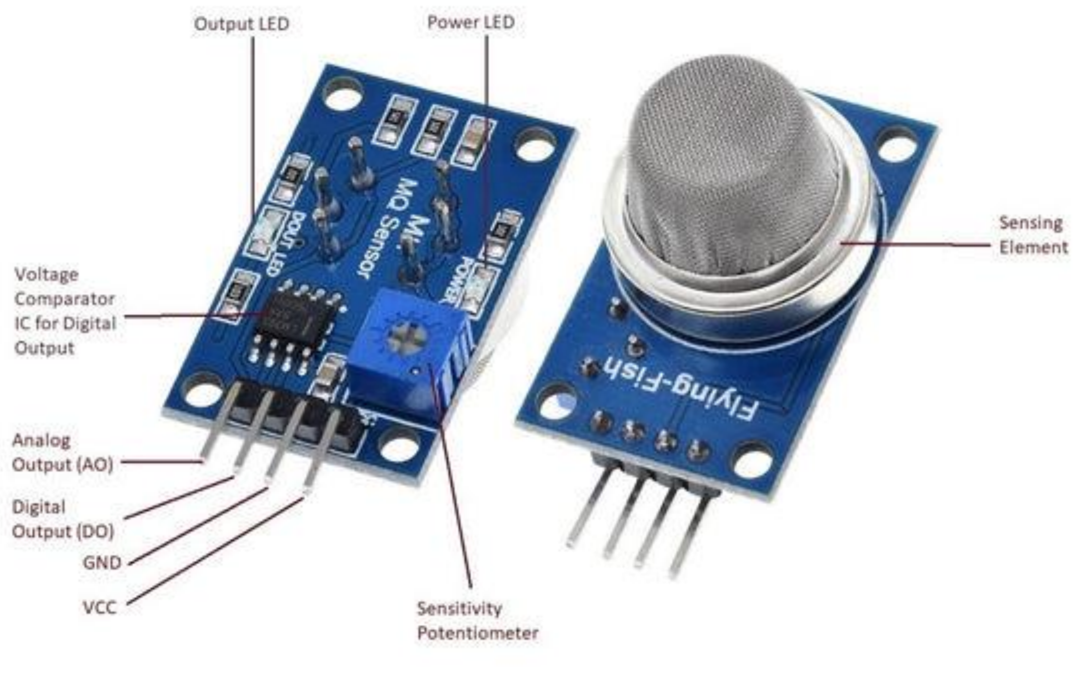
## COMPONENTS: -

1. MQ2 (Sensor)
2. DHT11 (Temperature and Humidity Sensor)
3. Oled Display
4. ESP32 (WiFi-Module)

## MQ2

The MQ2 gas sensor is a versatile and widely used sensor for detecting a variety of gases, primarily used in applications like gas leakage detection in homes and industries. Below is detailed information about the MQ2 gas sensor:

### MQ2 Diagram:-



### Specifications

- Operating Voltage: 5V
- Operating Current: 150 mA
- Load Resistance: Adjustable (typically 4.7KΩ)

- Heater Resistance: 31Ω ± 3Ω
- Sensing Resistance: 2KΩ - 20KΩ (depending on the gas concentration)
- Heating Consumption: ≤ 800 mW
- Sensor Type: Analog and Digital
- Dimensions: 32 mm x 22 mm x 27 mm (approx.)

## Features

- **Sensitivity:** High sensitivity to LPG, Propane, Hydrogen, Methane, Alcohol, and smoke.
- **Fast Response Time:** Responds quickly to gas presence.
- **Long Life and Low Cost:** Durable with a lifespan of several years and cost-effective.
- **Simple Circuitry:** Easy to integrate with microcontrollers, such as Arduino, ESP32, and others.
- **Dual Output:** Analog output for variable concentration measurement and digital output for threshold-based detection.

## Pin Configuration

1. **VCC**: Power supply pin (typically 5V)
2. **GND**: Ground pin
3. **D0**: Digital output pin (high/low signal when gas concentration exceeds a threshold)
4. **A0**: Analog output pin (provides an analog voltage that varies with gas concentration)

## Operational Concept

The MQ2 sensor has an electrochemical sensor that responds to the presence of gas and a tiny heating element. The gas interacts with the material of the sensor when it gets heated, altering the electrical resistance of the sensor. After that, the change is transformed into an analog signal that a microcontroller can read and process.

## Applications

- Gas leakage detection in homes and industrial setups.
- Safety systems for detecting flammable and combustible gases.
- Environmental monitoring.
- Breath analyzers.
- Smoke detectors.

