Sorana-Elena ION (125961) 2023/05/06 16:32

Weather Station

Introduction

My project is a weather station that gathers information about temperature, humidity, and pressure, and then sends it to a server in a MySQL database. From that database, a dedicated web page can display the data for the station.

The idea for this project started with my interest in weather monitoring and the desire to create a system that could gather and display weather data in a convenient way. I believe that this project could be useful for others who are interested in weather monitoring, as well as for myself to keep track of weather conditions in my area.

The purpose of the project is to provide real-time weather information that is accurate and accessible through the internet. This information can be useful for people who want to plan their day or activities based on the weather conditions.

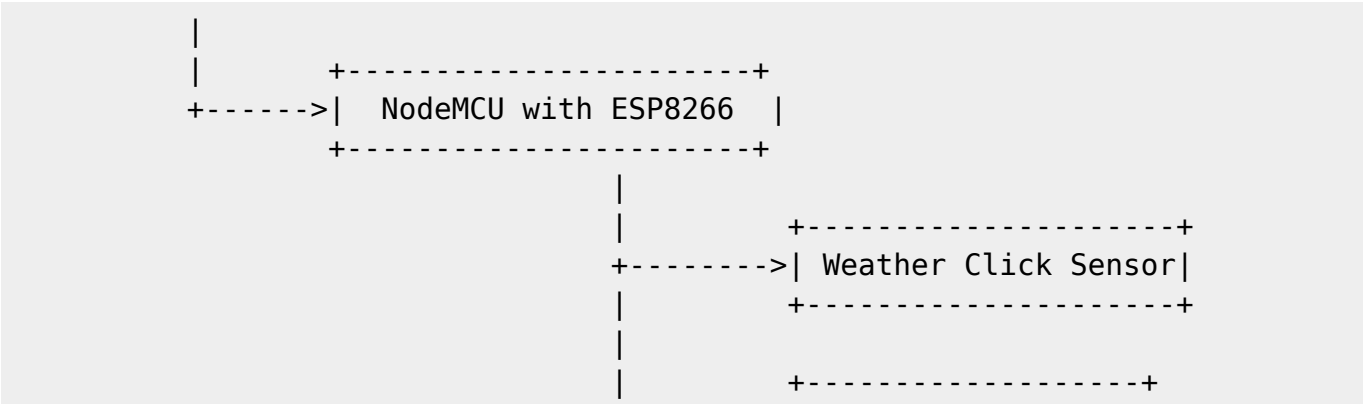
I believe that the project is useful for others because it provides accurate and real-time weather information that can help people plan their day. It can also be useful for farmers, gardeners, and other people who rely on weather conditions for their livelihoods. For me, the project is an opportunity to learn about sensors, microcontrollers, and web development while creating something that can benefit others and myself at the same time.

General Description

+————————-+

Weather Station

+————————-+



```
                                  +-------->|  LCD Display      |
                                  |         +-------------------+
                                  |
                                  |         +-------------------+
                                  +-------->|  Wi-Fi Connection|
                                  |         +-------------------+
                                  |
                                  |         +-----------------+
                                  +-------->|  MySQL Database |
                                  |         +-----------------+
                                  |
                                  |         +-----------------+
                                  +-------->| Django Admin    |
                                            +-----------------+
```

- **Weather Station**: The overall system that collects weather data and sends it to a server for storage and display.
- **NodeMCU with ESP8266**: The microcontroller that controls the system and provides wireless connectivity via Wi-Fi.
- **Weather Click Sensor**: A triple sensor that measures temperature, humidity, and pressure.
- **LCD Display**: An LCD screen that displays the weather data.
- **Wi-Fi Connection**: Provides a wireless connection to the internet via Wi-Fi.
- **MySQL Database**: Stores the weather data received from the NodeMCU with ESP8266.
- **Django Admin**: The administrative interface provided by Django web framework for managing and displaying the weather data stored in the MySQL Database. It allows administrators to access and manipulate the data, perform CRUD operations, and customize the display of weather information.

The NodeMCU with ESP8266 retrieves data from the Weather Click Sensor, which includes temperature, humidity, and pressure readings. This data is then displayed on the LCD Display. The NodeMCU with ESP8266 connects to the internet via Wi-Fi, allowing it to send the weather data to the MySQL Database for storage. The stored weather data can then be accessed and displayed on a web page.

