Abstract

This project presents a real-time environmental monitoring system using the ESP8266 microcontroller, BME280 sensor, and P10 LED display panel. The system measures temperature, humidity, and pressure, displaying the data on a scrolling LED panel and storing it in Google Sheets for further analysis. The project demonstrates an efficient and cost-effective approach to monitoring environmental parameters for smart city applications.

1. Introduction

Environmental monitoring is essential in various fields, including agriculture, weather forecasting, and pollution control. This project integrates hardware and software components to develop a functional system capable of measuring and displaying real-time data. The ESP8266, known for its Wi-Fi capabilities, and the BME280 sensor, renowned for its precision, form the core components of this system.

2. System Design

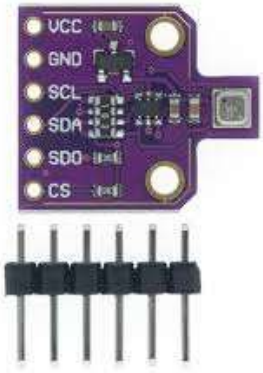
2.1 Hardware Design

The system consists of the following hardware components:

2.1.1 ESP8266 Microcontroller

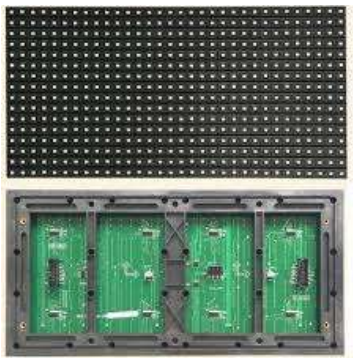
- **Role in the Project**: The ESP8266 acts as the central controller, managing communication between the BME280 sensor, the P10 LED display, and Google Sheets.
- **Usage**: It processes the sensor data and sends it to the LED panel for display and to the cloud for storage.
- **Why Used**: Its built-in Wi-Fi capability makes it ideal for IoT applications.

2.1.2 BME280 Sensor



- **Role in the Project**: This sensor is used to measure temperature, humidity, and atmospheric pressure.
- **Usage**: It communicates with the ESP8266 via I2C protocol to provide real-time environmental data.
- **Why Used**: The BME280 is highly accurate, energy-efficient, and compact, making it a reliable choice for environmental monitoring.

2.1.3 P10 LED Panel



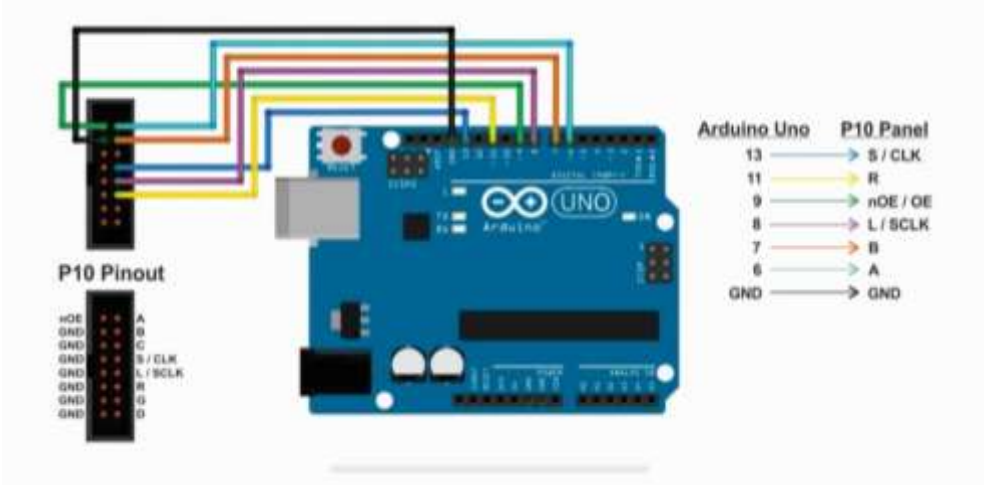
- **Role in the Project**: Displays the real-time sensor data in a scrolling text format.
- **Usage**: Controlled by the ESP8266 using SPI communication. Data such as temperature, pressure, and humidity is dynamically updated and displayed.
- **Why Used**: The panel provides a bright, clear, and scalable solution for displaying real-time data, suitable for public environments.

