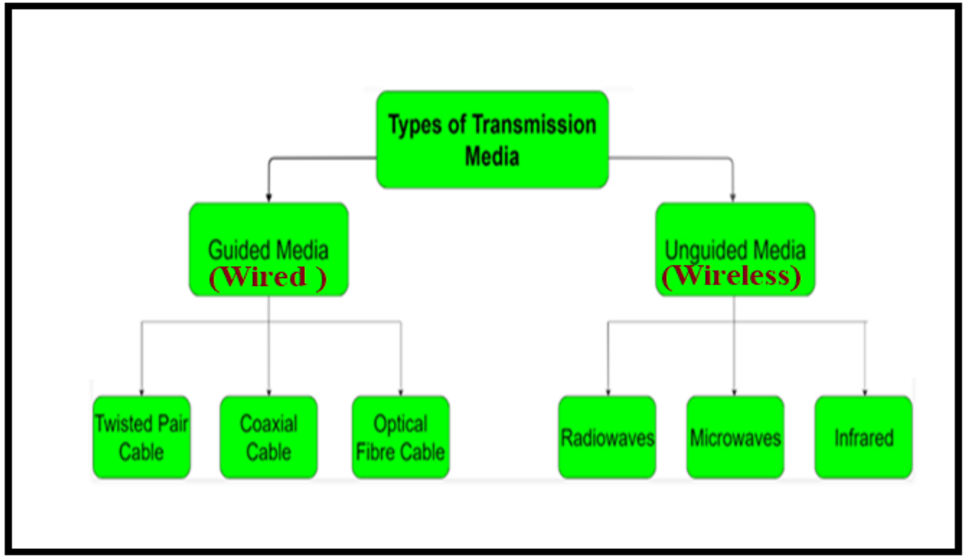
Chapter - 5

Connectivity Technology in Instrumentation System

Prepared By :

Asst. Prof. Sanjivan Satyal

Wired and Wireless Communications System



❑A transmission medium can be broadly defined as anything that can **carry information** from a **source** to **destination**

❑Transmission medium is usually **free space, metallic cable, or fibber-optic cable**

❑Information :- Signal that is the result of a conversion of data from another form

Guided Media

- Also called **Bounded media/ Wired Media**

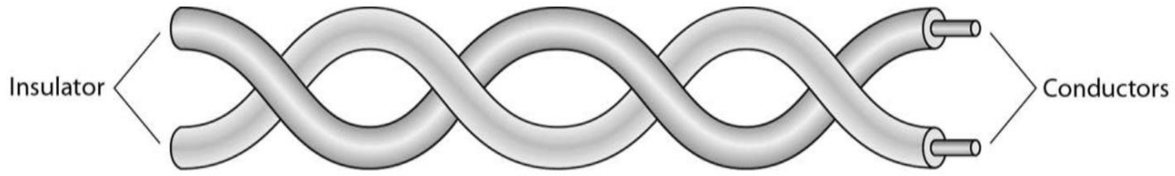
- Guided media, which are those that provide a conduit from one device to another

  1. **Twisted-pair cable**
  2. **Coaxial cable**
  3. **Fibber-optic cable**

- A signal traveling along any of these media is directed and contained by the **physical limits of the medium**

- Twisted-pair and coaxial cable -- Metallic (copper) conductors -- **signals in the form of electric current**

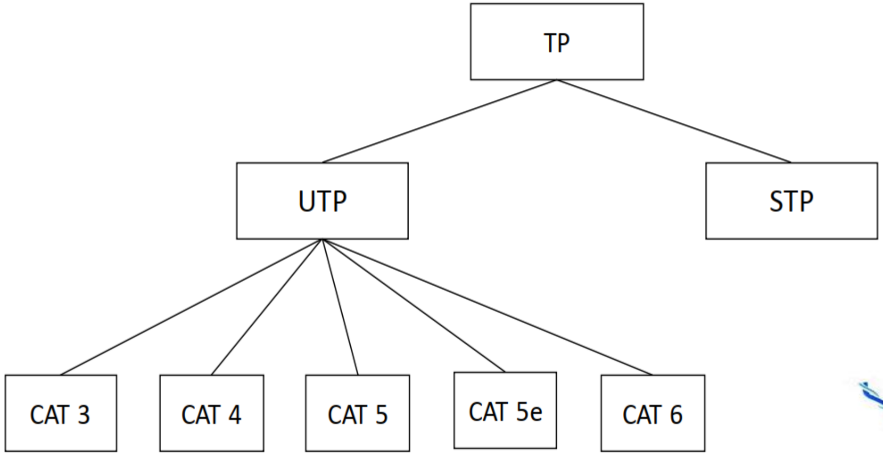
1. Twisted Pair

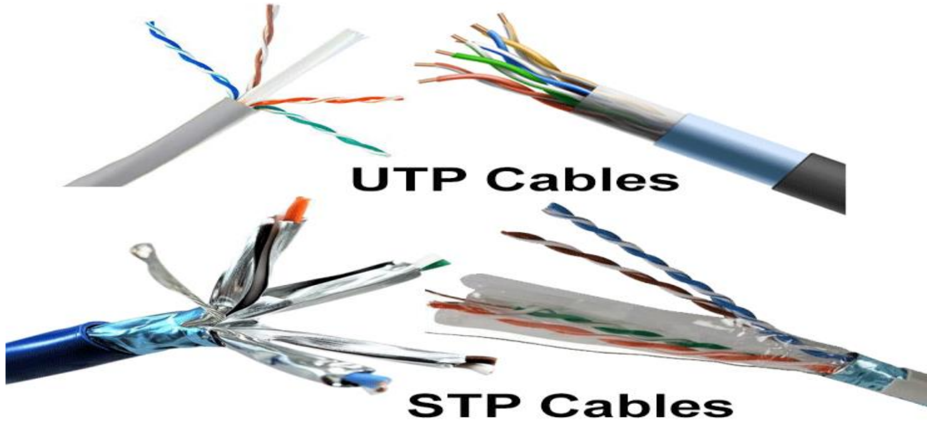


The wires are twisted together in a helical form.

- A twisted pair consists of two insulated copper wires twisted together regular spiral pattern
- Twisting tends to decrease the crosstalk
- Crosstalk is the interference due to the magnetic filed of 2 wires nearby
- Used to transmit both analog and digital transmission

- Twisted pair is limited in distance, bandwidth, and data rate.
- The attenuation for twisted pair is a very strong function of frequency

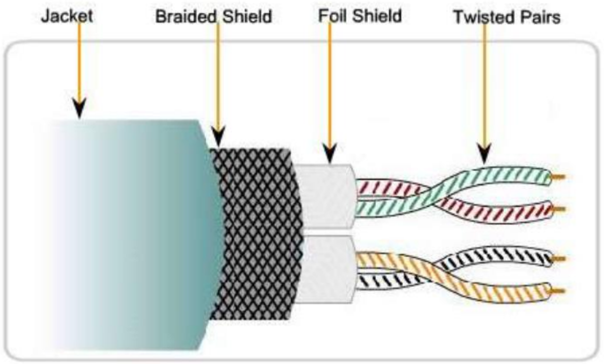




Shielded Twisted Pair(STP)

- STP uses two or more pairs of wires that are wrapped in an overall metallic braid or foil.
- Shields the entire bundle of wires within the cable as well as the individual wire pairs
- Provides better noise protection
- Higher price

Figure STP Cable



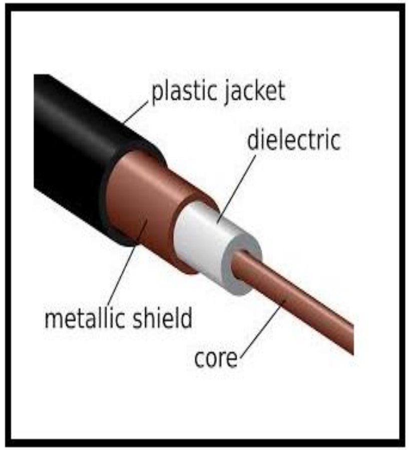
- More expensive
- Easiest to install
- Harder to handle (thick, heavy)
- Data Rate : 10- 100 Mbps
- Max Cable Length – 100M

Unshielded Twisted Pair(UTP)

- Flexible and cheap cable.
- Category rating based on number of twists per inch and the material used
- CAT 3, CAT 4, CAT 5, Enhanced CAT 5 and now CAT 6.