Hardware Design

The **weather station** is composed of the following important **components**:

1. An LCD screen with 20 columns and 4 rows;
2. A triple sensor for temperature, humidity, and pressure;
3. A NodeMCU v.1 board with ESP 8266 processor;

- The LCD screen can display 40 characters on 4 lines. To send the information that needs to be displayed, I will use the I^2c protocol, which requires only 2 signal wires: SDA (for data) and SCL (for clock). Of course, for powering the screen, an additional 2 wires are needed: VCC (from 3V to 5V) and GND.
- The triple sensor from Mikroe is called Weather click and is part of a family of products with a

standardized design meant to be placed in the same way in circuits. It includes a Bosch Sensortec BME 280 chip that operates between 1.7V and 3.6V. For communication, we also use the I^2c protocol.

- The “heart” of this device is the NodeMCU v.1 board with an ESP8266 microprocessor from the RISC family based on the Tensilica Xtensa 32-bit design. It “beats” at a frequency between 80MHz and 160MHz (we can compare it to 8MHz-16MHz for the 8-bit Arduino UNO) and offers much more RAM (32KB instruction, 32KB instruction cache, 80KB user-data, 16 KB ETS system-data) and Flash (4MB). For comparison, the Arduino UNO has 2KB RAM and 32KB Flash. For my application, the most important feature is that this board has wireless connectivity at 2.4GHz on IEEE 802.11 b/g/n Wi-Fi standards. Programming this board can also be done wirelessly (OTA - over the air) without requiring direct connection to a PC, but a power source of 3V-5V (battery) will be required to provide power to it.

Software Design

Description of the Application Firmware

The application firmware is developed using the Arduino IDE, which provides support for ESP8266 microcontrollers. It leverages various third-party libraries to enable specific functionalities and streamline the development process.

Development Environment

The code is developed using the Arduino IDE, which supports ESP8266 microcontrollers.

Third-Party Libraries

- ESP8266WiFi: This library provides the necessary functions to connect to a Wi-Fi network.
- ESP8266HTTPClient: It allows making HTTP requests to send data to the server.
- WiFiClient: Provides the functionality to establish a Wi-Fi client connection.
- Wire: This library is used for I2C communication between the NodeMCU and the BME280 sensor and LCD display.
- Adafruit_BME280: Library for interacting with the BME280 sensor to read temperature, humidity, and pressure.
- LiquidCrystal_I2C: Library for controlling the LCD display using the I2C protocol.

Algorithms and Structures

The firmware code implements various algorithms and data structures to ensure efficient data processing and handling.

Sensor Data Reading:

- The firmware utilizes algorithms provided by the Adafruit_BME280 library to read temperature, humidity, and pressure values from the BME280 sensor.
- These algorithms employ specific calculations and calibration techniques to obtain accurate sensor readings.

Data Transmission:

- To send the collected sensor data to the server, the firmware implements an algorithm using the ESP8266HTTPClient library.
- This algorithm establishes an HTTP connection and crafts the necessary POST request to transmit the data.
- It ensures that the data is properly formatted and sent to the server for further processing.

Communication Protocols:

- The firmware uses the Wire library to facilitate I2C communication between the NodeMCU and the BME280 sensor and LCD display.
- I2C (Inter-Integrated Circuit) is a widely used protocol for communication between integrated circuits.
- The code leverages the I2C protocol to exchange data and commands between the microcontroller and these peripheral devices.

Database Integration:

- Although the specifics of the database implementation (MySQL) are handled outside the firmware code, certain data structuring and organization are considered.
- The firmware structures the data received from the sensor into suitable data structures (e.g., arrays, objects) to ensure efficient storage and retrieval in the database.
- Algorithms may be implemented to handle data formatting, conversion, and encoding as required by the database.

Integration with Django and MySQL

In addition to the firmware code, the project also involves integration with MySQL and Django for data storage and web application development, respectively.

MySQL: The firmware code is designed to send sensor data to a MySQL database. To achieve this, the firmware code includes functionality to establish a connection with the MySQL server and send the collected sensor data using an HTTP POST request. The server-side script, written in PHP, receives the data and stores it in a MySQL database table.

