# Synchronous and Asynchronous Communication Protocols

Presented by group 5

# Content



# Communication protocols

01. ***Synchronous Communication:*** Data is sent with a shared clock signal between devices. This ensures that all data transfers in sync with the clock, minimizing errors **it is fast and reliable**

02. ***Asynchronous Communication:*** Data is sent without a shared clock signal. Instead, devices agree on specific speeds to communicate effectively.

**Why Serial communication is important** ?
because it minimizes **wiring complexity, reduces power consumption** . Sending data one bit at a time **requires fewer connections,** making it ideal for compact, resource-limited environments

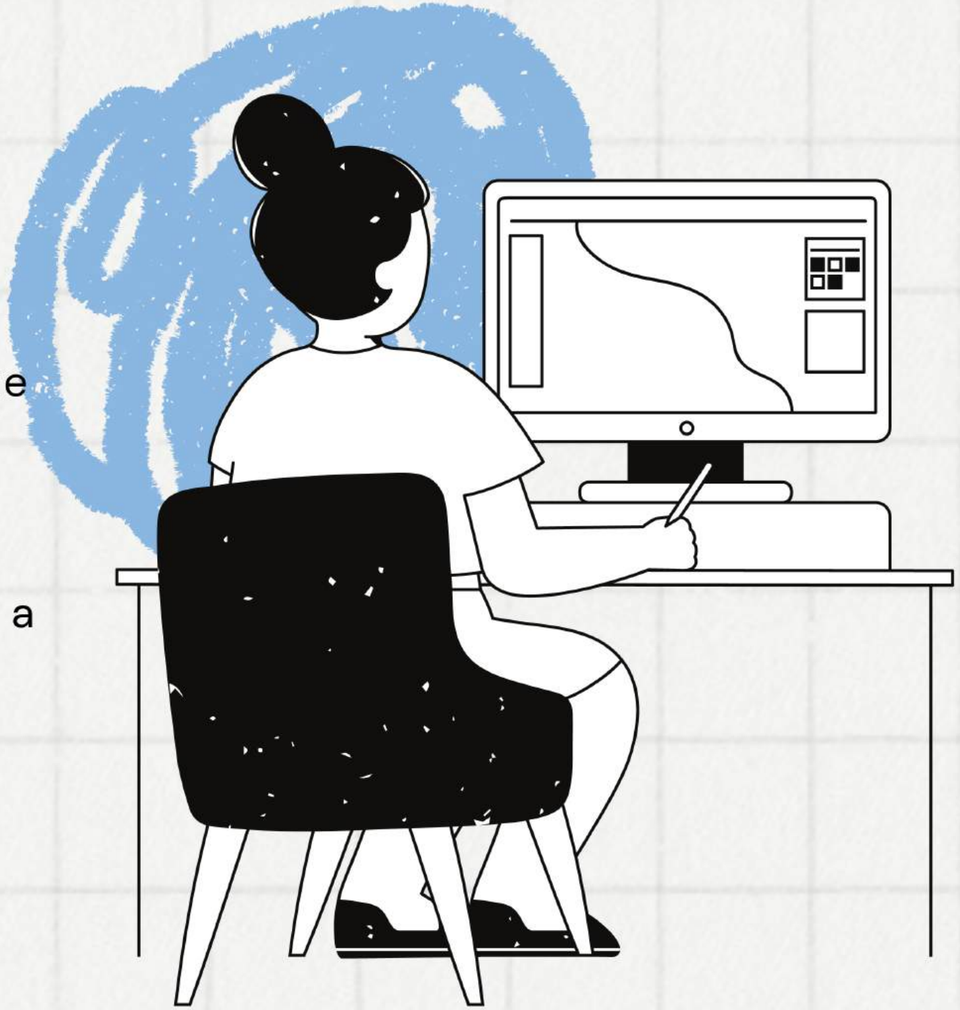
# Overview of Communication Types

**Synchronous Communication** uses a clock line to coordinate data exchange.

- Common for faster data rates.
- Examples: I2C, SPI.

**Asynchronous Communication** doesn't use a clock, usually a pre-determined baud rate ( data transfer rate ) .

- slower communication needs.
- Example: UART (Universal Asynchronous Receiver/Transmitter).



# Serial Communication Protocols

Serial communication refers to sending data one bit at a time over a single channel, which simplifies wiring compared to parallel communication (which sends multiple bits at once over multiple lines) because of the few following reasons .

