Classroom Air Quality Monitor

Mosawar Jamshady, Johnny Rosas, Angel Guzman, Hengcheng Zhang

Motivation

Poor indoor air quality can negatively impact students' health and cognitive performance

Project Objectives: Monitor classroom air quality in real time and send alerts when conditions deteriorate

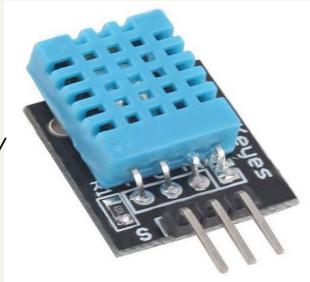
Overview

The Air Quality Monitoring System is designed to continuously monitor critical air quality parameters such as carbon dioxide (CO_2) concentration, temperature, and humidity.

- **The system uses sensors and automated notifications to proactively address air quality issues. When CO_2 levels rise above a safe threshold or temperature and humidity reach undesirable levels, the system triggers an alarm to prompt immediate action, such as opening windows or turning on the air conditioner.**
- **Sends real-time alerts via SMS, enabling prompt action and ensuring that the area's conditions remain optimal.**

System Architecture

MH-Z19C CO_2 sensor



DHT11 Temperature and Humidity Sensor

MH-Z19C CO_2 sensor: collects CO_2 concentration data.
DHT11 temperature and humidity sensor: collects temperature and humidity data.

ESP32: Connect sensors and aggregate data. Send data to Raspberry Pi via WiFi.

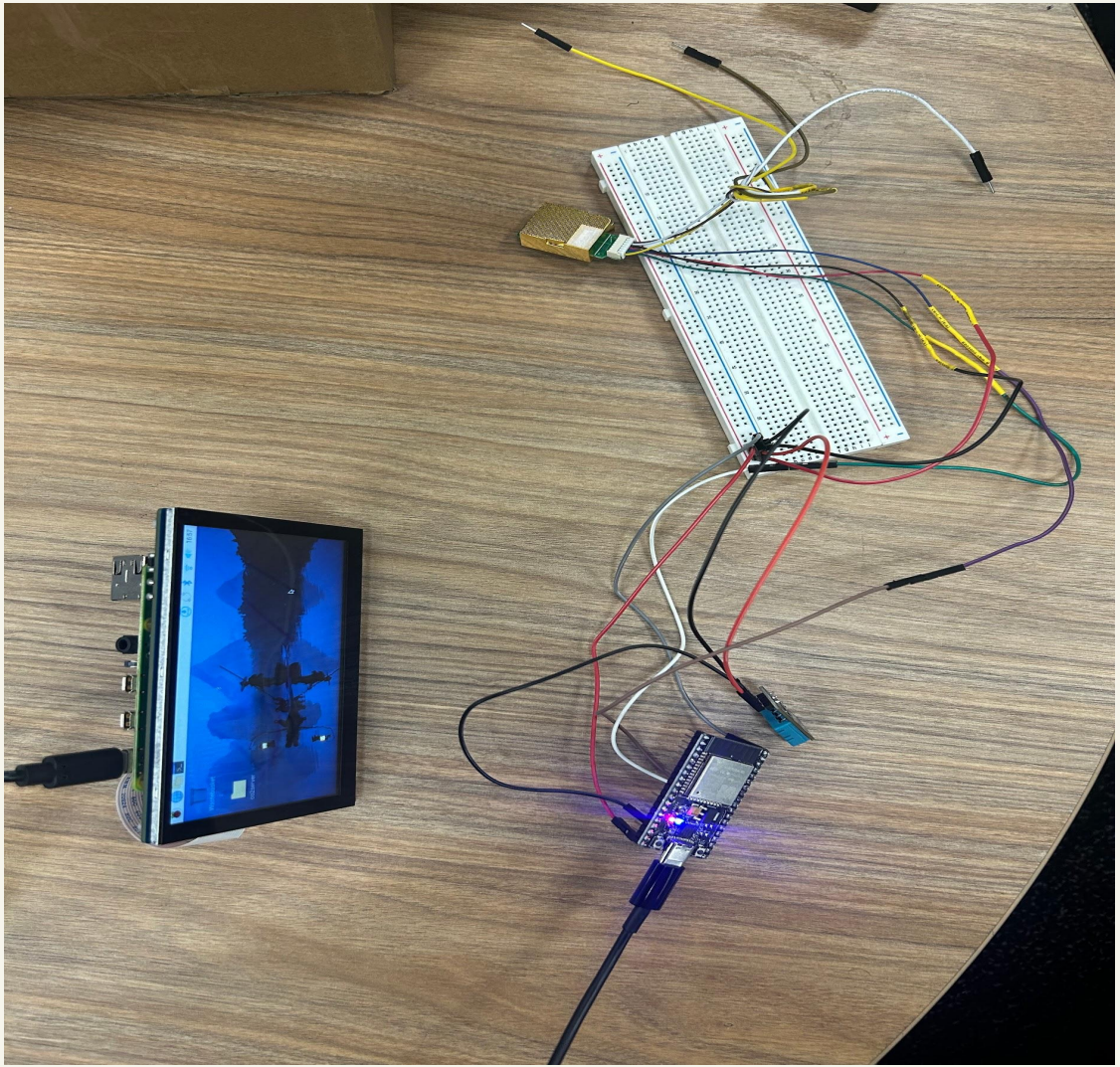
Raspberry Pi: Receive and process data. Trigger an alarm and send SMS notifications based on conditions.

