Air Quality Monitoring in IoT

Air quality monitoring is a critical process that involves the continuous measurement and analysis of various Air pollutants, as particulate matter, gases, and volatile organic compounds. This data helps authorities and communities assess pollution levels, make informed decisions, and take actions to protect public health and the environment. Advanced sensor technologies and networks play a key role in

providing real-time data for effective air quality management. Air quality monitoring in IoT (Internet of Things) utilizes a network of interconnected sensors and devices to continuously collect and transmit real-time data on air pollutants like PM2.5, VOCs, and more. This technology enables remote monitoring, data analysis, and prompt responses to air quality changes, enhancing our ability to mitigate pollution and protect public health. IoT-based air quality monitoring systems are increasingly integrated into smart cities and homes to provide valuable insights and alerts.

Air quality monitoring involves assessing and measuring pollutants in the atmosphere to gauge its health and environmental impact.

Air Pollution:

Air pollution refers to the presence of harmful substances, known as pollutants, in the Earth's atmosphere. These pollutants can be natural, like dust and volcanic ash, or human-made, such as industrial emissions

and vehicle exhaust. Common air pollutants include particulate matter (PM2.5 and PM10), sulfur dioxide (SO2), nitrogen oxides (NOx), carbon monoxide (CO), and volatile organic compounds (VOCs).

Air pollution has severe environmental and health consequences. It contributes to global warming, damages ecosystems, and harms human health. Exposure to polluted air can lead to respiratory problems, cardiovascular diseases, and even premature death. Sources of air pollution include transportation, industrial processes, agriculture, and energy production.

Efforts to combat air pollution include regulations, emission controls, promoting clean energy sources, and public awareness campaigns. Reducing air pollution is essential for a sustainable and healthy future.

Requirements:

While DHT22 sensor is not the ideal choice for air Quality monitoring (as it's typically used for temperature measurement), you can still use an ESP32, an symbols,and an LCD display for an educational or demonstration project related to air quality. This project can simulate air levels based on proximity to the DHT22 sensor. Keep in mind that it won't provide accurate temperature or humidity data

Here's how you can create a basic project using an ESP32, an DHT22 sensor, an LCD display and a symbols.

Hardware Set-up:

Connect the DHT22 sensor pins to the Esp32 following connection.

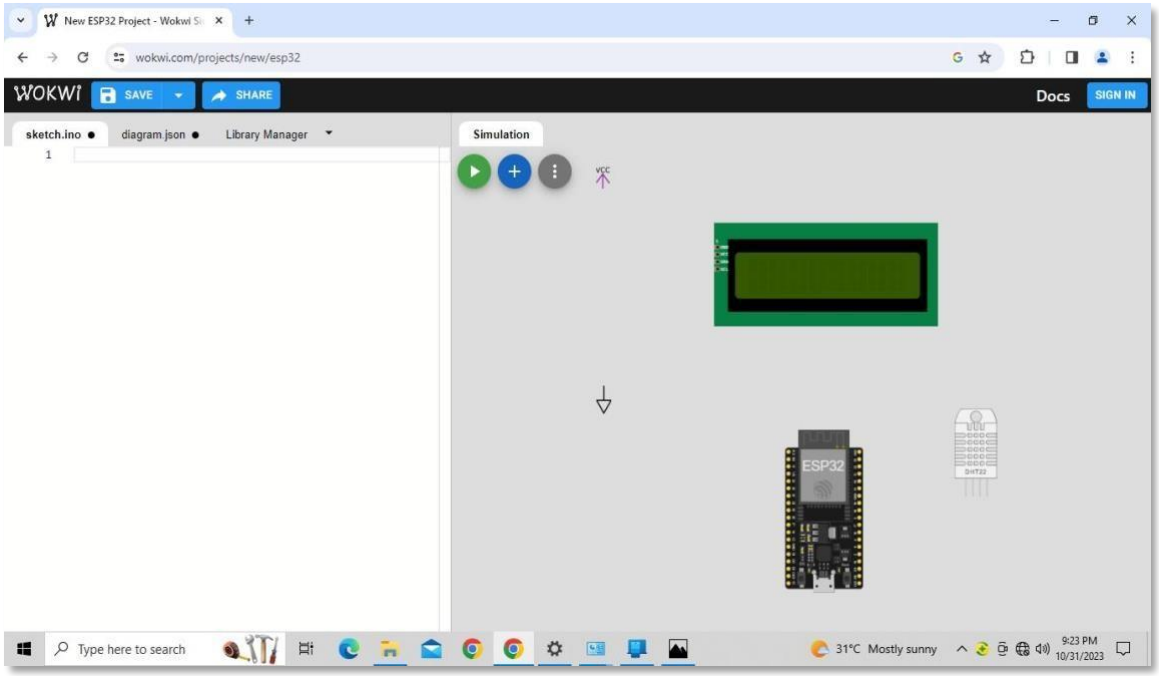
Connect the LCD pins to the required Ep32 pins.

