intensive or specialized tasks
  - Architecture: complement the basic system board and integrated via the bus system

- The other boards
  - Kinematics board
  - Dynamics board
  - Servo board
  - Force board
  - Vision board