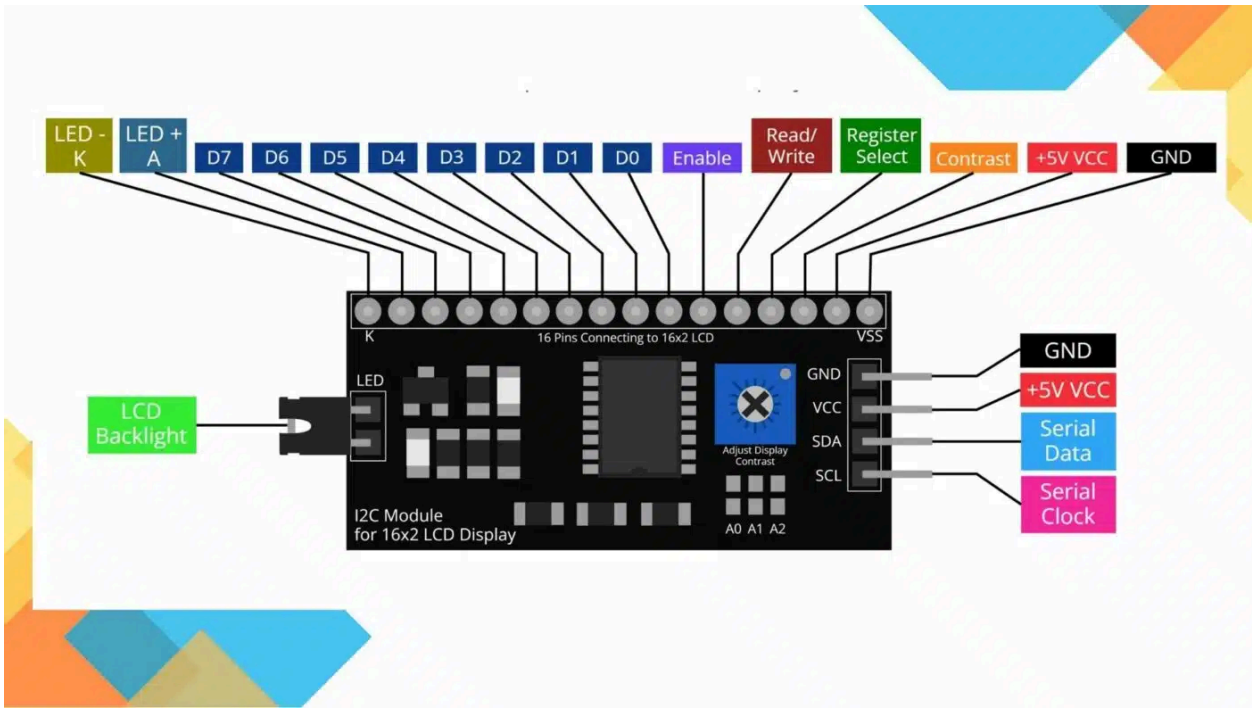
🧠 I2C (Inter-Integrated Circuit) Communication Protocol

I2C stands for **Inter-Integrated Circuit.** It is a bus interface connection protocol incorporated into devices for serial communication. It was originally designed by Philips Semiconductor in 1982. Recently, it is a widely used protocol for short-distance communication. It is also known as Two Wired Interface(TWI).



🔧 Overview

- **I2C** is a **serial** communication protocol designed for **short-distance communication** between ICs

  .

- Uses **two bi-directional open-drain lines**:
  - **SDA (Serial Data Line)** – Transfers data.
  - **SCL (Serial Clock Line)** – Carries clock signals.

- Both lines are **pulled high** using pull-up resistors.
- Supports **multi-master and multi-slave** configuration.

🛠️ Working Principle

- Operates in two modes:
  - **Master Mode** – Initiates communication and generates clock.
  - **Slave Mode** – Responds to master's commands.

- Each bit transfer is synchronized with the **falling edge of SCL**.