2.1.4 Power Supply

- **Role in the Project**: Provides the necessary voltage and current to the components.
- **Usage**: A stable 5V power supply is used to ensure uninterrupted operation of the ESP8266, BME280 sensor, and P10 LED panel.
- **Why Used**: Ensures stability and prevents issues like voltage drops that could disrupt the system.

Pin Configuration

The following table shows the pin mapping for connecting the P10 LED panel to the ESP8266:



2.2 Software Design

The software integrates multiple libraries to manage sensor readings, LED panel control, and data storage in Google Sheets. Key libraries include:

1. **SPI.h**: Enables SPI communication with the P10 LED panel.
2. **DMD.h**: Provides functions to control the P10 LED panel, including scrolling text and display settings.
3. **TimerOne.h**: Sets up a timer interrupt to manage LED scanning.
4. **Wire.h**: Handles I2C communication with the BME280 sensor.
5. **Adafruit_Sensor.h** and **Adafruit_BME280.h**: Interface with the BME280 sensor to read temperature, pressure, and humidity.
6. **Arial_Black_16_ISO_8859_1.h**: A font library for rendering text on the LED panel.
7. **Google Sheets API Integration**:
    - **Role in the Project**: Enables cloud storage of environmental data.
    - **Implementation**:
        - Use the ESP8266's Wi-Fi capabilities to send data to Google Sheets.
        - Utilize third-party libraries or tools such as IFTTT, Google Apps Script, or Firebase for seamless integration.
        - Format the data in JSON or HTTP POST requests to communicate with the Google Sheets API.
    - **Why Used**: Provides a scalable and accessible solution for data logging and analysis.

3. Working Procedure

1. **Initialization**:
    - Power up the ESP8266 microcontroller and initialize the BME280 sensor and P10 LED panel.
    - Establish Wi-Fi connectivity for cloud integration.

2. **Data Acquisition**:
    - The BME280 sensor continuously measures temperature, humidity, and pressure.
    - The ESP8266 reads sensor data using I2C communication.
3. **Data Display**:
    - The sensor data is formatted into a scrolling text message.
    - The ESP8266 controls the P10 LED panel using SPI communication to display the formatted data.
4. **Cloud Storage**:
    - The sensor data is packaged into an HTTP POST request.
    - The ESP8266 sends the data to Google Sheets via the Google Sheets API or other intermediary tools like IFTTT.
    - Data is stored in a tabular format for further analysis.
5. **Real-Time Updates**:
    - The display and cloud storage are updated periodically to reflect real-time environmental conditions.
6. **Monitoring and Debugging**:
    - Use Serial Monitor in the IDE to view real-time data and debug any issues in the system.

4. Testing and Evaluation

4.1 Implementation Code

The implementation code integrates the hardware and software components for real-time monitoring and display.

Code

```
#include <SPI.h>

#include <DMD.h>

#include <TimerOne.h>

#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>

#include "Arial_Black_16_ISO_8859_1.h"



// Panel configuration
```

```
#define DISPLAYS_ACROSS 2

#define DISPLAYS_DOWN 1



DMD dmd(DISPLAYS_ACROSS, DISPLAYS_DOWN);

Adafruit_BME280 bme;



char displayText[100]; // Text buffer for scrolling

long marqueeTimer = 0; // Timer for marquee scrolling

int marqueeSpeed = 50; // Scrolling speed in milliseconds

int scrollPosition = (32 * DISPLAYS_ACROSS) - 1; // Start at the
rightmost edge



// DMD pin configuration

#define PIN_DMD_CLK 13

#define PIN_DMD_OE 9

#define PIN_DMD_LAT 8

#define PIN_DMD_A 6

#define PIN_DMD_B 7

#define PIN_DMD_DAR 11



void ScanDMD() {

  dmd.scanDisplayBySPI();

}
```