2. Coaxial Cables

- ❑ The coaxial cables have a central copper conductor, surrounded by an insulating layer, a conducting shield, and the outermost plastic sheath.
- ❑ Thus, there are three insulation layers for the inner copper cable.
- ❑ There are two basic modes of data transmission in coaxial cables: baseband mode that has dedicated bandwidth, and broadband mode that has distributed cable bandwidth.

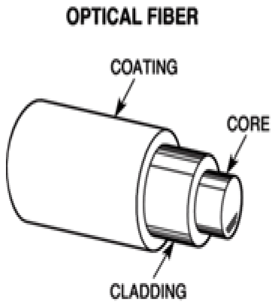


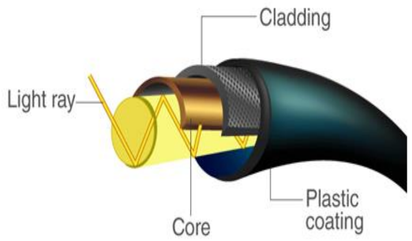
Characteristics

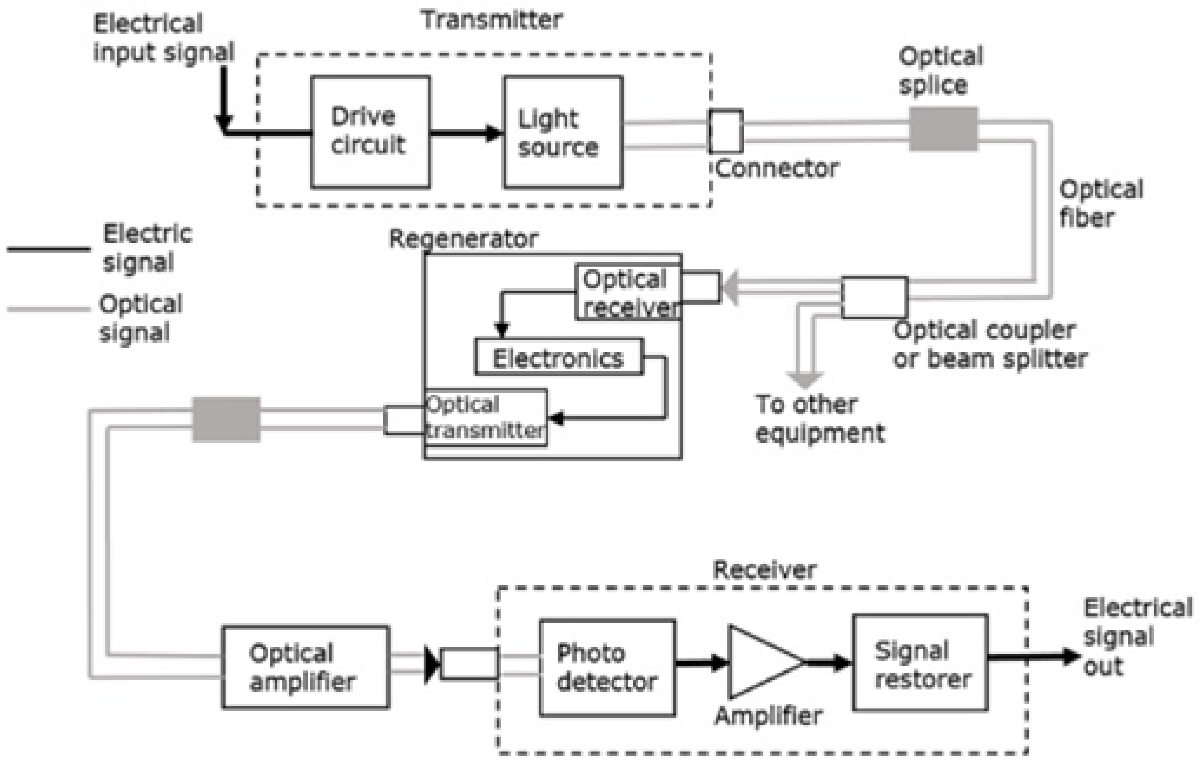
- It is comparatively inexpensive
- Its installation us comparatively simple
- It must be grounded properly in a network connection
- Its bandwidth capacity is around 10 Mbps
- It suffers from data attenuation

3. Optical Fibers

- Optical fibers use light waves for transmission. Crosstalk, EMI, and attenuation aren't issues with optical fibers.







Block Diagram of Optical Fibre Communication

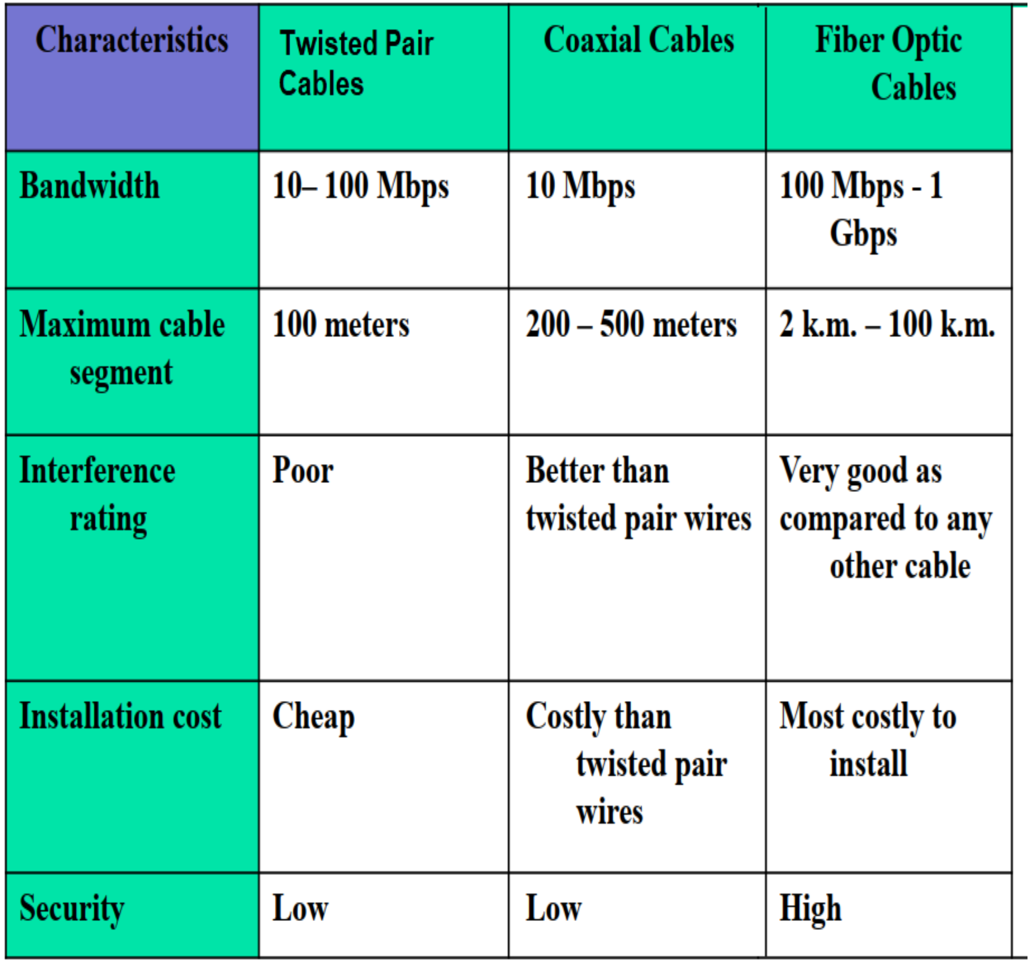
Advantages of Optical Fiber

1. The transmission **bandwidth of the fiber optic cables is higher** than the metal cables
2. The **power loss is very low** and hence helpful in long-distance transmissions.
3. Fiber optic cables **provide high security** and cannot be tapped.
4. Fiber optic cables are **immune to electromagnetic interference**.
5. These are not affected by electrical noise
6. The capacity of these cables is much higher than copper wire cables.
7. Though the capacity is higher, the size of the cable doesn't increase like it does in copper wire cabling system.
8. **The space occupied by these** cables is much less.
9. The **weight of these FOC cables** is much lighter than the copper ones.
10. Since these cables are di-electric, **no spark hazards are present**.
11. These cables are more corrosion resistant than copper cables, as they are bent easily and are flexible.
12. The raw material for the manufacture of fiber optic cables is glass, which is cheaper than copper Fiber optic cables last longer than copper cables