ESP32

Raspberry Pi

SMS Alert

Classroom Air Quality Monitor



Features & Implementation


```python
def get_co2_status(co2_value):
    if co2_value <= 799:
        return "Good"
    elif 800 <= co2_value <= 1099:
        return "Moderate"
    elif 1100 <= co2_value <= 1499:
        return "Poor"
    elif 1500 <= co2_value <= 1999:
        return "Unhealthy"
    elif 2000 <= co2_value <= 2999:
        return "Very Unhealthy"
    elif 3000 <= co2_value <= 4999:
        return "Hazardous"
    else:
        return "Extreme"
```

Data collection

Collect data from sensors and classify air quality status

```python
def send_sms_to_users():
    # Read the latest data from the 'data.json' file
    data = read_data_from_json()

    if data:
        # Get the most recent entry
        latest_data = data[-1]
        co2_value = latest_data.get('CO2', 0)
        temperature = latest_data.get('temperature', 0)
        humidity = latest_data.get('humidity', 0)
        timestamp = latest_data.get('timestamp', 'N/A') # defaults N/A

        # Get CO2 status
        co2_status = get_co2_status(co2_value)

        # put together the message
        message_body = (
            f"~Status~\n"
            f"Timestamp: {timestamp}\n"
            f"CO2 Level: {co2_value} PPM.\n"
            f"Temperature: {temperature}°F.\n"
            f"Humidity: {humidity}%\n."
            f"Air Quality: {co2_status}."
        )
        # Get registered phone numbers from the file
        phone_numbers = read_registered_phone_numbers()

        # Send SMS using Twilio to all registered phone numbers
        for phone_number in phone_numbers:
            try:
                client.messages.create(
                    body=message_body,
                    from_=TWILIO_PHONE_NUMBER,
                    to=phone_number
                )
                print(f"Message sent to {phone_number}")
            except Exception as e:
                print(f"Error sending message to {phone_number}: {e}")
    else:
        print("No data available to send.")
```

SMS reminder function

The system sends the latest air quality data to all registered users via the Twilio API

```python
# Setting up APScheduler to send SMS every minute
scheduler = BackgroundScheduler()
scheduler.add_job(func=send_sms_to_users, trigger="interval", minutes=1)
scheduler.start()
```