## Circuit Diagram

A typical circuit diagram for using the MQ2 sensor with an Arduino or similar microcontroller is as follows:

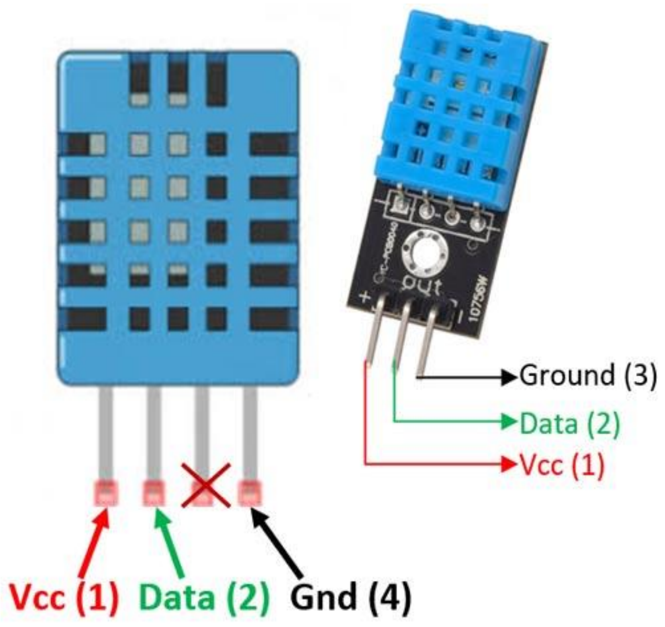
1. Connect the VCC pin of the MQ2 sensor to the 5V pin of the microcontroller.

2. Connect the GND pin to the ground.
3. Connect the A0 (analog output) pin to an analog input pin on the microcontroller (e.g., A0 on an Arduino).
4. Optionally, connect the D0 (digital output) pin to a digital input pin on the microcontroller for threshold-based detection.

# DHT11

The DHT11 is a popular temperature and humidity sensor widely used in various electronic projects. Below is detailed information about the DHT11 sensor:

## DHT11 Diagram:-



## Specifications

- Humidity Range: 20-90% RH
- Humidity Accuracy: ±5% RH
- Temperature Range: 0-50°C
- Temperature Accuracy: ±2°C
- Operating Voltage: 3.3V to 5.5V
- Current Consumption: 0.3mA (measuring), 60µA (standby)
- Sampling Rate: 1 Hz (one reading per second)
- Interface: Single-wire digital interface
- Dimensions: 15.5mm x 12mm x 5.5mm

## Pin Configuration

1. VCC: Power supply (3.3V to 5.5V)
2. Data: Serial data output
3. Not Connected (NC)
4. GND: Ground

## Features

- Low Cost: Affordable for most hobbyist projects.
- Easy to Interface: Simple communication with microcontrollers using a single digital pin.
- Compact Size: Fits easily into various designs.

## Working Principle

The DHT11 sensor uses a resistive-type humidity measurement component and an NTC temperature measurement component. It includes a high-performance 8-bit microcontroller that outputs the temperature and humidity readings as a digital signal.

## Communication Protocol

The DHT11 uses a proprietary 1-wire protocol to communicate with microcontrollers. The communication process involves:

1. Start Signal: The microcontroller sends a start signal (low for at least 18ms) to the DHT11.
2. Response Signal: The DHT11 responds with a low signal for 80µs followed by a high signal for 80µs.
3. Data Transmission: The DHT11 sends 40 bits of data (16 bits for humidity, 16 bits for temperature, and 8 bits for the checksum).

## Applications

- Weather Stations: For monitoring environmental conditions.
- HVAC Systems: To maintain optimal indoor air quality.
- Greenhouses: To ensure proper growing conditions.
- Smart Homes: For automation and monitoring of living conditions.

## Advantages

- Affordable: Low cost makes it suitable for educational and hobby projects.
- Simple to Use: Easy to integrate with various microcontrollers.
- Low Power Consumption: Suitable for battery-operated devices.

## Limitations

- Accuracy: Less accurate compared to some other sensors.

- Range: Limited humidity and temperature range.
- Sampling Rate: Limited to one reading per second.

Because of its affordability and ease of use, the DHT11 is a fantastic option for novices and enthusiasts wishing to add temperature and humidity monitoring to their projects.

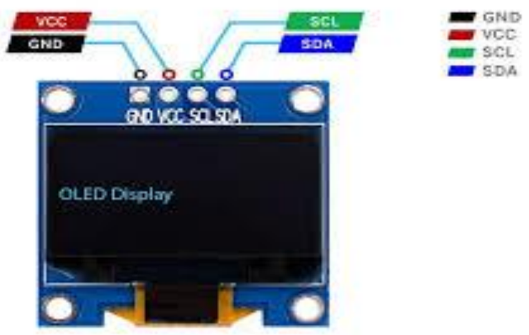
## Oled Display

Making sure you have the appropriate libraries and understand how to format the data appropriately will be necessary if you want to use the ESP32 for your Air Pollution Monitoring System to show all of the information on an OLED. Here's a step-by-step setup instruction to assist you.