Applications of Fiber Optics

The optical fibers have many applications. Some of them are as follows –

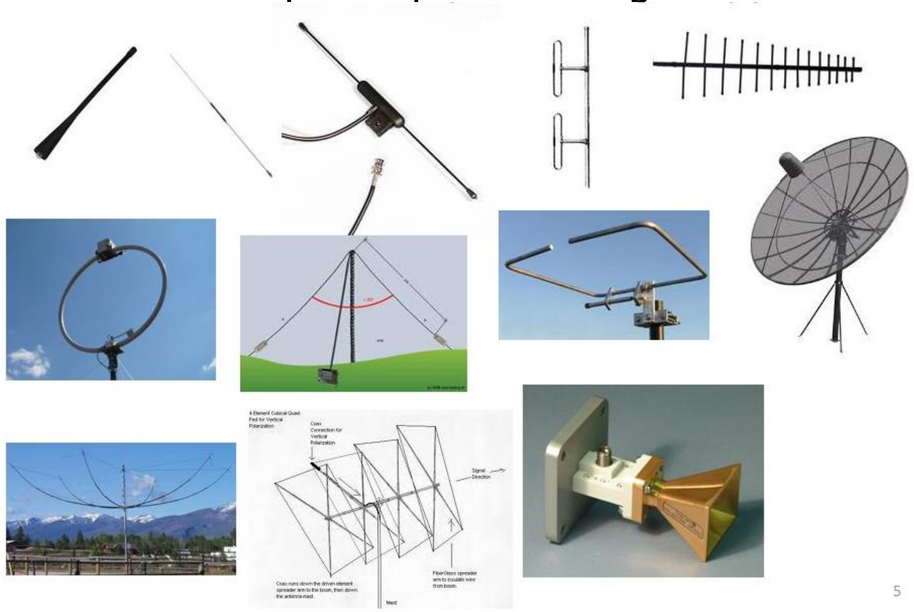
- Used in telephone systems
- Used in sub-marine cable networks
- Used in data link for computer networks, CATV Systems
- Used in CCTV surveillance cameras
- Used for connecting fire, police, and other emergency services.
- Used in hospitals, schools, and traffic management systems.
- They have many industrial uses and also used for in heavy duty constructions.



Characteristics	Twisted Pair Cables	Coaxial Cables	Fiber Optic Cables
Bandwidth	10– 100 Mbps	10 Mbps	100 Mbps - 1 Gbps
Maximum cable segment	100 meters	200 – 500 meters	2 k.m. – 100 k.m.
Interference rating	Poor	Better than twisted pair wires	Very good as compared to any other cable
Installation cost	Cheap	Costly than twisted pair wires	Most costly to install
Security	Low	Low	High

Unbounded Media(Wireless Media)

- Very useful in difficult terrain where cable laying is not possible
- Provides mobility to communication nodes
- Right of way and cable laying costs can be reduced
- Antenna radiates electromagnetic energy into the medium(air)
- Antenna picks up electromagnetic waves from the surrounding medium



An antenna is a structure that is generally a metallic object, often a wire or group of wires, used to convert high- frequency current into electromagnetic waves, and vice versa.

Antennas are essentials while wirelessly signals are transmitting.

Disadvantages

- Susceptible to rain, atmospheric variations
- Objects in transmission path will reduce the signal strength

Advantages

- ❑ Greater Bandwidth
- ❑ Low Power Loss
- ❑ Less Interference
- ❑ Scalable Size
- ❑ Safety
- ❑ Security
- ❑ Flexibility

Frequency Bands

Band	Range	Propagation	Application
VLF	3–30 KHz	Ground	Long-range radio navigation
LF	30–300 KHz	Ground	Radio beacons and navigational locators
MF	300 KHz–3 MHz	Sky	AM radio
HF	3–30 MHz	Sky	Citizens band (CB), ship/aircraft communication
VHF	30–300 MHz	Sky and line-of-sight	VHF TV, FM radio
UHF	300 MHz–3 GHz	Line-of-sight	UHF TV, cellular phones, paging, satellite
SHF	3–30 GHz	Line-of-sight	Satellite communication
EHF	30–300 GHz	Line-of-sight	Long-range radio navigation

Wireless Technology

- Radio Waves 3 KHz to 1 GHz
- Micro Waves

1. Radio Waves 3 KHz to 1 GHz

Radio waves are normally Omni directional.

- When an antenna transmits radio waves, they are propagated in all directions

- Sending and receiving antennas do not have to be aligned.

- The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers.

- Our AM and FM radio stations, cordless phones and televisions are examples of multicasting.

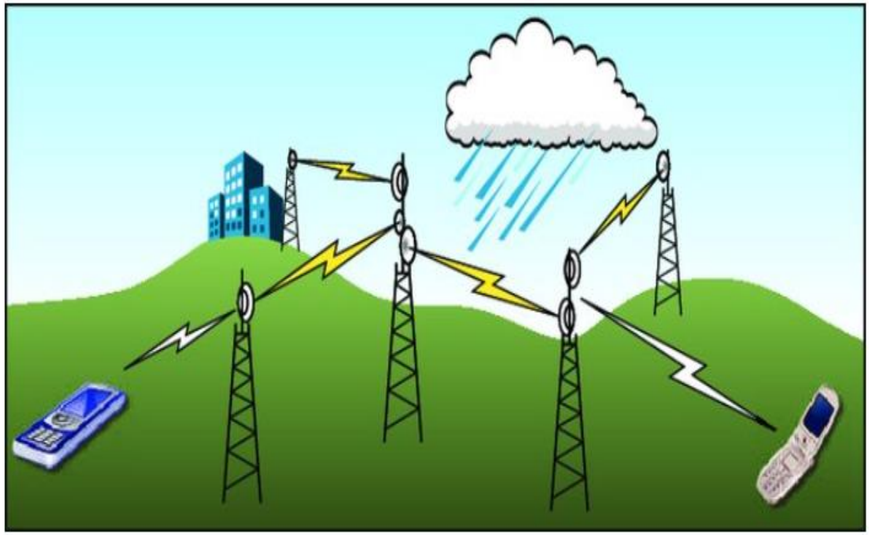
- Bluetooth ,Wi-Fi, GSM, CDMA

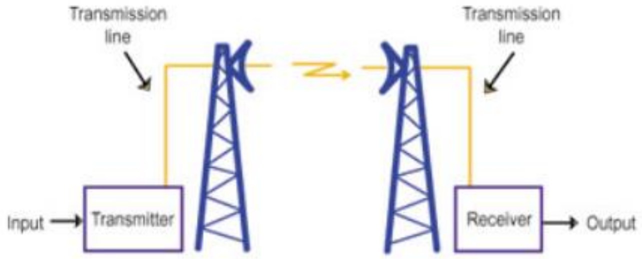
2. Infra Red 300 GHz to 400 THz

- Infrared signals have frequencies between 300 GHz to 400 THz.
- They are used for short-range communication.
- Infrared signals have high frequencies and cannot penetrate walls.
- Line of Sight is needed.
- Infrared is used in devices such as the mouse, wireless keyboard and printers.
- Due to its short-range communication system, the use of an infrared communication system in ONE ROOM will not be affected by the use of another system in the next room

3. Micro Waves (1 GHz to 300 GHz)

- Electronic waves with frequencies between 1 GHz to 300 GHz are normally called microwaves.
- Microwaves are unidirectional, in which the sending and receiving antennas need to be aligned.
- Microwaves propagation is line-of-sight therefore towers with mounted antennas need to be in direct sight of each other
- Due to the unidirectional property of microwaves, a pair of antennas can be placed aligned together without interfering with another pair of antennas using the same frequency.
- High-frequency microwaves cannot penetrate walls. This is why receiving antennas cannot be placed inside buildings.




