Green House Monitoring system with Power BI Integration

Project Overview:

The Greenhouse Monitoring System uses IoT sensors to track various environmental parameters within a greenhouse, including temperature, humidity, soil moisture, light intensity, and CO levels. The system sends sensor data to Power BI for real-time visualization, making it easier for greenhouse managers to monitor and take necessary actions based on the collected data.

Key Features:

1. Real-time Monitoring: Monitors key parameters such as temperature, humidity, soil moisture, light intensity, and CO levels.

2. Data Visualization: Displays real-time data on Power BI dashboards with dynamic charts, gauges, and data streams.

3. Alert System: Sends alerts (e.g., buzzer, LED) when thresholds for temperature, humidity, light, or CO levels are exceeded.

4. Data Logging: Collects and logs environmental data at regular intervals for analysis and future reference.

5. Power BI Integration: Uses Power BI's streaming dataset API to push real-time sensor data for visualization in dashboards.

6. Threshold-based Control: Based on sensor readings, control mechanisms (like water sprinklers, fans, or lights) can be automated, ensuring optimal greenhouse conditions.

Technologies Used:

1. Arduino: Used to interface with various environmental sensors (DHT11, MQ135, Soil Moisture Sensor, LDR).
2. IoT Sensors:
   - DHT11: Temperature and humidity sensor.
   - MQ135: Gas sensor for CO detection.
   - Soil Moisture Sensor: Monitors soil moisture levels.
   - LDR (Light Dependent Resistor): Measures light intensity.
3. Python: Handles communication between Arduino and Power BI. Collects data from the Arduino and pushes it to Power BI for visualization.
4. Power BI: Used for creating real-time dashboards to visualize sensor data and monitor greenhouse conditions.

5. Serial Communication (via Python): Reads the data from Arduino over the serial connection and sends it to Power BI using the streaming API.

Challenges & Solutions:

1. Real-time Data Integration:
    - Challenge: Sending real-time data from the Arduino to Power BI in a consistent and seamless manner.
    - Solution: Implemented a Python script that reads the data from the Arduino over serial communication and pushes it to Power BI's streaming dataset via API calls. Used a structured format with a timestamp to ensure data consistency.
2. Sensor Calibration and Accuracy:
    - Challenge: Some sensors, like MQ135, have variable output depending on environmental conditions.
    - Solution: Applied appropriate calibration techniques for each sensor. Tested and adjusted thresholds for soil moisture, CO levels, and temperature to ensure reliable monitoring.

3. Handling Data Latency:
    - Challenge: Latency in real-time data updates from sensors to Power BI.
    - Solution: Increased the frequency of data transmissions, ensured that the Arduino script was running efficiently, and utilized a sleep delay to avoid overloading the API or serial communication.
4. Power BI Configuration:
    - Challenge: Ensuring Power BI correctly handles the incoming data stream and visualizes it properly.
    - Solution: Carefully set up the Power BI streaming dataset with the correct data schema and ensured the data refresh interval was synchronized with the incoming data.

Key Features:

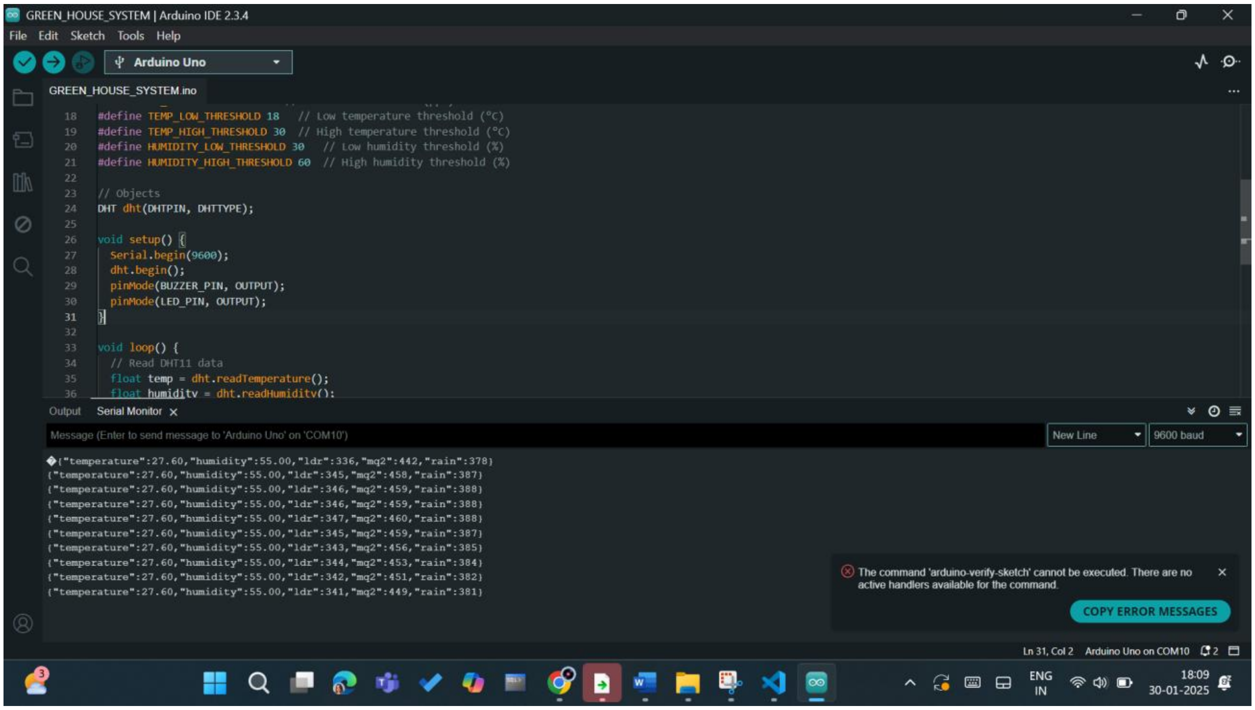
1. Real-time Dashboard in Power BI:
    - Line Chart for monitoring temperature, humidity, light intensity, and CO levels over time.
    - Gauge Visuals for soil moisture and temperature readings.

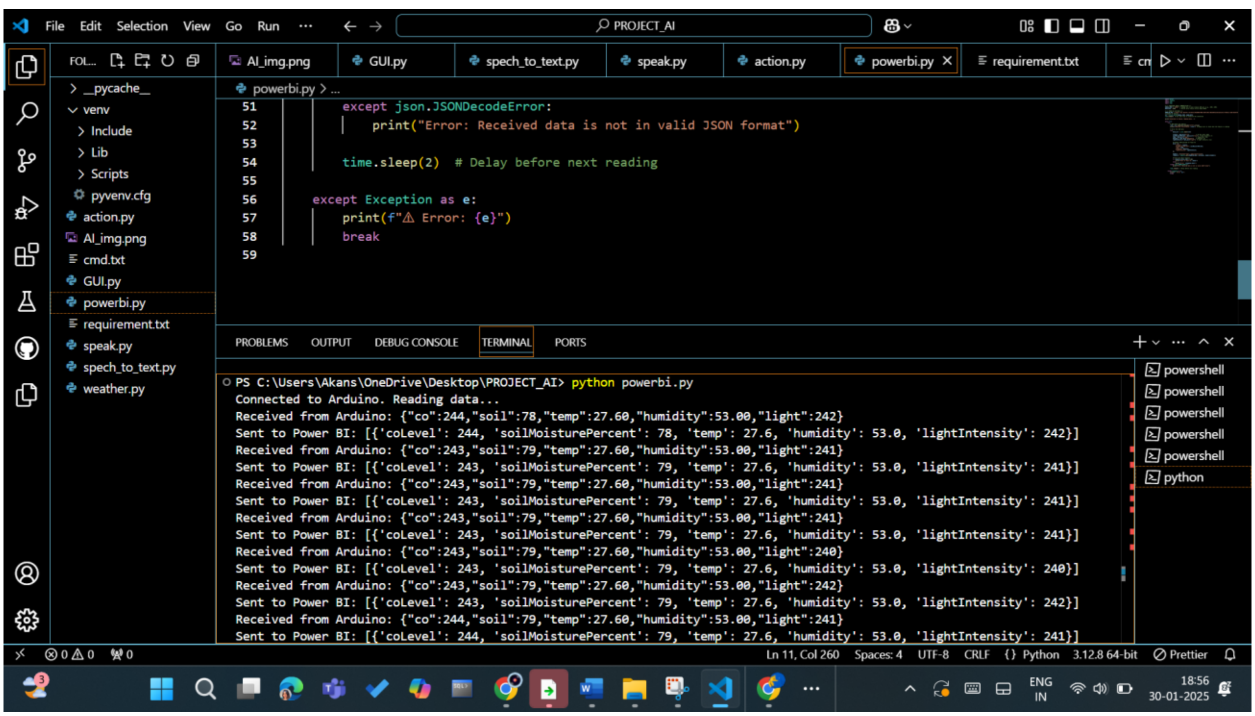
- Alert Triggers to highlight when any environmental condition exceeds set thresholds.

