Inter-Process Communication (IPC), TCP/IP, and Socket Communication

---

1. Inter-Process Communication (IPC)

Definition: IPC is a mechanism that allows **different processes** (separate running programs) to **exchange data** or **coordinate actions**.

Purpose: - Share data between processes. - Synchronize tasks. - Make software modular and scalable.

Common IPC Methods: - **Pipes / Named Pipes:** simple unidirectional data flow. - **Message Queues:** structured message passing. - **Shared Memory:** fastest, share memory directly. - **Semaphores:** synchronization for shared resources. - **Sockets:** communicate between processes, possibly over a network.

In Your Project: - Different hardware modules (Camera, Laser Marker, Printer, PLC) run in separate processes. - IPC ensures these processes can exchange commands, status, and data efficiently.

---

2. TCP/IP Communication

Definition: TCP/IP is a standard protocol for **reliable network communication** between devices or processes.

Why TCP/IP is Used: - Reliable and ordered data transfer. - Error checking and correction. - Standardized communication across different hardware.

Application in Your Project: - Each hardware device has an IP address and port. - Your C++ application opens a TCP socket to communicate. - Commands like "CAPTURE" (camera) or "MARK_TEXT" (laser marker) are sent over TCP. - Responses or status are received reliably.

---

3. Socket Communication

Definition: A socket is an **endpoint** for sending or receiving data over TCP/IP.

Key Functions: - `socket()` → create a socket. - `bind()` → assign IP/port (for server). - `listen()` → server waits for connections. - `accept()` → server accepts client connection. - `connect()` → client connects to server. - `send()` / `recv()` → transfer data. - `close()` → close the socket.

Client-Server Flow: 1. Device (server) listens on IP and port. 2. Your C++ application (client) connects to the device. 3. Send commands and receive data. 4. Each device runs in its own thread for concurrency.

Example in Project: - Camera: trigger image capture → receive image bytes. - Laser Marker: send marking data → confirm marking done. - PLC: read/write control signals.

Why Useful: - Modular: devices can operate independently. - Reliable: TCP ensures commands/data arrive intact. - Concurrent: multithreading allows simultaneous hardware interaction.

---

4. Summary for Interview

> "In our PCB inspection system, different hardware modules like the camera, laser marker, and PLC communicate with our C++ application over TCP/IP sockets. Each module runs independently, and the software uses multithreading to handle concurrent communication. IPC ensures safe and efficient exchange of commands, status, and data, making the system modular, reliable, and scalable."