Transmission line
Transmission line
Input
Transmitter
Receiver
Output

Feature	Wired Networks	Wireless Networks
Speed	Generally faster, offering higher data transfer speeds.	Often slower compared to wired, speed can fluctuate.
Reliability	More reliable, less prone to interference.	Susceptible to interference and signal degradation.
Latency	Lower latency, better for real-time applications.	Higher latency, which can affect performance in real-time applications.
Security	More secure, as physical access to the network is required for unauthorised entry.	More vulnerable to unauthorised access and hacking.
Installation & Setup	Can be labor-intensive and more expensive; involves running physical cables.	Easier and less expensive to set up; no need for physical cables.
Flexibility & Mobility	Limited; devices need to be physically connected to the network.	High; allows for mobility and easy reconfiguration.
Range	Limited by the length of the cables.	Limited by signal range; affected by physical obstructions.
Bandwidth Sharing	Each connection has its own bandwidth.	Shared bandwidth; performance can decrease as more devices connect.
Ideal Use Case	Suitable for environments where speed and security are critical.	Ideal for areas where flexibility and ease of use are prioritised.

Synchronous Transmission

- In serial synchronous data transmission, data is transmitted or received based on a clock signal.
- there must be synchronization between the transmitter and receiver to interpret data correctly
- Usually one or more SYNC characters are used to indicate the start of each synchronous data stream or frame of data.
- Data is sent in blocks at a constant rate.

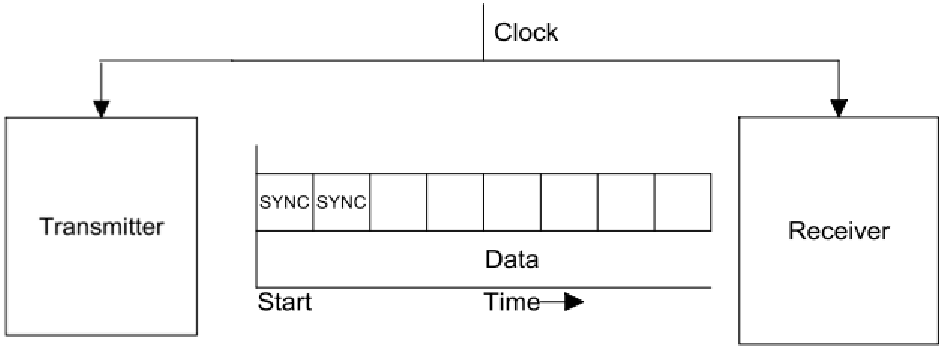


Fig: Synchronous Serial Transmission Format

- Synchronous transmission has the advantage that the timing information is accurately aligned to the received data, allowing operation at much higher data rates
- Data transmission takes place without any gap between two adjacent characters. However data is send block by block.
- a SYNC bit pattern is find between any two blocks of data and hence the data transmission is synchronized
- Synchronous communication is used generally when two computers are communicating to each other at a high speed or a buffered terminal is communicating to the computer.

Asynchronous Transmission

- The receiving device does not need to be synchronized with the transmitting device.
- The transmitting device can send one or more data units when it is ready to send data.
- Each data unit must be formatted i.e. must contain start and stop bits for indicating beginning and the end of data unit. It also includes one parity bit to identify odd or even parity of data.
- To send ASCII character, the framing of data should contain:
  - 1 start bit: Beginning of data
  - 8 bit character: Actual data transferred
  - 1 Parity bit (Optional) for error detection
  - 1 or 2 stop bits: End of data

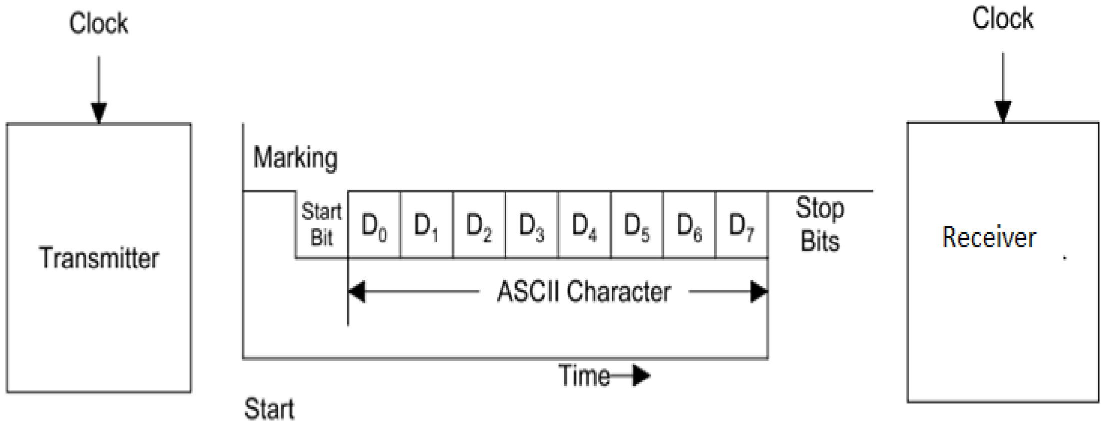


Fig: Asynchronous Serial Transmission Format

- When no data is being sent, the signal line is in a constant high or marking state. The beginning of the data character is indicated by the line going low for 1 bit time and this bit is called a start bit

Wired Connectivity

1. **UART**

- UART, which stands for **Universal Asynchronous Receiver-Transmitter**, is a widely used and fundamental wired connectivity protocol.

- It's a hardware component, often integrated into microcontrollers, that facilitates serial data communication between two devices.

- It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data.
- It's commonly found in embedded systems, microcontrollers (like Arduino, STM32, etc.), computers, and peripheral devices.

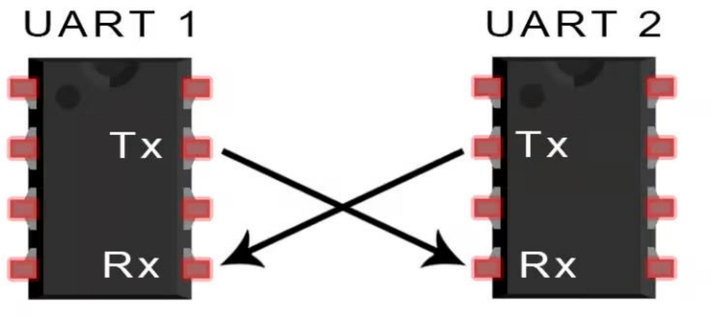
UART Features

- **Asynchronous**: No clock signal is shared between devices. Instead, both must agree on the baud rate (speed).
- **Full-duplex**: Can transmit and receive data simultaneously.
- **Point-to-point**: Typically involves two devices (one transmitter, one receiver).
- **Simple**: Uses only two wires for communication — TX (transmit) and RX (receive).

➢ UART transmitted data is organized into *packets*. Each packet contains 1 start bit, 5 to 9 data bits (depending on the UART), an optional *parity* bit, and 1 or 2 stop bits:

Working on UART

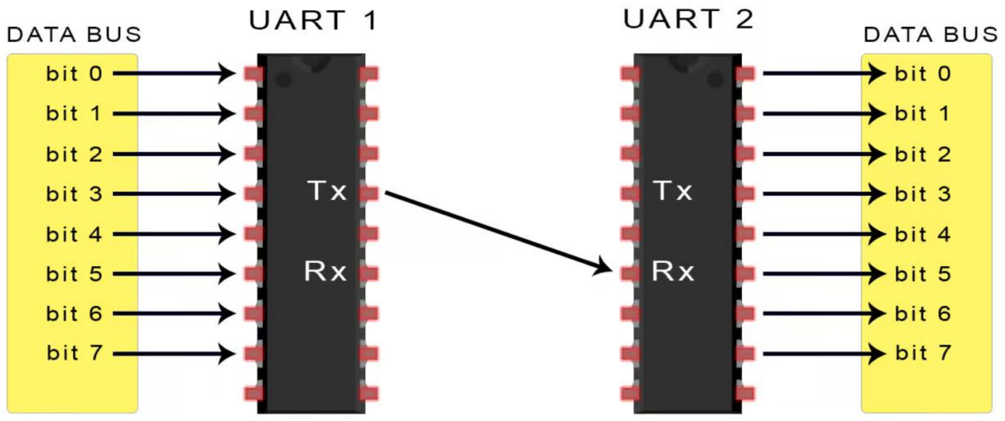
Only two wires are needed to transmit data between two UARTs. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART:



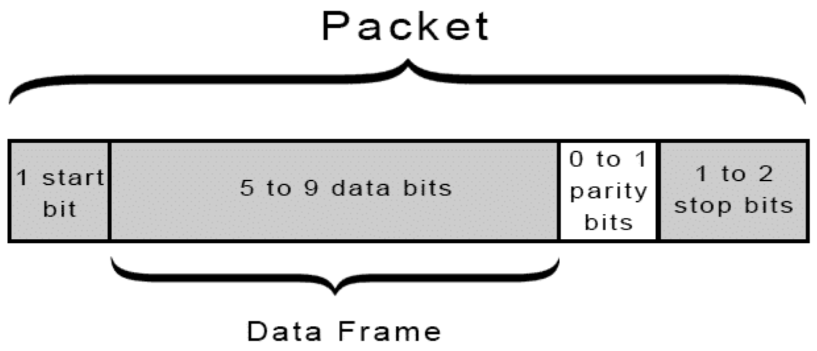
The UART that is going to transmit data receives the data from a data bus. The data bus is used to send data to the UART by another device like a CPU, memory, or microcontroller.