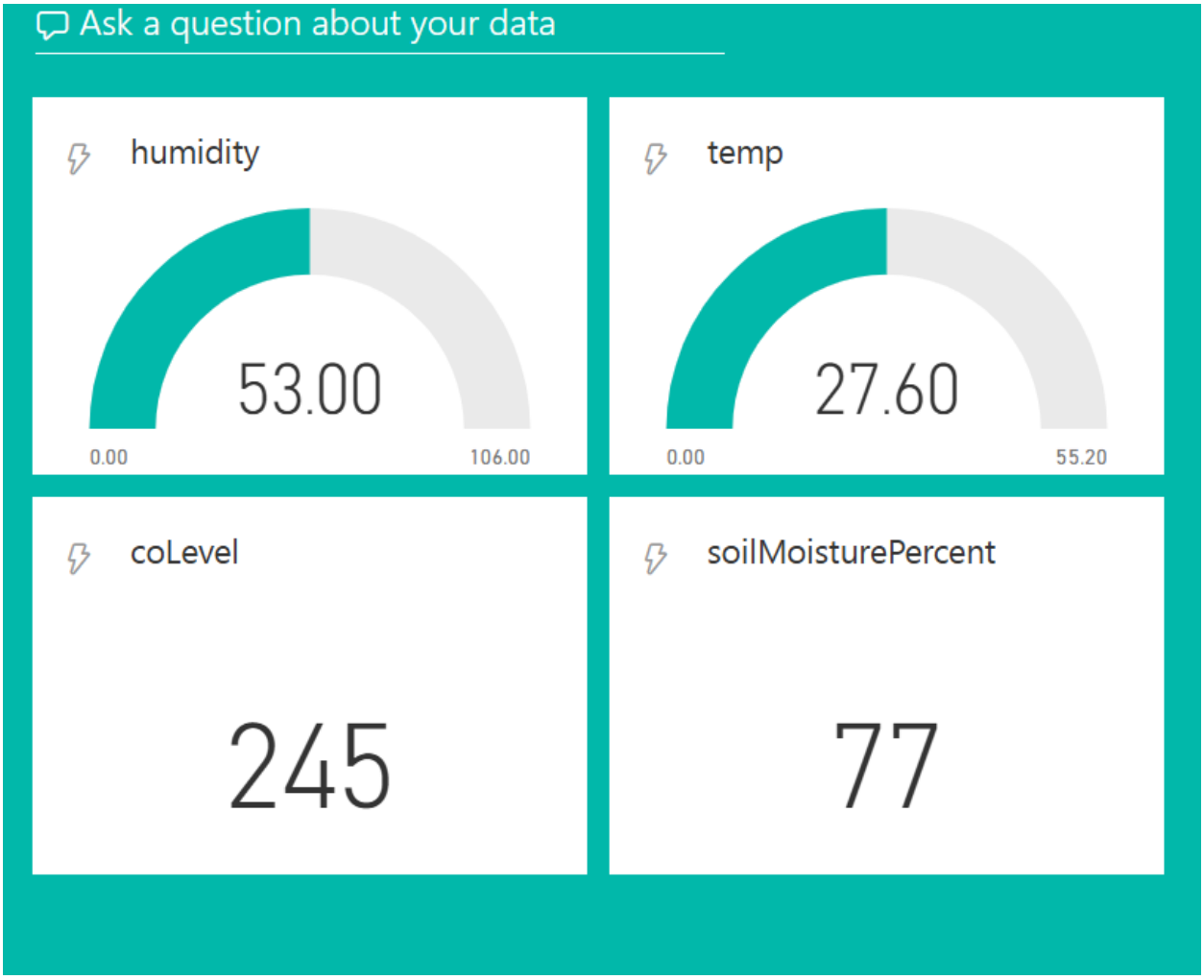
2. Threshold-Based Alerts:
   - Automated alerts when temperature, humidity, light intensity, or CO levels exceed defined thresholds, triggering control mechanisms or visual indicators.
3. Historical Data Logging:
   - Power BI stores the historical data for analysis, allowing users to track trends and make informed decisions for greenhouse operations.
4. Customizable Dashboard:
   - The dashboard is customizable with different types of charts, filters, and data displays based on the user's needs.