Connect the LCD (VCC and GND) to its corresponding symbols.

Software Set-up:

You will need to install the required libraries for the LCD display and DHT sensor.

In this project we'll use the LiquidCrysral I2C DHT sensor library for ESPx libraries to simulate the air quality levels based on calefaction.



ESP32 :

Connect your ESP32 to your computer for programming and power.

DHT22:

Connect the DHT:VCC to Esp:CLCK Pin.

Connect the DHT:SDA to Esp:D1 Pin.

Connect the DHT:GND to Esp:D0 Pin.

DISPLAY: {eg..,LCD 16×2(I2C)}:

Connect the display(VCC and GND) to its corresponding VCC and GND Symbols Pins.

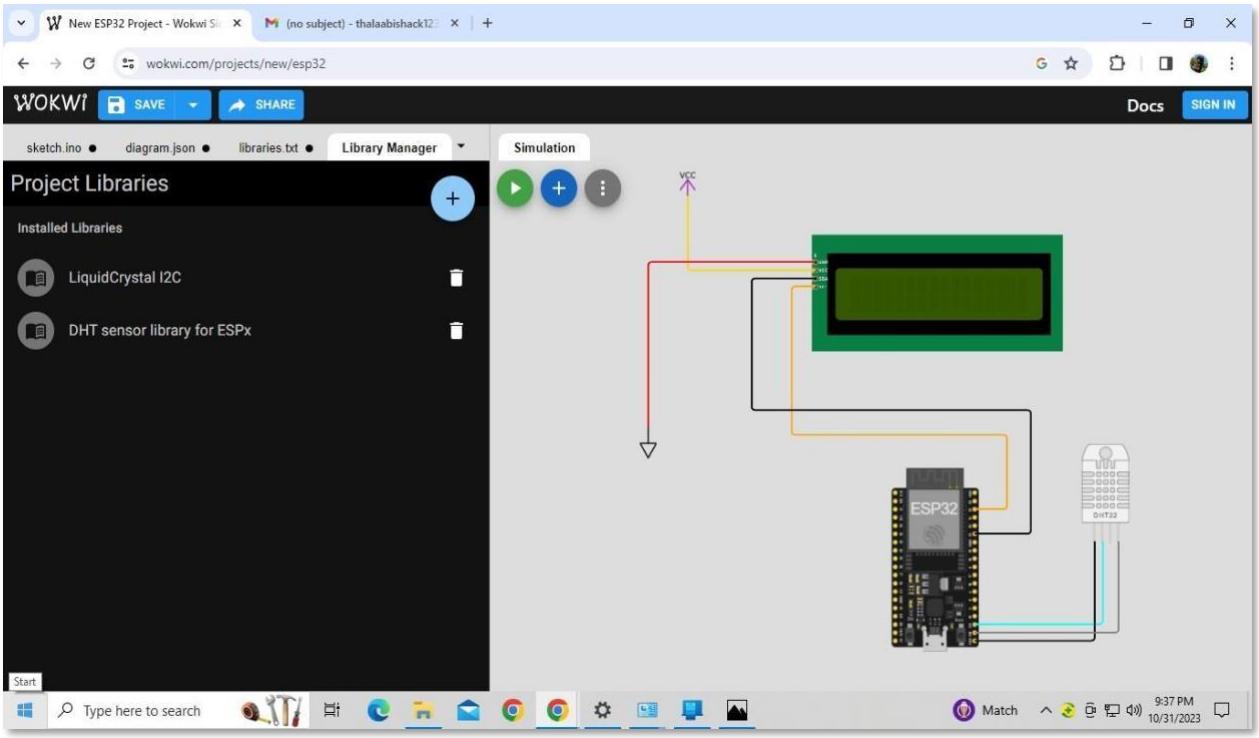
Connect the display(SDA) to the Esp:21 Pin.