- Data is transferred from the data bus to the transmitting UART in parallel form. After the transmitting UART gets the parallel data from the data bus, it adds a start bit, a parity bit, and a stop bit, creating the data packet.

- Next, the data packet is output serially, bit by bit at the Tx pin. The receiving UART reads the data packet bit by bit at its Rx pin.
- The receiving UART then converts the data back into parallel form and removes the start bit, parity bit, and stop bits. Finally, the receiving UART transfers the data packet in parallel to the data bus on the receiving end:



Frame Structure



- **Start Bit**: 1 bit, always low (0), signals the beginning of transmission.
- **Data Bits**: Usually 8 bits (can be 5 to 9), representing the actual data.
- **Parity Bit (optional)**: Error-checking bit.
- **Stop Bit(s)**: 1 or 2 bits, always high (1), marking the end of a frame.

Parameter	Description
Baud Rate	Speed of transmission (e.g., 9600, 115200 bps)
Data Bits	Number of data bits (5–9)
Parity	None, Even, or Odd
Stop Bits	1 or 2
Flow Control	Optional (e.g., RTS/CTS)

<u>Advantages</u>

- Only uses two wires
- No clock signal is necessary
- Has a parity bit to allow for error checking
- The structure of the data packet can be changed as long as both sides are set up for it
- Well documented and widely used method

<u>Disadvantages</u>

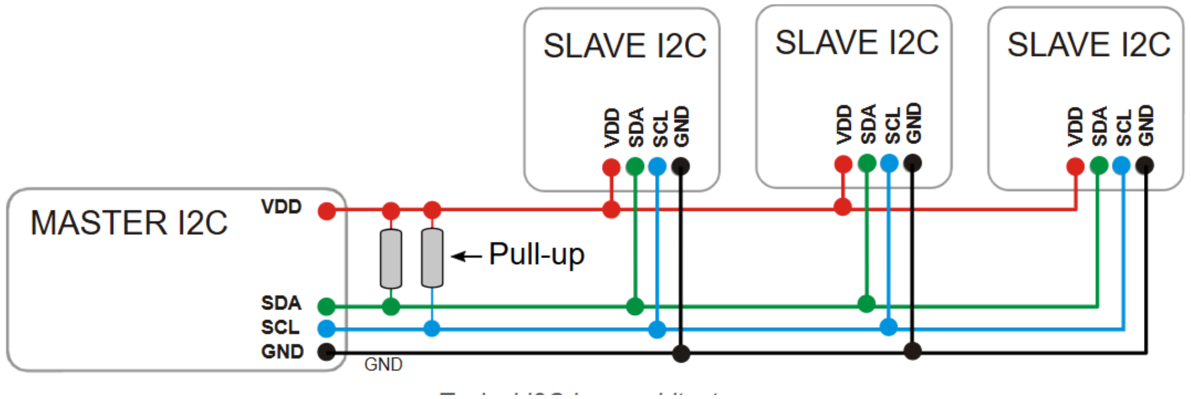
- The size of the data frame is limited to a maximum of 9 bits
- Doesn't support multiple slave or multiple master systems
- The baud rates of each UART must be within 10% of each other

2. Inter-Integrated Circuit (I2C) Communication Protocol

I²C (Inter-Integrated Circuit) is a widely used **synchronous, multi-master, multi-slave** serial communication protocol. It's particularly popular for connecting low-speed peripherals to microcontrollers in embedded systems.

- ❑ This bus has some similarities and differences to the UART communication that was studied earlier
- ❑ Firstly, on each device there is only **one pin that is used to exchange data with other devices, called SDA** (Serial Data Line), while in the UART serial port each device has a TxD pin to transmit data and an RxD pin to receive data.
- ❑ Secondly, there is a pin called **SCL (Serial Clock Line), which is used to establish a common clock signal used to control the timing of the data interchange**. In this way, the I2C bus establishes a synchronous communication (because the clock signal is delivered), and it is not necessary to agree the transmission rate in advance, as in the UART interface, which does not have a clock signal and, therefore, is based on asynchronous communication

- A third point, which is very important to highlight, is that UART connection only allows a point-to point connection between two devices, while an I2C bus allows up to 127 devices to be connected together using only the two connections, SDA and SCL



In order to implement the connection of multiple devices using only two wires, a method to unequivocally identify each device must be implemented. In this way, the manager can send a message to any device, and it can be guaranteed that the destination device will recognize that the message is for it alone.

Pull-up resistors

- Pull-up resistors are essential in the **I2C protocol** because the bus lines (**SDA** and **SCL**) are **"open-drain"** (or "open-collector") by design. Here's a breakdown of **why** they are needed:

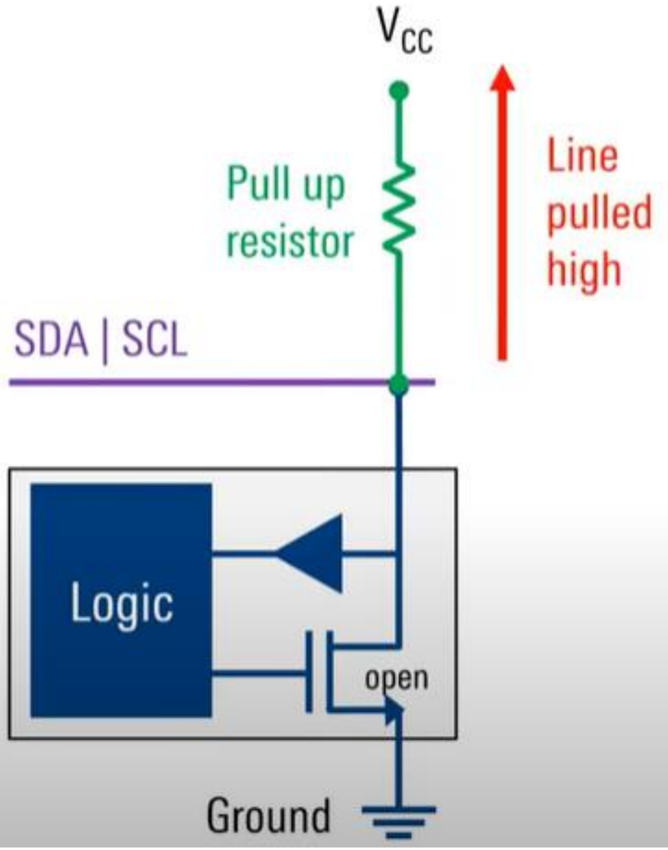
•I2C devices can **only pull the line LOW**.

•They **cannot drive the line HIGH**.

•This allows **multiple devices** to share the bus without causing a short circuit.