Django: On the web application side, Django is used to develop a user interface for accessing and visualizing the stored sensor data. Django is a high-level Python web framework that follows the Model-View-Controller (MVC) architectural pattern. It provides a robust set of tools and features for rapid web application development.

In the Django code, models are defined to represent the structure of the MySQL database table where the sensor data is stored. Views are implemented to handle user requests and retrieve data from the database.

The integration between the firmware code, MySQL, and Django allows for real-time data collection from the sensor device and seamless storage and presentation of the data through a web application.

By combining the firmware code with Django and MySQL, I could create a comprehensive system that collects, stores, and presents sensor data, providing valuable insights and facilitating further analysis.

Results Achieved

1. Sensor Data Acquisition:

`'Result:'` Successful acquisition of temperature, humidity, pressure, and altitude data.

`'Description'`: The weather station project successfully acquired data from the BME280 sensor, including temperature, humidity, pressure, and altitude. These readings provide valuable information for weather monitoring and analysis.

2. Display Integration:

`'Result:'` Successful integration of the LCD display to visualize sensor data.

`'Description:'` The project integrated an LCD display to visualize the acquired sensor data. The display effectively presented the temperature, humidity, pressure, and altitude readings, allowing users to easily view the current weather conditions.

3. Wi-Fi Connectivity:

`'Result:'` Successful establishment of a Wi-Fi connection for data transmission.

`'Description:'` The weather station project established a Wi-Fi connection, enabling the transmission of sensor data to a remote server. This connectivity allowed for real-time monitoring and remote access to weather information.

4. Server Communication:

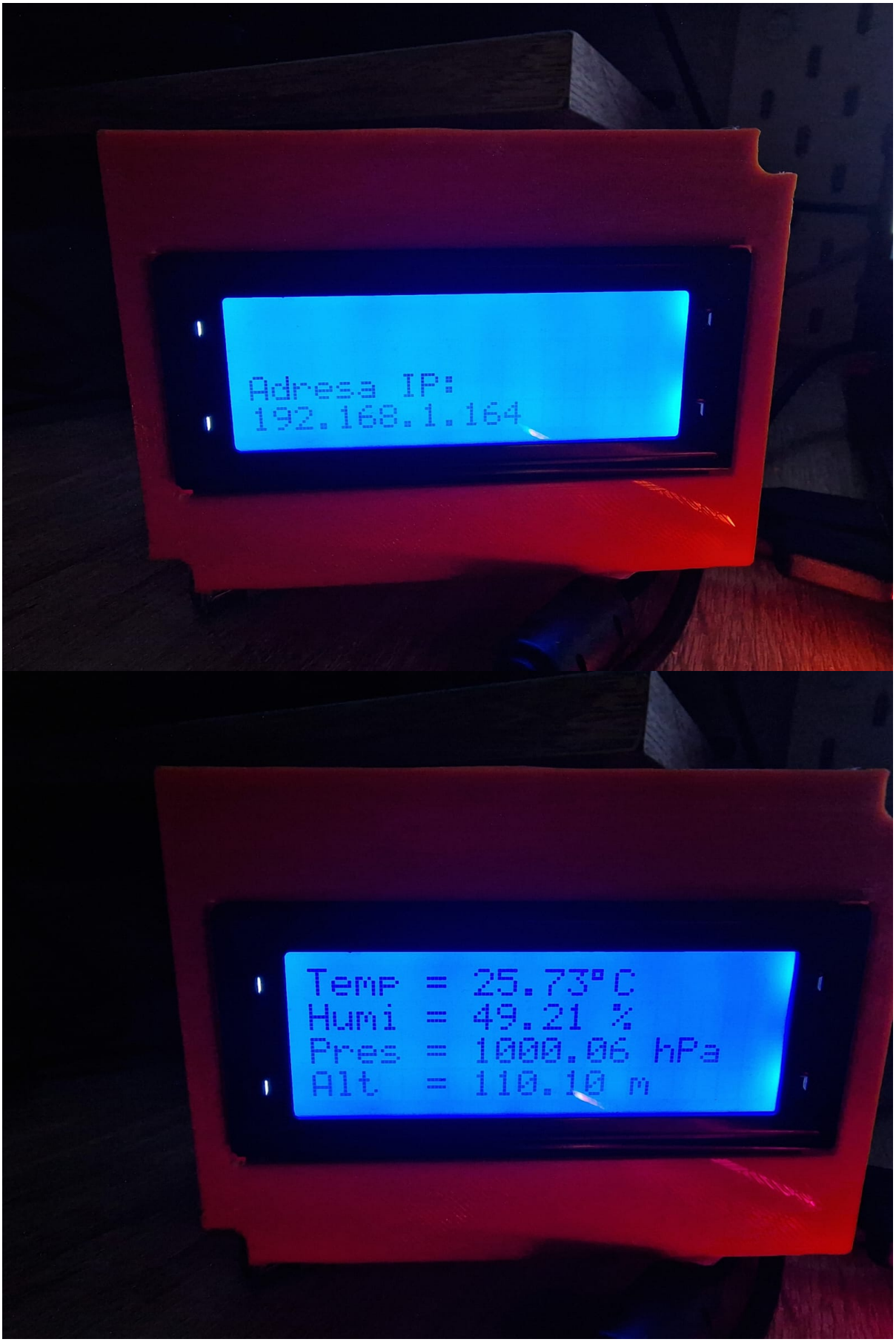
`'Result:'` Successful communication with the server to send sensor data.