Connect the display(SCL) to the Esp:22 Pin.

LIBRARIES:

Install the required libraries to run the program

- Δ LiquidCrysral I2C ,
- Δ FHT sensor library for ESPx .



Using an Esp32 ,an DHT22 sensor and an LCD display for air quality monitoring while not the most accurate method, can be simplified educational or demonstration project to raise awareness about air pollution .Here's a steep-by-step process to set this project.

Operation:

When the set-up is complete and power on the Esp32 ,the DHT sensor will measure the temperature and humidity in its surroundings and display it on the LCD display.

Purpose:

Air quality monitoring using an ESP32 microcontroller and a DHT22 sensor serves the purpose of assessing environmental conditions in a specific location. Here's how this combination can be beneficial:

- Real-Time Data Collection: The ESP32, equipped with a DHT22 sensor, can continuously collect real-time data on temperature and humidity. This information is valuable for understanding the immediate environmental conditions.

- Indoor Air Quality:The DHT22 can provide data on indoor temperature and humidity, which is essential for maintaining a comfortable and healthy indoor environment. Monitoring indoor air quality can help identify issues such as poor ventilation or high humidity levels.

- Basic Pollution Monitoring:While the DHT22 primarily measures temperature and humidity, it can also offer insights into indoor air quality to some extent. For example, a significant increase in humidity might indicate the presence of moisture or potential mold growth.

- DIY Projects: Using an ESP32 and DHT22 for air quality monitoring can be part of DIY projects and educational initiatives to teach individuals about environmental monitoring and sensor technology.

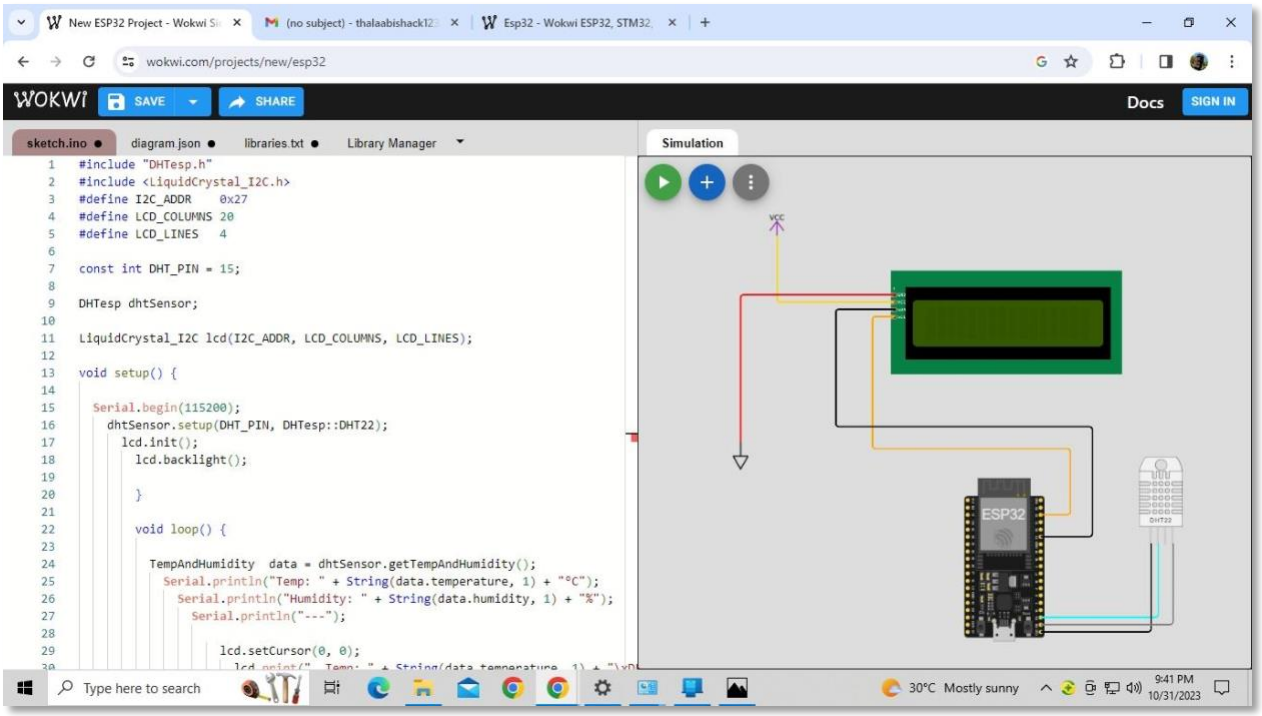
It's important to note that the DHT22 sensor is not specifically designed for comprehensive air quality monitoring in terms of pollutant detection. If you want to monitor specific air pollutants (e.g., particulate matter, gases like CO2 or CO), you would need additional sensors that are designed for that purpose. Nonetheless, using an ESP32 and DHT22 is a cost-effective way to start with basic environmental monitoring and can be a valuable component of a larger monitoring system.