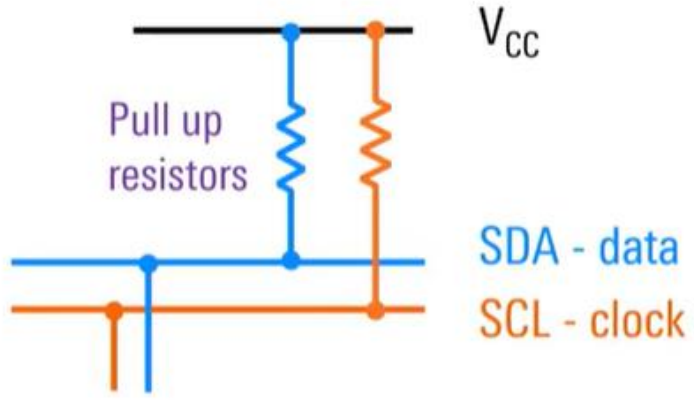
About "open drain"

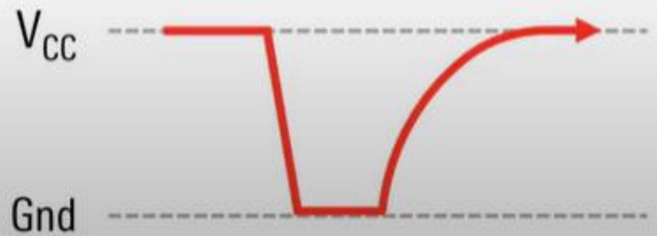
- Each line (SDA and SDL) is connected to voltage (V_{CC} or V_{DD}) via a "pull up" resistor
  - One resistor per line (not per device)
- Each I2C device contains logic that can open and close a drain
- When drain is "closed," the line is pulled low (connected to ground)
- When drain is "open," the line is pulled high (connected to voltage)
- I2C lines are high in the idle state



Pull up resistor values

- Pulling down a line is usually much faster than pulling up a line
  - Pull-up time is a function of bus capacitance and values of pull-up resistors
- Values of pull up resistors are a compromise
  - **Higher resistances** increase the time needed to pull up the line and thus **limit bus speed**
  - **Lower resistances** allow faster communications, but **require higher power**
- Typical pull-up resistor values are in the range of 1 kΩ– 10 kΩ





Pull-up resistors help:

- Achieve clean HIGH-to-LOW transitions.
- Restore the line to HIGH when a device releases it.
- Without them:
- Clock (SCL) and data (SDA) lines wouldn't return to HIGH.
- Communication would **fail**.

Example:

- Let's say the master releases the SDA line after sending a byte:
- If there's **no pull-up resistor**, the SDA line could remain low, or float, and the next device can't send or receive data properly.
- With a pull-up, the line returns to **HIGH**, ready for the next transmission.

Working Principle

- When a device is not transmitting, it does not establish either a low or high state in its SDA and SCL pins. If no device is transmitting, the SDA and SCL lines are in high state, because of the pull-up resistors (Rp1 and Rp2) that can be seen in Figure **a**. Therefore, a manager that wants to transmit data establishes a start bit condition on the bus by setting SDA to a low state when the signal in SCL is high, as can be seen in Figure a. The same figure also shows how the stop bit condition is established.

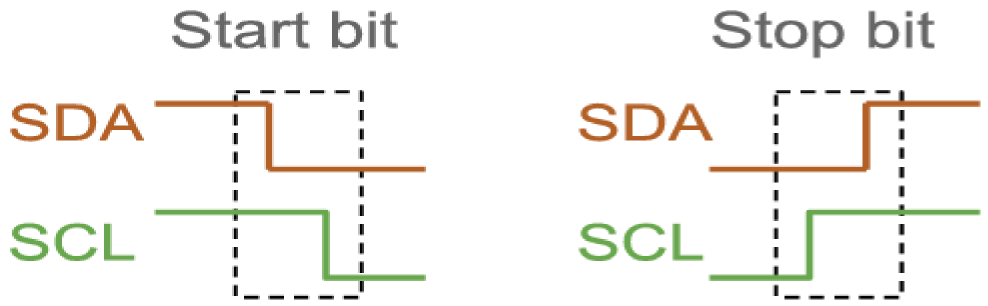


Figure a Example of I2C bus start and stop conditions.

- The first message that the transmitting device sends is the address of the device for which the message is intended. This message is depicted in Figure b.
- The transmitting device establishes, one after the other, the seven bits of the address of the device for which the message is intended (A6 to A0). This is followed by an R/W bit that indicates if the message is a read or a write operation (it is high if it is a read operation and low if it is a write operation).
- The ACK (acknowledge) bit (shown in blue) is established by the destination device to confirm that it has received the message.

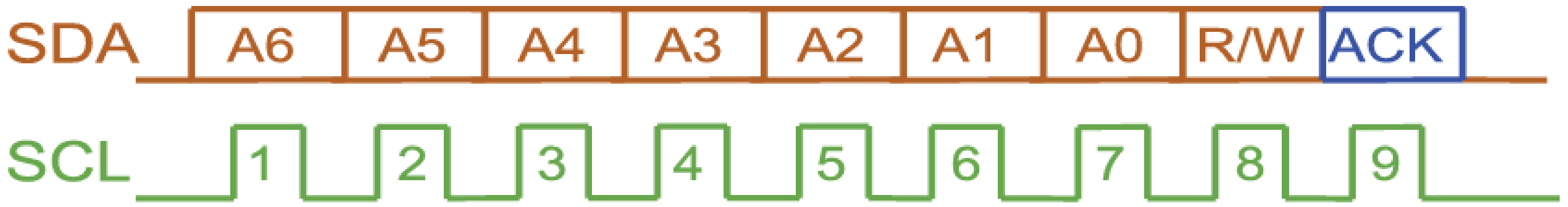
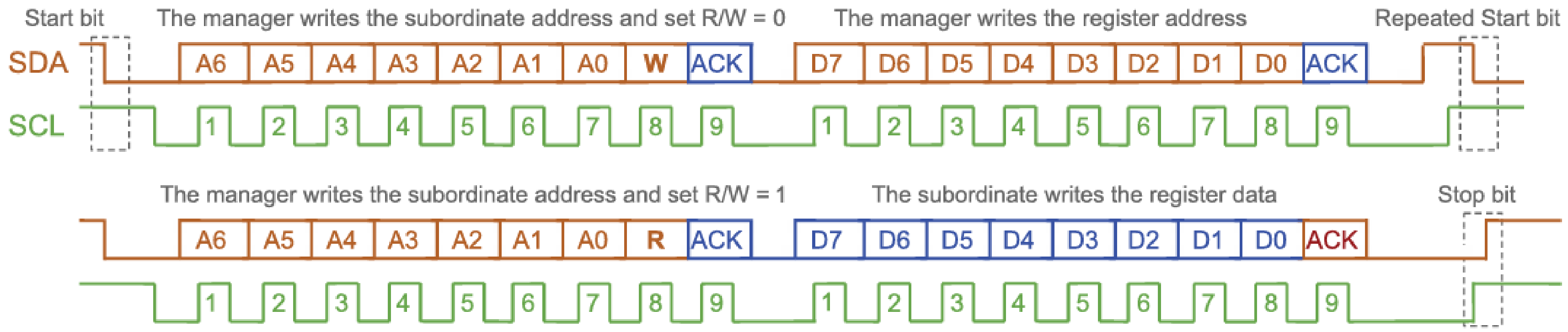


Figure b. Example of a typical I2C bus address message.



Start bit
The manager writes the subordinate address and set R/W = 0
The manager writes the register address
Repeated Start bit
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
1
2
3
4
5
6
7
8
9
The manager writes the subordinate address and set R/W = 1
The subordinate writes the register data
Stop bit
R

Figure c Example of a typical I2C bus communication.

- Figure c shows the typical sequence of an I2C communication. Firstly, the device that starts the communication, called the manager device, establishes the start sequence, followed by the address message corresponding to another I2C device, called the subordinate. Note that the SCL signal is generated by the manager. In Figure c the first R/W bit indicates that a read operation is to be made. The subordinate acknowledges in the ninth clock pulse (shown in blue).
- Then, the manager indicates which register of the subordinate it wants to read using D7 to D0. Note that this is followed by another start bit without any prior stop bit, after which the manager repeats the address of the subordinate, using the R/W bit to indicate that a read operation is being made. After that, the subordinate writes the 8 bits (indicated in blue) corresponding to the register data following the SCL pulses that are established by the manager. Finally, the manager generates the ACK bit (indicated in brown) and the stop bit.

- Note: In a read operation, if the subordinate is not ready to send the data it is allowed to hold the SCL line low. The manager should wait until the SCL line goes high before continuing.
- Note: The I2C bus allows more than one manager on the bus, but only one manager can use the bus at any one time

Advantages of I2C:

- **Minimal Wiring:**
- **Multi-Master Support:**
- **Multi-Slave Support**
- **Built-in Addressing:**
- **Acknowledgement (ACK/NACK):**
- **Clock Stretching**.
- **Widely Supported:**

3. Serial Peripheral interface (SPi) Communication Protocol

Overview of the SPI Bus

- Originally created by Motorola, the SPI bus is a full-duplex serial communication standard that enables simultaneous bidirectional communication between a master device and one or more slave devices.

