UART, SPI, I2C Protocols in Embedded Systems

Communication between microcontrollers and peripheral devices is a cornerstone of embedded systems. Three widely used communication protocols for this purpose are **UART**, **SPI**, and **I2C**. Each protocol has distinct characteristics, trade-offs, and use cases.

🔶 2. SPI (Serial Peripheral Interface)

➤ Overview

SPI is a **synchronous**, **full-duplex** protocol typically used for high-speed communication between a master device and one or more slaves.

➤ Characteristics

- **Lines Required**: 4+

- MOSI (Master Out Slave In)

- MISO (Master In Slave Out)
- SCLK (Serial Clock)
- SS/CS (Slave Select)
- **Speed**: Typically up to 10+ Mbps
- **Multi-Device**: ✅ Yes, using separate SS lines or daisy-chaining

➤ Pros

- High-speed communication
- Simple protocol, low overhead
- Full-duplex (can send and receive simultaneously)

➤ Cons

- Requires more GPIOs for multiple devices
- No error checking or acknowledgment
- No formal standard, leading to compatibility issues

🔍 Protocol Comparison Table

Feature	**UART**	**SPI**	**I2C**
Wires Required	2 (TX, RX)	4+ (MOSI, MISO, SCLK, SS)	2 (SDA, SCL)
Speed	Medium	High	Low to Medium
Duplex Mode	Full	Full	Half
Clock Sync	No	Yes (Master)	Yes (Master)
Multiple Devices	❌	✅ (with multiple SS)	✅ (with addressing)
Complexity	Low	Medium	High
Use Case Examples	Debug UART, GPS	Flash memory, LCD	Sensors, RTC, EEPROM

🔧 When to Use What?

- **UART**: Use for **console communication**, **GPS modules**, or where simple one-to-one connection is enough.
- **SPI**: Ideal for **high-speed**, **low-pin-count** communication with **displays**, **flash memory**, or **high-speed ADCs**.
- **I2C**: Best for connecting **multiple low-speed peripherals** like **temperature sensors**, **RTC**, or **EEPROM** on the same bus.