### Libraries Needed

- 'Wire.h' - for I2C communication.
- 'Adafruit_GFX.h' - for graphics functions.
- 'Adafruit_SSD1306.h' - for the OLED display.

### Oled Display:-



### Connections

Ensure your OLED is connected correctly to the ESP32:

- VCC to 3.3V
- GND to GND
- SCL to GPIO 22 (default for ESP32 I2C)
- SDA to GPIO 21 (default for ESP32 I2C)

## Explanation:

1. Initialization:
   - Initialize the OLED display and DHT11 sensor in the setup() function.
2. Reading Sensors:
   - In the loop() function, read the temperature and humidity from the DHT11 sensor.
   - Read the air quality value from the MQ135 sensor.
3. Displaying Data:
   - Clear the display before updating.
   - Use setCursor() to set the position where you want to display text.
   - Use print() and println() to print the sensor values on the display.
4. Loop Delay:
   - The loop runs every 2 seconds, updating the display with new sensor readings.

## Customization

If more sensor data is required, you can add it and change the text's size. The code mentioned above offers a fundamental framework to initiate the process of presenting data on the OLED.

# ESP32 (WiFi Module)

The ESP32 is a multipurpose Wi-Fi and Bluetooth combo chip that may be used for a variety of functions, including speech encoding, music streaming, MP3 decoding, and low-power sensor networks. An extensive description of the ESP32 Wi-Fi module can be found below:

## ESP32 Diagram:-

## Key Features

1. Wireless Connectivity:
   - Wi-Fi: 802.11 b/g/n
   - Bluetooth: v4.2 BR/EDR and BLE
2. Core:
   - Dual-core Tensilica LX6 microprocessor
   - Clock frequency: up to 240 MHz
   - Performance: up to 600 DMIPS
3. Memory:
   - 520 KB SRAM
   - External memory: Support for up to 16 MB external SPI flash

4. Peripherals:
    - 34 programmable GPIOs
    - 12-bit SAR ADC up to 18 channels
    - 2 × 8-bit DACs
    - 10 × touch sensors
    - 4 × SPI, 2 × I2S, 2 × I2C, 3 × UART
    - SD/SDIO/MMC host controller
    - Ethernet MAC
    - CAN 2.0
    - Infrared remote controller (TX/RX)
    - Hall sensor
    - Temperature sensor
5. Power Management:
    - Wide range of low-power modes
    - Power consumption as low as 5 µA in deep sleep mode
6. Security:
    - WPA/WPA2/WPA3 personal and enterprise
    - Secure boot
    - Flash encryption
    - Cryptographic hardware acceleration (AES, SHA-2, RSA, ECC, RNG)
7. Development Support:
    - Supported by Espressif's software development framework: ESP-IDF (IoT Development Framework)
    - Compatible with Arduino IDE, PlatformIO, and other third-party development environments

## Applications

- IoT (Internet of Things): Smart home devices, industrial automation, wearables, health monitoring systems
- Consumer Electronics: Audio streaming devices, video over Wi-Fi
- Communication Systems: Mesh networks, gateways
- Embedded Systems: Robotics, drones

## Advantages

- Low Cost: High performance at a low price point
- Flexibility: Can be used for a wide range of applications due to its extensive peripheral set and connectivity options
- Community and Support: Strong community support with numerous online resources, tutorials, and libraries

## Common Development Boards

- ESP32 DevKitC: One of the most popular development boards, includes a USB-to-serial converter, a few buttons, and an integrated antenna
- ESP32-WROVER Kit: Includes additional PSRAM and an onboard camera interface
- NodeMCU-32S: Based on the ESP-32S module, integrates USB to serial, a power circuit, and basic GPIO access

## Getting Started

1. Setup:
    - Install the required drivers for your development board.
    - Set up your development environment (Arduino IDE, PlatformIO, or ESP-IDF).
2. Programming:
    - Use examples and libraries to start with basic projects like blinking an LED, reading sensor data, or connecting to a Wi-Fi network.
    - Gradually move to more complex projects involving multiple sensors, actuators, and communication protocols.
3. Debugging:
    - Use the serial monitor to print debug information.
    - Utilize tools like JTAG for more advanced debugging.

Because of its adaptability, robust capabilities, and robust community support, the ESP32 is a well-liked option for embedded systems and Internet of Things development enthusiasts as well as pros.