- Because the SPI protocol does not follow a formal standard, it is common to find SPI devices that operate in a slightly different way; for example, the number of transmitted bits may differ, or the slave select line might be omitted, among other things.

- Commonly Spi use for interfacing with devices that implement the most common SPI interfaces (which are the ones that are supported by the Arduino IDE, and the third-party libraries that you'll use)

- ❑ SPI buses can act in four main ways, which depend on the requirements of your device. SPI devices are often referred to as slave devices.
- ❑ SPI devices are synchronous, meaning that data is transmitted in sync with a serial clock line (SCLK).
- ❑ Data can be shifted into the slave device on either the rising or falling edge of the clock signal (called the clock phase), and the SCLK default state can be set to either high or low (called the clock polarity).
- ❑ Because there are two options for the clock phase and two options for the clock polarity, you can configure the SPI bus in a total of four ways. Table 11-1 shows each of the possibilities and the modes that they correspond to in the Arduino SPI library. When you use a library that is explicitly designed to interface with a particular device, the library will generally operate automatically in the correct mode. Still, understanding the low-level communication hurdles will help you troubleshoot potential issues down the road.

SPI Communication Modes

SPI Mode	Clock Polarity	Clock Phase
Mode 0	Low at Idle	Data Capture on Clock Rising Edge
Mode 1	Low at Idle	Data Capture on Clock Falling Edge
Mode 2	High at Idle	Data Capture on Clock Falling Edge
Mode 3	High at Idle	Data Capture on Clock Rising Edge

SPI Hardware and Communication Design

- The SPI system setup is relatively simple. Three pins are used for communicating bet ween a master and all slave devices:

- Shared/Serial Clock (SCLK)
- Master Out/Slave In (MOSI)
- Master In/Slave Out (MISO)

- Each slave device also requires an additional slave select (SS) pin. Hence, the total number of I/O pins required on the master device will always be 3 + n, where n is the number of slave devices. Following Figure shows an example SPI system with two slave devices.

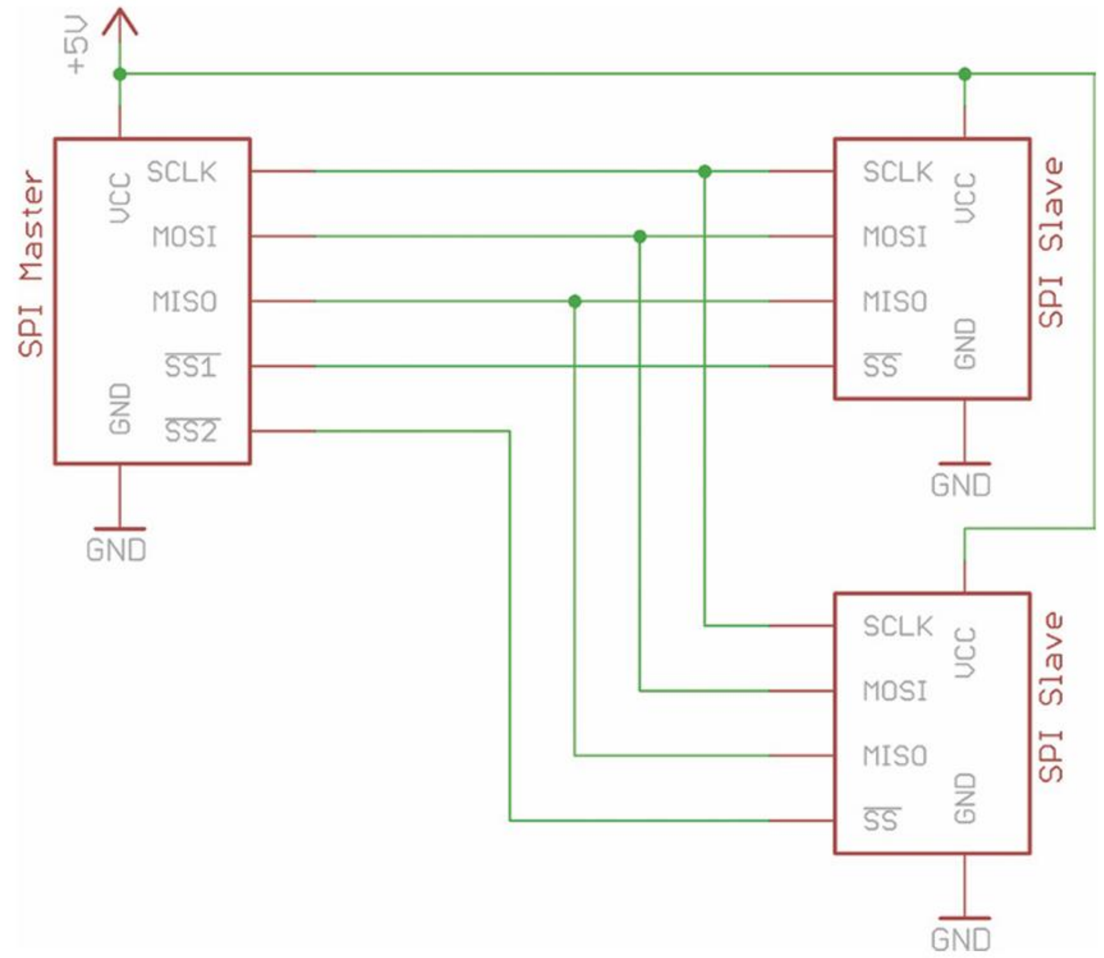


Figure : SPI reference hardware configuration

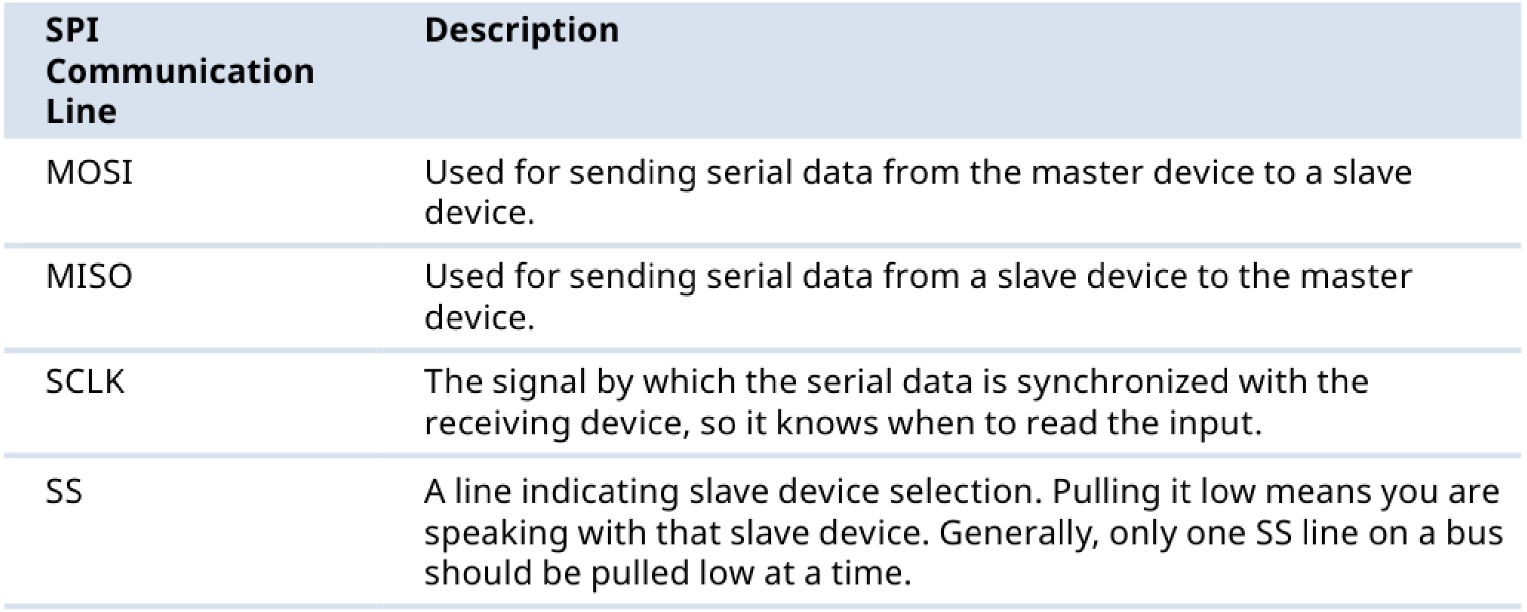
Hardware Configuration

- Four data lines, at a minimum, are present in any SPI system. Additional SS lines are added for each slave device appended to the network. Before you learn how to actually send and receive data to and from an SPI device, you need to understand what these I/O lines do and how they should be wired. Table in next slide describes these lines.