**1.Simplified wiring**

**2.Cost effectiveness**

**3.Reduce signal interface**

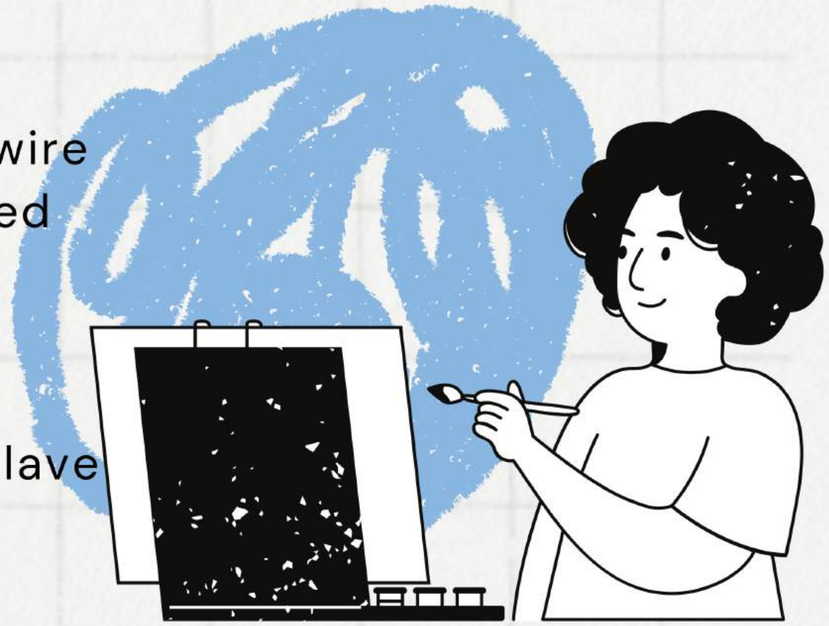
**4.Long distance communication**

# I2C (Inter-Integrated Circuit) Protocol

The I2C protocol, developed by Philips, is a synchronous, two-wire protocol ideal for short-distance communication between multiple devices.

• How I2C Works:
• Master-Slave Topology: One device (master) initiates and controls communication, while others (slaves) respond.
• Two Wires: Data (SDA) and Clock (SCL) lines.
• Data Transmission Process:
• Communication begins with a start condition, and ends with a stop condition.
• Addressing: Each device has an address, allowing the master to specify which device it wants to communicate with.
• Clock Stretching: If a slave device needs more time to process, it can hold the clock line low to delay the master.
• Use Cases:
• Common in embedded systems to connect peripherals like temperature sensors, EEPROMs, and other I/O devices.

# SPI (Serial Peripheral Interface) Protocol



The SPI protocol is another synchronous protocol but uses a four-wire interface for faster data transfers. It's widely used for its high-speed capabilities but requires more lines.

**How SPI Works**:

• Uses SCLK (clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), and SS (Slave Select).

• Data is sent in a continuous stream, allowing full-duplex communication (simultaneous send and receive).

**Data Transmission Process:**

• Data transfer occurs in frames, with configurations for clock polarity (CPOL) and clock phase (CPHA) to adjust how data is sampled.

**Use Cases:**

• SPI is often used for high-speed devices like SD cards, display controllers, and ADCs.

# UART (Universal Asynchronous Receiver/Transmitter) Protocol

UART is an asynchronous protocol that requires no clock signal, making it simple and common for basic serial communication.

**How UART Works:**

• Relies on a baud rate for timing, with start and stop bits to indicate the beginning and end of each data frame.

• It's half-duplex (one-way at a time) or full-duplex depending on the setup.

**Data Transmission Process:**

• Data is sent with a frame structure of start, data bits, optional parity bit, and stop bits.

**Use Cases:**

• Common in serial communication with devices like GPS and Bluetooth modules.

# Comparison of I2C, SPI, and UART

This section compares the protocols, focusing on their data speed, complexity, use cases, and wiring requirements. For example:

- Data Speed: SPI generally has the fastest data rates, followed by I2C, with UART being slower.
- Complexity/Wiring: SPI uses four wires (complex but fast), I2C uses two wires (simple and effective), and UART uses two wires but needs precise timing (simplest for basic needs).
- Application Suitability: SPI is ideal for high-speed devices, I2C is good for low-speed sensors, and UART is commonly used for device-to-device communication over serial connections.

# Conclusion

Summarize the key points and emphasize how I2C, SPI, and UART play significant roles in embedded systems, each with unique strengths and use cases. Additionally, mention the potential for future developments in communication protocols to support IoT and automation technologies.

# Thank you very much!