`'Description:'` The project implemented functionality to send the acquired sensor data to a server. Through HTTP POST requests, the weather station transmitted the temperature, humidity, pressure, and altitude readings to the server for further processing and storage.

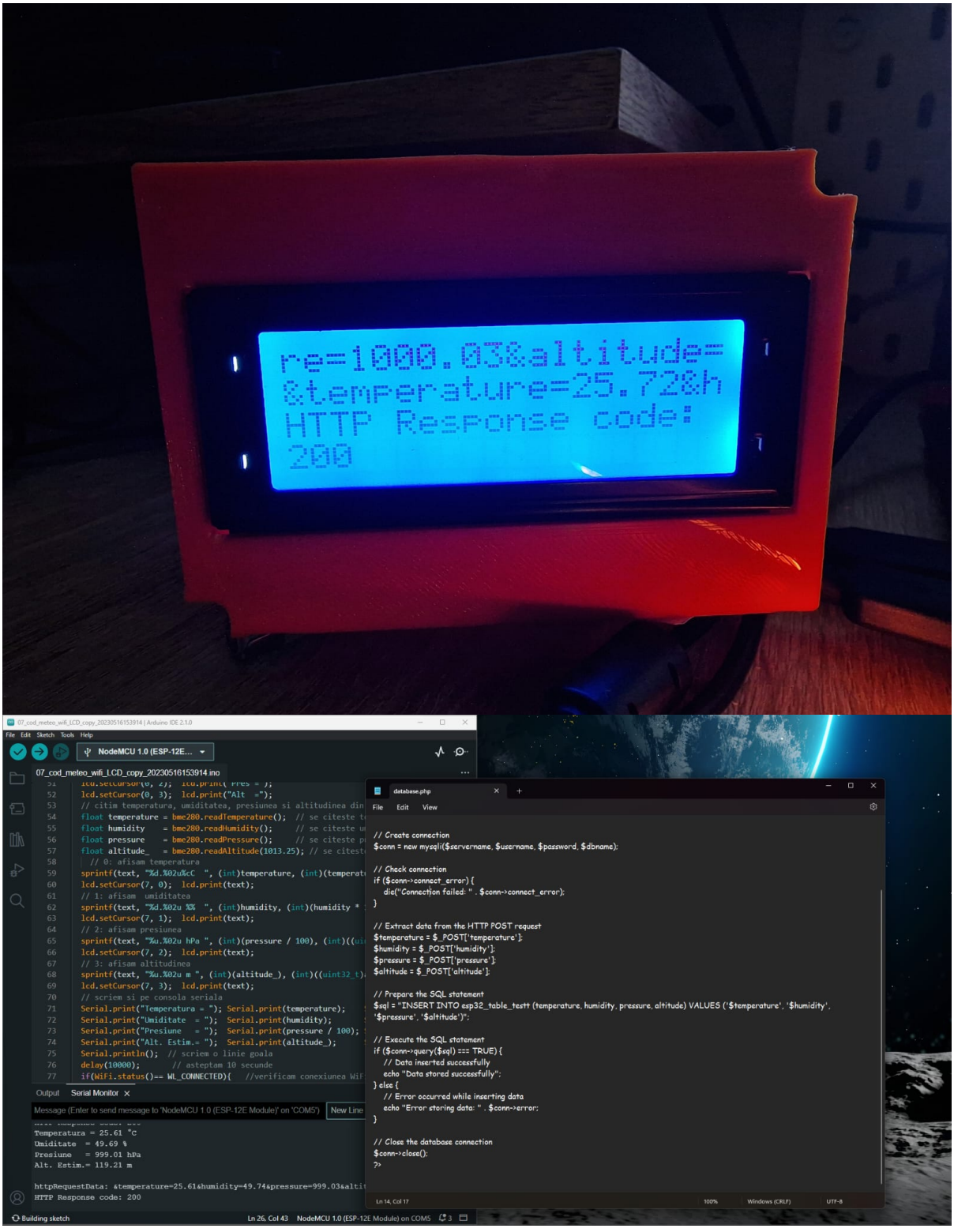
5. Data Logging and Analysis:

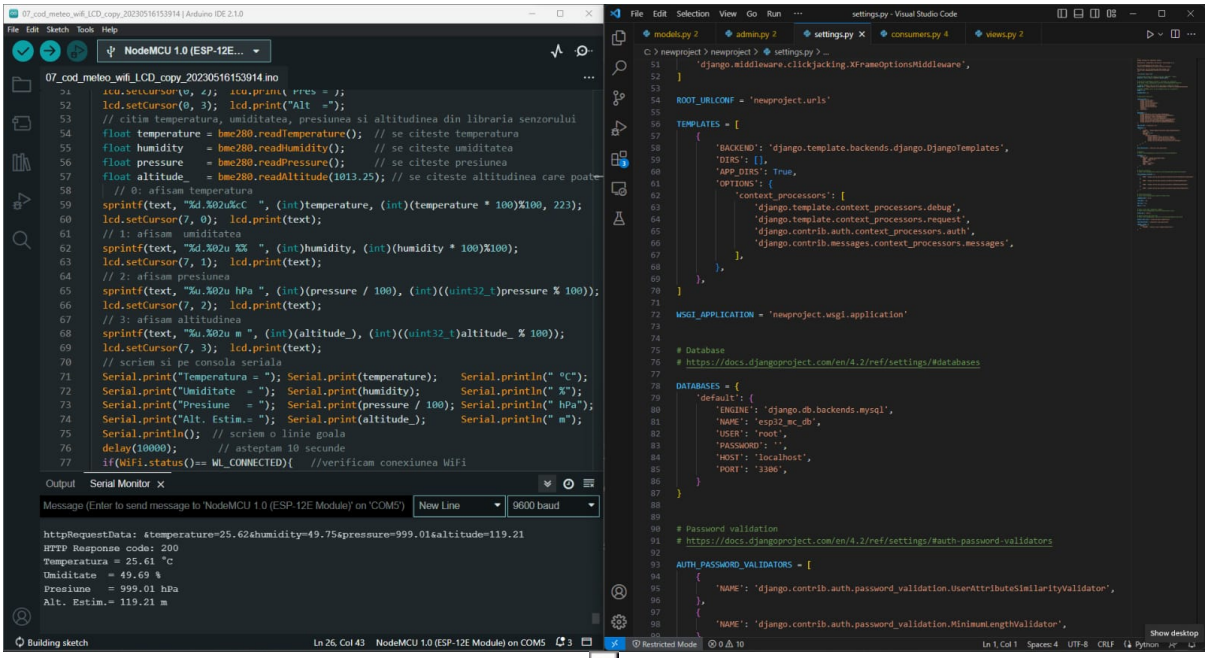
`'Result:'` Successful logging and analysis of weather data.

`'Description:'` The weather station project implemented a server-side component, likely using Django, to receive and store the transmitted sensor data.