Codes to stimulate:

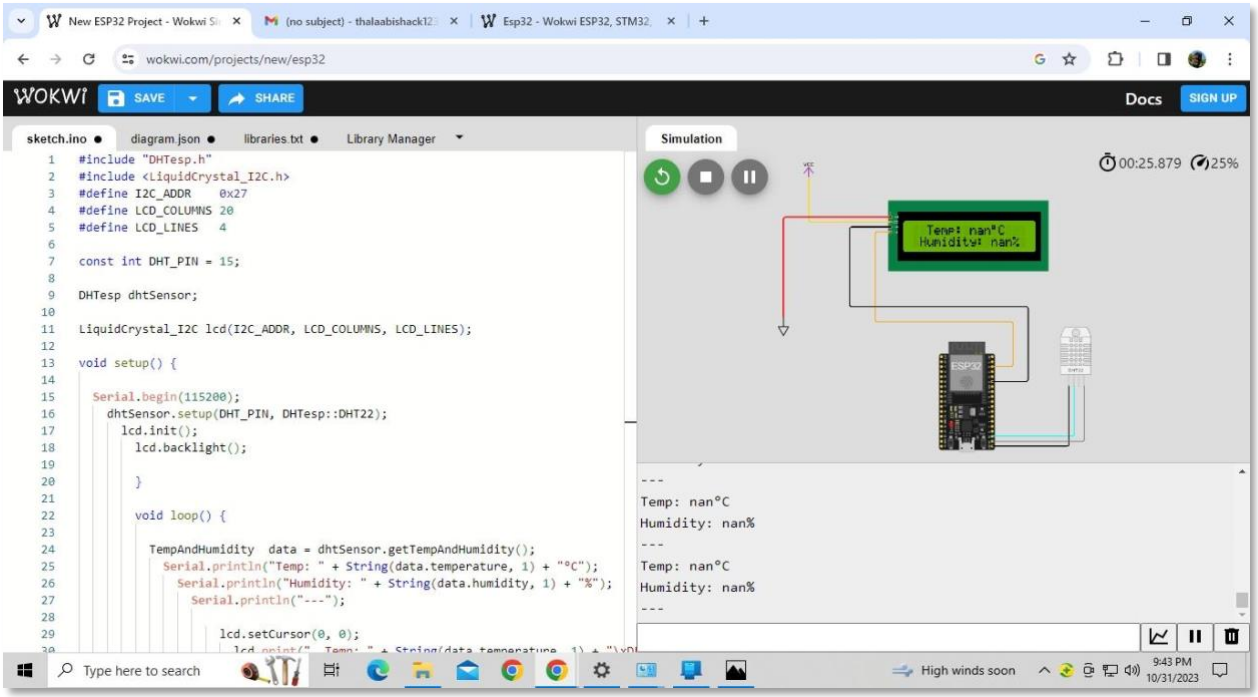
```
#include "DHTesp.h"
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR    0x27
#define LCD_COLUMNS 20
#define LCD_LINES   4
Const int DHT_PIN = 15;
DHTesp dhtSensor;
LiquidCrystal_I2C lcd(I2C_ADDR, LCD_COLUMNS, LCD_LINES);
Void setup() { Serial.begin(115200);
dhtSensor.setup(DHT_PIN, DHTesp::DHT22); lcd.init();

lcd.backlight(); }
Void loop(){
TempAndHumidity  data = dhtSensor.getTempAndHumidity();
```

```
Serial.println("Temp: " + String(data.temperature, 1) + "°C");
Serial.println("Humidity: " + String(data.humidity, 1) + "%");
Serial.println("---");
Lcd.setCursor(0, 0);
Lcd.print("  Temp: " + String(data.temperature, 1) + "\xDF"+"C  ");
Lcd.setCursor(0, 1);
Lcd.print(" Humidity: " + String(data.humidity, 1) + "% ");
Lcd.print("Wokwi Online IoT");
Delay(1000);
}
```