- Unlike with the I2C bus, pull-up resistors are not required (the I/O pins are all push/ pull instead of open-drain), and communication is fully bidirectional.

- To wire an SPI device to the Arduino, all you have to do is connect the MOSI, MISO, SCLK, and SS. Don't forget to be mindful of voltage and logic levels; if you are using a 5V Arduino (like the Uno), then you should ensure that your SPI slave device can also talk at 5V logic levels

SPI Communication Lines



SPI Communication Line	Description
MOSI	Used for sending serial data from the master device to a slave device.
MISO	Used for sending serial data from a slave device to the master device.
SCLK	The signal by which the serial data is synchronized with the receiving device, so it knows when to read the input.
SS	A line indicating slave device selection. Pulling it low means you are speaking with that slave device. Generally, only one SS line on a bus should be pulled low at a time.

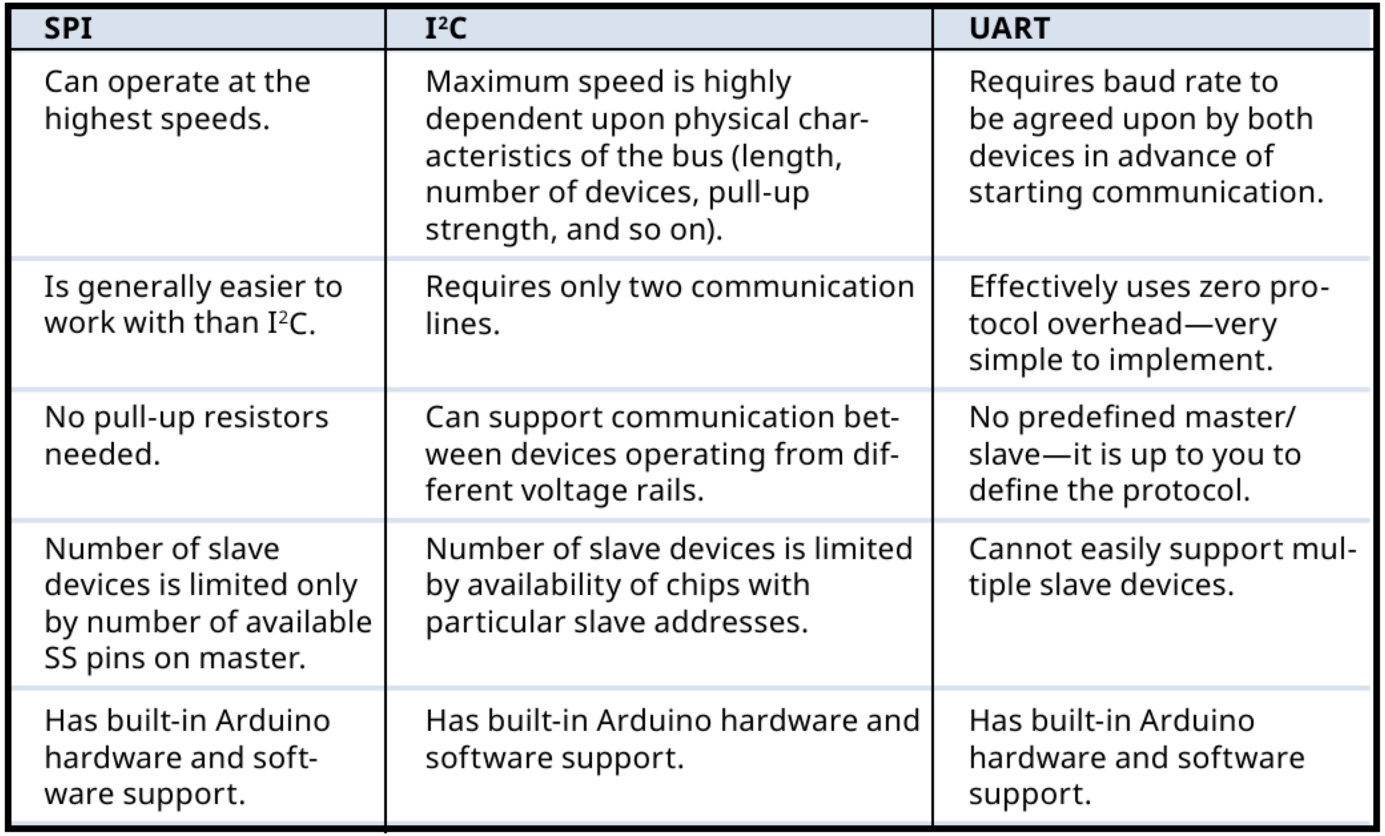
- Note: Because SPI is not a universal standard, some devices and manufacturers may use different names for the SPI communication lines. Slave select is sometimes referred to as chip select (CS); serial clock is sometimes just called clock (CLK or SCK); and MOSI and MISO pins on slave devices are sometimes abbreviated to serial data in (SDI) and serial data out (SDO).

Communication Scheme

- The SPI communication scheme is synced with the clock signal and depends on the state of the SS line. Because all devices on the bus share the MOSI, MISO, and SCLK lines, all commands sent from the master arrive at each slave.
- The SS pin tells the slave whether it should ignore this data or respond to it. Importantly, this means that you must make sure to only have one SS pin set low (the active mode) at a time in any program that you write.
- The basic process for communicating with an SPI device is as follows:

1. Set the SS pin low for the device you want to communicate with.
2. Toggle the clock line up and down at a speed less than or equal to the transmission speed supported by the slave device.
3. For each clock cycle, send 1 bit on the MOSI line, and receive 1 bit on the MISO line.
4. Continue until transmitting or receiving is complete, and stop toggling the clock line.
5. Return the SS pin to high state.

SPI, I2C, and UART Comparison



SPI	I^{2}C	UART
Can operate at the highest speeds.	Maximum speed is highly dependent upon physical characteristics of the bus (length, number of devices, pull-up strength, and so on).	Requires baud rate to be agreed upon by both devices in advance of starting communication.
Is generally easier to work with than I^{2}C.	Requires only two communication lines.	Effectively uses zero protocol overhead—very simple to implement.
No pull-up resistors needed.	Can support communication between devices operating from different voltage rails.	No predefined master/slave—it is up to you to define the protocol.
Number of slave devices is limited only by number of available SS pins on master.	Number of slave devices is limited by availability of chips with particular slave addresses.	Cannot easily support multiple slave devices.
Has built-in Arduino hardware and software support.	Has built-in Arduino hardware and software support.	Has built-in Arduino hardware and software support.

4. CAN (Controlled Area Network) (Assignment)

- Wired connectivity CAN" most likely refers to **Controller Area Network (CAN)**, a robust and widely used wired communication protocol commonly found in automotive and industrial systems.
- **CAN** is a **serial communication** protocol designed for reliable, high-speed communication between microcontrollers and devices **without a host computer**. It was developed by **Bosch** in the 1980s for automotive applications and has since expanded into various industries.

Wired Connectivity via CAN – Key Features

Feature	Description
Medium	Twisted-pair copper wires (usually shielded)
Topology	Bus topology
Speed	Up to 1 Mbps (CAN 2.0); up to 5 Mbps (CAN FD)
Distance	~40m at 1 Mbps, up to 1 km at lower speeds
Reliability	Excellent error detection and fault confinement
Use Case	Vehicles (ECU to ECU), industrial automation, medical equipment, robotics, etc.

How CAN Works

- **Multi-master**: Any node can transmit data when the bus is free.
- **Message-based**: Data is sent in frames; each has an ID that determines priority.
- **Arbitration**: If two devices try to send at the same time, the one with the higher-priority message wins (non-destructive).