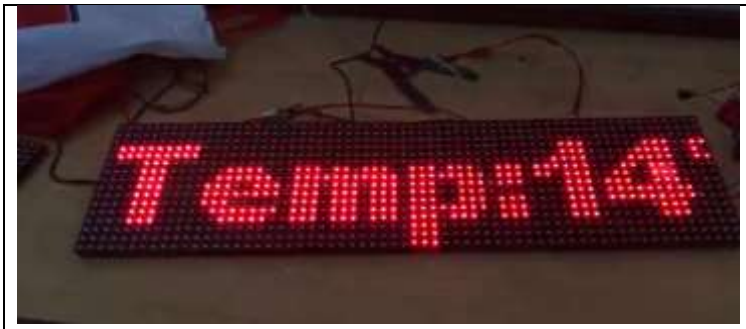
```
void setup() {

  Wire.begin();

  Serial.begin(9600);



  // Initialize BME280

  if (!bme.begin(0x76)) { // Adjust the I2C address if necessary

    Serial.println("Could not find a valid BME280 sensor, check
wiring!");

    while (1);

  }



  // Initialize DMD

  Timer1.initialize(1000); // Timer for refreshing the DMD
delay(1)

  Timer1.attachInterrupt(ScanDMD); //ScanDMD();

  dmd.clearScreen(true);

  dmd.selectFont(Arial_Black_16_ISO_8859_1);

}



void loop() {

  // Read BME280 values

  int temperature = bme.readTemperature();

  int pressure = bme.readPressure() / 100.0F; // Convert from Pa
```

```
to hPa
  int humidity = bme.readHumidity();


  char degreeSymbol = 176;


  // Prepare the scrolling text
  snprintf(displayText, sizeof(displayText),
           "Temp:%d%cC  Pres:%dhPa  Hum:%d%%
",
           temperature, degreeSymbol, pressure, humidity);


  // Display on the SMD panel with scrolling
  long currentTime = millis();
  if (currentTime - marqueeTimer > marqueeSpeed) {
    dmd.clearScreen(true); // Clear the screen for the new frame
    dmd.drawMarquee(displayText, strlen(displayText),
scrollPosition, 0);
    scrollPosition--; // Move text left


    // Reset scroll position when text is off-screen
    if (scrollPosition < -((int)strlen(displayText) * 6)) { // 6
pixels per character
      scrollPosition = (32 * DISPLAYS_ACROSS) - 1; // Reset to
starting position
    }
```

```
    marqueeTimer = currentTime;

  }



  // Debugging information

  Serial.print("Temperature: "); Serial.print(temperature);
Serial.println(" C");

  Serial.print("Pressure: "); Serial.print(pressure);
Serial.println(" hPa");

  Serial.print("Humidity: "); Serial.print(humidity);
Serial.println(" %");

}
```

4.2 Output

- **Real-Time Display**: Successfully displayed temperature, humidity, and pressure data on the P10 LED panel.
- **Cloud Integration**: Data was stored in Google Sheets for further analysis, though occasional communication delays were observed.



5. Challenges Faced

1. **Hardware Integration**: Identifying correct SDA and SCL pins for stable communication.

2. **Cloud Storage**: Data storage worked during software testing but failed when hardware was connected.
3. **LED Scrolling Optimization**: Achieving smooth text transitions on the P10 panel required fine-tuning.
4. **Dislpaying Data**: The ESP8266 did not correctly interpret the %C format specifier in string
5. **Timer One**: TimerOne library was not supported by the ESP8266.

6. Learning Outcomes

- Mastered I2C communication and sensor interfacing.
- Gained proficiency in debugging hardware-software integration issues.
- Acquired hands-on experience with IoT-based cloud storage solutions.

7. Future Enhancements

- **Enhanced Connectivity**: Implement MQTT protocol for real-time, low-latency data transfer.
- **Expanded Metrics**: Incorporate additional sensors for air quality, light intensity, and noise pollution.
- **Mobile App**: Develop a companion app for real-time monitoring and notifications.

8. Conclusion

This project demonstrates a scalable solution for environmental monitoring. By leveraging IoT and cloud technologies, the system efficiently captures and displays real-time data, making it a valuable tool for smart city applications. Future enhancements will expand its functionality and reliability.