To compile and upload code to the Esp32 using Visual Studio Code(VS Code), You'll need to set up the necessary Tools and extensions. Here's a step-by-step guide on how to compile and upload code to an Esp32 using vs code.



Prerequisites:

Install the Arduino IDE: If you haven't already, install the Arduino IDE, which includes the required ESP32 board support package.

Steps:

Install Visual Studio Code:

If you don't have Visual Studio Code installed, download and install it from the official website.

Install Visual Studio Code Extensions:

Open VS Code.

Install the following extensions if you haven't already:

"PlatformIO" extension (for managing libraries and building/uploading firmware).

"C/C++" extension (for code editing and highlighting).

"Arduino" extension (provides Arduino-specific support).

Set Up PlatformIO:

Open VS Code.

Go to the PlatformIO Home tab.

Click “Quick Access” or “Project Examples.”

Click “New Project” to create a new PlatformIO project.

Select the ESP32 platform for your project.

Choose a location for your project directory.

Specify a name for your project.

Write Your Code:

Write or paste your Arduino code into the src folder of your PlatformIO project.

Configure PlatformIO:

Open the platformio.ini file in your project directory.

Ensure you have the correct settings for your ESP32 board, upload speed, and serial port.

Build and Upload:

Open the PlatformIO terminal from the VS Code terminal menu.

Use the following PlatformIO commands to build and upload your code to the ESP32:

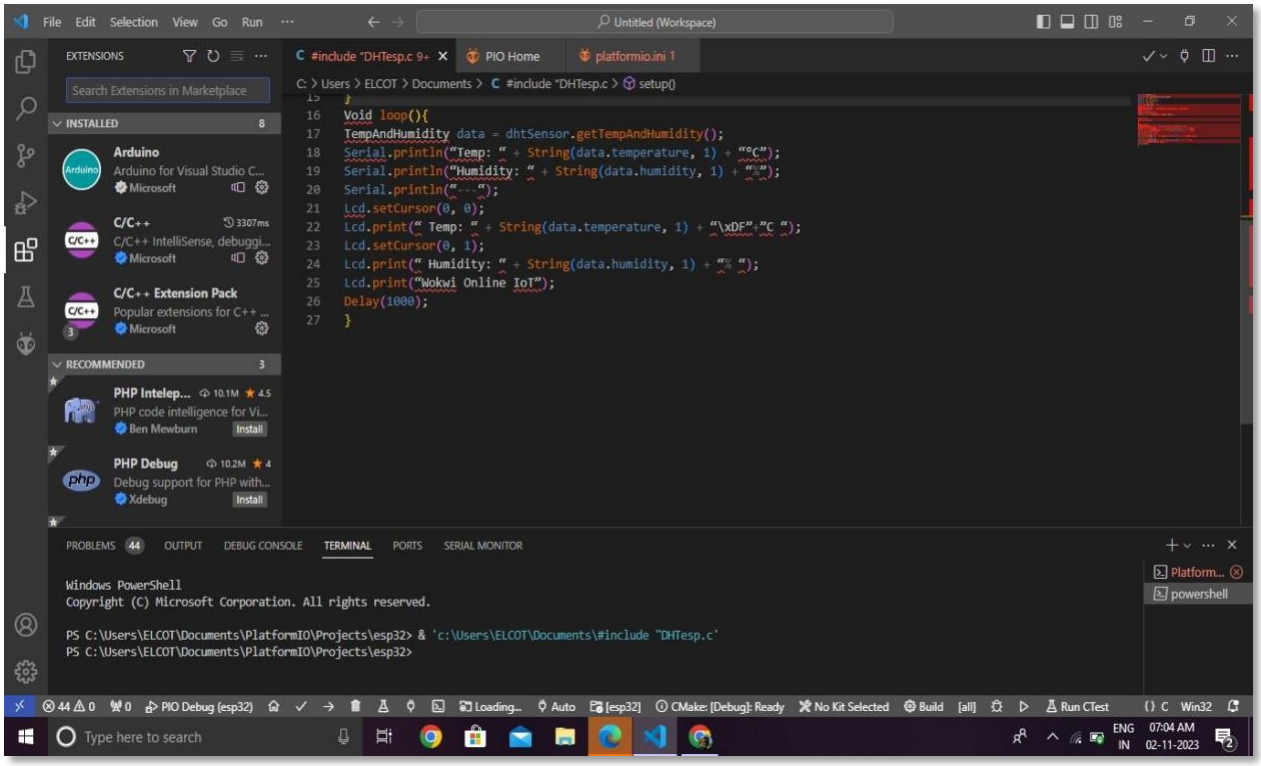
Pio run to compile your code.