Project Link and Screenshot :

GitHub Repository: https://github.com/Aakanksssha/Green-House-Monitoring-system-with-Power-BI-Integration.git


GREEN_HOUSE_SYSTEM | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Uno
GREEN_HOUSE_SYSTEM.ino
#define TEMP_LOW_THRESHOLD 18 // Low temperature threshold (°C)
#define TEMP_HIGH_THRESHOLD 30 // High temperature threshold (°C)
#define HUMIDITY_LOW_THRESHOLD 30 // Low humidity threshold (%)
#define HUMIDITY_HIGH_THRESHOLD 60 // High humidity threshold (%)
// Objects
DHT dht(DHTPIN, DHTTYPE);
void setup() {
Serial.begin(9600);
dht.begin();
pinMode(BUZZER_PIN, OUTPUT);
pinMode(LED_PIN, OUTPUT);
}
void loop() {
// Read DHT11 data
float temp = dht.readTemperature();
Output Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM10')
New Line
9600 baud
{"temperature":27.60,"humidity":55.00,"ldr":336,"mq2":442,"rain":378}
{"temperature":27.60,"humidity":55.00,"ldr":345,"mq2":458,"rain":387}
{"temperature":27.60,"humidity":55.00,"ldr":346,"mq2":459,"rain":388}
{"temperature":27.60,"humidity":55.00,"ldr":346,"mq2":459,"rain":388}
{"temperature":27.60,"humidity":55.00,"ldr":347,"mq2":460,"rain":388}
{"temperature":27.60,"humidity":55.00,"ldr":345,"mq2":459,"rain":387}
{"temperature":27.60,"humidity":55.00,"ldr":343,"mq2":456,"rain":385}
{"temperature":27.60,"humidity":55.00,"ldr":344,"mq2":453,"rain":384}
{"temperature":27.60,"humidity":55.00,"ldr":342,"mq2":451,"rain":382}
{"temperature":27.60,"humidity":55.00,"ldr":341,"mq2":449,"rain":381}
The command 'arduino-verify-sketch' cannot be executed. There are no active handlers available for the command.
COPY ERROR MESSAGES
Ln 31, Col 2 Arduino Uno on COM10
ENG IN
18:09
30-01-2025


File Edit Selection View Go Run
PROJECT_AI
AI_img.png GUI.py spech_to_text.py speak.py action.py powerbi.py requirement.txt
__pycache__
venv
Include
Lib
Scripts
pyvenv.cfg
action.py
AI_img.png
cmd.txt
GUI.py
powerbi.py
requirement.txt
speak.py
spech_to_text.py
weather.py
except json.JSONDecodeError:
print("Error: Received data is not in valid JSON format")
time.sleep(2) # Delay before next reading
except Exception as e:
print(f"⚠ Error: {e}")
break
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Akans\OneDrive\Desktop\PROJECT_AI> python powerbi.py
Connected to Arduino. Reading data...
Received from Arduino: {"co":244,"soil":78,"temp":27.60,"humidity":53.00,"light":242}
Sent to Power BI: [{'coLevel': 244, 'soilMoisturePercent': 78, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 242}]
Received from Arduino: {"co":243,"soil":79,"temp":27.60,"humidity":53.00,"light":241}
Sent to Power BI: [{'coLevel': 243, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 241}]
Received from Arduino: {"co":243,"soil":79,"temp":27.60,"humidity":53.00,"light":241}
Sent to Power BI: [{'coLevel': 243, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 241}]
Received from Arduino: {"co":243,"soil":79,"temp":27.60,"humidity":53.00,"light":241}
Sent to Power BI: [{'coLevel': 243, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 241}]
Received from Arduino: {"co":243,"soil":79,"temp":27.60,"humidity":53.00,"light":240}
Sent to Power BI: [{'coLevel': 243, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 240}]
Received from Arduino: {"co":243,"soil":79,"temp":27.60,"humidity":53.00,"light":242}
Sent to Power BI: [{'coLevel': 243, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 242}]
Received from Arduino: {"co":244,"soil":79,"temp":27.60,"humidity":53.00,"light":241}
Sent to Power BI: [{'coLevel': 244, 'soilMoisturePercent': 79, 'temp': 27.6, 'humidity': 53.0, 'lightIntensity': 241}]
powershell
python
Ln 11, Col 260 Spaces: 4 UTF-8 CRLF {} Python 3.12.8 64-bit Prettier
ENG IN
18:56
30-01-2025


Ask a question about your data
humidity
53.00
0.00
106.00
temp
27.60
0.00
55.20
coLevel
245
soilMoisturePercent
77