- **Rule**: **SDA can only change when SCL is LOW**.
  This ensures valid data capture when SCL is HIGH.

📦 Packet Structure (9 bits)

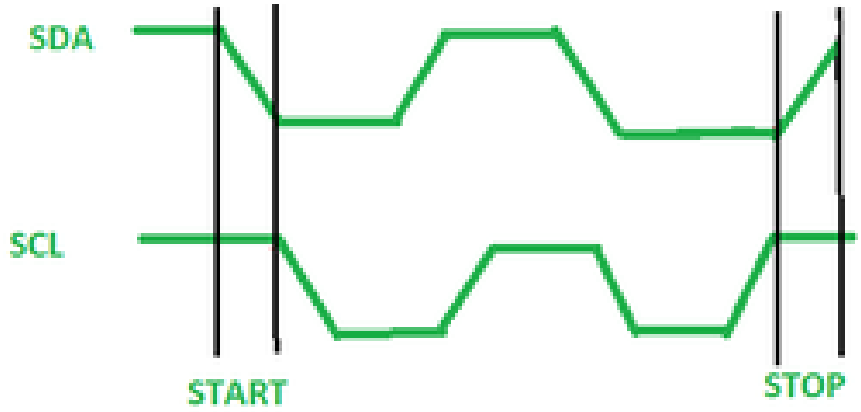
- I2C transmits data in **packets** of **9 bits**:

1. **Start Condition** (initiated by master)
2. **7-bit Slave Address** + **1-bit R/W flag**
3. **1-bit Acknowledge (ACK/NACK)**

Each data transfer byte (8 bits) is followed by 1 ACK/NACK bit.

🔄 Step-by-Step I2C Data Transmission

1. **Start Condition**
    - **SDA goes LOW** while **SCL is HIGH**.
    - Signals start of communication.



2. **Slave Address + R/W Bit**
    - Master sends:
        - **7-bit address** of the slave.

        - **1-bit Read/Write (R/W)**:
            - `0`: Write (Master → Slave)
            - 1: Read (Slave → Master)

3. **ACK/NACK Bit**
    - Slave pulls **SDA LOW** on the next clock pulse if it acknowledges the address.
4. **Data Transmission**
    - 8-bit data is sent by master or slave.
    - After each byte, receiver sends **ACK/NACK**.
5. **Stop Condition**
    - **SDA goes HIGH** while **SCL is HIGH**.
    - Marks the end of transmission.

🌟Benefits of the I2C Module:

The I2C module offers several advantages, including:

a. Simplicity: The I2C module requires minimal wiring and uses just two pins on the Arduino board, freeing up other pins for additional components.

b. Flexibility: The I2C module supports connecting multiple devices on the same bus, allowing for a wide range of configurations.

c. Addressing: Each device connected to the I2C bus has a unique address, enabling the Arduino to communicate with specific devices as needed.

d. Speed: The I2C module operates at varying speeds, accommodating different data transfer rates based on the requirements of the connected devices

Programming I2C Communication:

The Arduino IDE provides a library called “Wire” that simplifies I2C communication. You can include this library in your sketch to utilize the functions and commands needed to interact with I2C devices.

a. Initializing I2C: Start by initializing the Wire library in your code using the `Wire.begin()` function.

b. Reading from I2C Devices: Use the `Wire.requestFrom()` function to request data from an I2C device. Once received, you can read the data using the `Wire.read()` function.

c. Writing to I2C Devices: To send data to an I2C device, use the `Wire.beginTransmission()` function, followed by `Wire.write()` to send the data, and `Wire.endTransmission()` to complete the process.

d. Addressing I2C Devices: Specify the I2C device’s address in the code using the `Wire.beginTransmission()` function before reading or writing to that device.

Reference

- [ ] https://www.geeksforgeeks.org/i2c-communication-protocol/

- [ ] https://www.okuelectronics.com/demystifying-the-i2c-module-for-arduino/