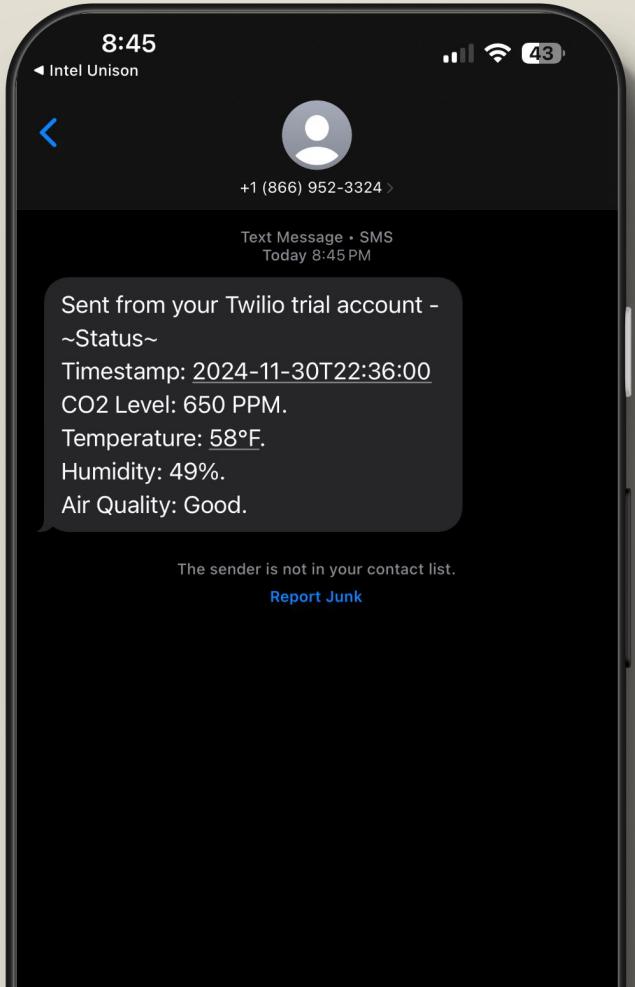
Automated SMS Scheduling

The system uses APScheduler to periodically run the SMS reminder function, checking the latest data and sending alerts every minute.

System Demonstration: Real-Time Monitoring and Alerts

Real-time data display:

SMS reminder demo:



Challenges & Solutions

PWM Mode Precision Issue

PWM mode is prone to noise interference, leading to insufficient precision in data collection

Solution: Switch to UART mode, leveraging serial communication to improve data accuracy.

Limited ESP32 Memory

The limited memory of ESP32 can lead to overflows during heavy data transmission and processing.

Solution: Optimize the code structure to reduce redundant processing and improve efficiency.

SMS Notification Failure

Network interruptions or API exceptions may cause SMS notifications to fail

Solution: Improve the error handling logic in the Twilio API implementation and add a retry mechanism.

```
    for phone_number in phone_numbers:
        try:
            client.messages.create(
                body=message_body,
                from_=TWILIO_PHONE_NUMBER,
                to=phone_number
            )
            print(f"Message sent to {phone_number}")
        except Exception as e:
            print(f"Error sending message to {phone_number}: {e}")
else:
    print("No data available to send.")
```

Summary

Successfully implemented CO_2, temperature, and humidity monitoring.

Developed a reliable SMS alert system triggered by air quality changes.

Created a Web Dashboard for real-time data visualization.

Credits

- CO2 sensor:
  - https://www.winsen-sensor.com/d/files/mh-z19c-pins%26terminal-type-co2-manual(ver1_2).pdf
  - https://www.souichi.club/en/m5stack/co2sensor-mhz19c/
- DHT11:
  - https://components101.com/sensors/dht11-temperature-sensor
- Raspberry pi - data graphic:
  - https://grafana.com/grafana/dashboards/
- Other:
  - https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/
  - https://github.com/geerlingguy/internet-pi
  - https://www.jeffgeerling.com/blog/2021/airgradient-diy-air-quality-monitor-co2-pm25
- Libraries:
- DHT11:https://github.com/dhrubasaha08/DHT11
- MHZ19C: https://github.com/WifWaf/MH-Z19