Pio run -t upload to upload the code to the ESP32.

Monitor Serial Output

You can use the “Serial Monitor” feature in VS Code to view the serial output of your ESP32. Make sure you have the correct baud rate specified in your code.

With these steps, you should be able to compile and upload code to your ESP32 using Visual Studio Code and the PlatformIO extension. This setup provides a convenient and feature-rich development environment for working with ESP32 and Arduino-based projects.



Creating a practical application for air quality monitoring using esp32 , an LCD display, and DHT sensor and a symbols may be a bit unconventional ,as DHT22 sensor is not typically used for air quality monitoring. However ,you can create a educational or interactive project to raise awareness about air pollution and demonstrate the concept in a simplified Manner.

Here's an example of an application that uses these components:

Application: DIY Indoor Air Quality Functionality

Components Needed:

- ✓ ESP32 microcontroller.
- ✓ DHT22 sensor (for temperature and humidity).
- ✓ OLED display (optional but recommended for real-time data visualization).
- ✓ Power source (e.g., USB power supply or a rechargeable battery).
- ✓ Enclosure for housing the components (optional).

Functionality:

- ✓ The ESP32 continuously reads temperature and humidity data from the DHT22 sensor.

- ✓ The collected data is displayed on the OLED screen in real-time, providing users with information about the indoor environment.

Features:

- ✓ Real-time Temperature and Humidity Display: The OLED screen shows the current temperature and humidity levels.
- ✓ User Alerts: Set threshold values for temperature and humidity. When these thresholds are exceeded, the system can trigger visual or audible alerts.
- ✓ Data Logging: Store historical data on the ESP32's memory or upload it to a cloud service for later analysis.
- ✓ Mobile App Integration: Develop a mobile app that connects to the ESP32 via Wi-Fi, allowing users to monitor air quality remotely and receive alerts on their smartphones.

Use Cases:

- ✓ Home Air Quality Monitoring: Use it in homes to ensure comfortable indoor conditions and to detect potential issues like excess humidity.
- ✓ Baby Room Monitor: Install it in a nursery to keep an eye on temperature and humidity for a baby's comfort.
- ✓ Classroom or Office Environment: Implement this system in educational or work environments to create a comfortable atmosphere.
- ✓ Learning Tool: Use it as an educational tool to teach students or enthusiasts about air quality monitoring and IoT.

Remember that this setup is primarily for monitoring temperature and humidity. If you want to monitor specific air pollutants like particulate matter or gases, you'd need additional sensors designed for those purposes. Additionally, you could expand the application by integrating more advanced features and connecting to a cloud platform for long-term data storage and analysis.