Adresa IP:
192.168.1.164

Temp = 25.73°C
Humi = 49.21 %
Pres = 1000.06 hPa
Alt = 110.10 m


re=1000.03&altitude=
&temperature=25.72&h
HTTP Response code:
200
NodeMCU 1.0 (ESP-12E...
07_cod_meteo_wifi_LCD_copy_20230516153914.ino
Output
Serial Monitor
Temperatura = 25.61 °C
Umiditate = 49.69 %
Presiune = 999.01 hPa
Alt. Estim.= 119.21 m
httpRequestData: &temperature=25.61&humidity=49.74&pressure=999.03&alti
HTTP Response code: 200
database.php
// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
die("Connection failed: " . $conn->connect_error);
}
// Extract data from the HTTP POST request
$temperature = $_POST['temperature'];
$humidity = $_POST['humidity'];
$pressure = $_POST['pressure'];
$altitude = $_POST['altitude'];
// Prepare the SQL statement
$sql = "INSERT INTO esp32_table_testt (temperature, humidity, pressure, altitude) VALUES ('$temperature', '$humidity',
'$pressure', '$altitude')";
// Execute the SQL statement
if ($conn->query($sql) === TRUE) {
// Data inserted successfully
echo "Data stored successfully";
} else {
// Error occurred while inserting data
echo "Error storing data: " . $conn->error;
}
// Close the database connection
$conn->close();
?>



Conclusion

In conclusion, the weather station project successfully utilized the NodeMCU ESP8266 microcontroller along with various components such as the BME280 sensor, LCD display, and Wi-Fi module to create a functional and efficient weather monitoring system.

The NodeMCU acted as the central control unit, collecting temperature, humidity, pressure, and altitude data from the BME280 sensor. The acquired data was then displayed on the LCD screen, providing real-time weather information to users.

With the integration of the Wi-Fi module, the weather station enabled remote access and data transmission capabilities. Users could remotely monitor the weather conditions by accessing the data through a server or a connected mobile device.

Furthermore, the project demonstrated successful communication with the server, allowing for data

logging and analysis. This feature facilitated the storage and retrieval of weather data, enabling users to track weather patterns and generate insights for further analysis.

The weather station project showcased the power of combining hardware components and software programming to create a robust and user-friendly weather monitoring system. It provided a solid foundation for future enhancements, such as integrating additional sensors or expanding the system's capabilities through the use of advanced technologies like NFC or remote control via a Wi-Fi module.

Overall, this weather station project offered a practical solution for monitoring and analyzing weather conditions, with potential applications in various settings such as personal weather tracking, agriculture, or environmental monitoring.

Download

In addition to the comprehensive description of the weather station project, I have prepared a zip file that contains all the necessary resources for a better understanding of the project. The zip file includes:

Code: The complete Arduino code that was used to program the NodeMCU ESP8266 microcontroller and control the weather station functionality.

Printscreens: A collection of printscreens showcasing the working weather station. These images provide a visual representation of the project, including the LCD display showing real-time weather data and other relevant information.

Demo: A demonstration video that highlights the functionality and features of the weather station. This video serves as a practical demonstration of the project in action, allowing you to see how the weather data is collected, displayed, and accessed remotely.

weatherstation_project_ion_sorana_elena_1222a_fils.zip

Resources/References

Adafruit BME280 Library - Official documentation and examples for interacting with the BME280 sensor using the Adafruit library. Available at: https://github.com/adafruit/Adafruit_BME280_Library

Arduino Official Website - The official website of Arduino, providing documentation, tutorials, and forums for Arduino programming and projects. Available at: https://www.arduino.cc/

ESP8266 Arduino Core - Documentation and resources for programming ESP8266 microcontrollers using the Arduino IDE. Available at: https://github.com/esp8266/Arduino

DHT Library - Library for reading temperature and humidity values from DHT sensors. Available at:

https://github.com/adafruit/DHT-sensor-library

LiquidCrystal_I2C Library - Library for controlling LCD displays using the I2C protocol. Available at: https://github.com/johnrickman/LiquidCrystal_I2C

Arduino Project Hub - A project tutorial on creating an Arduino weather station. Available at: https://projecthub.arduino.cc/woutvdr/arduino-weather-station-9dd87f

YouTube Video Tutorial - A YouTube video tutorial demonstrating the construction and operation of an Arduino weather station. Available at: https://www.youtube.com/watch?v=RLLH5Flbm1M&t=224s

Medium Article - A Medium article providing guidance on integrating Django with an existing MySQL database. Available at: https://medium.com/@kivaimuinde/how-to-use-django-with-an-existing-database-mysql-8ff0e8446c2f

GitHub Repository - A GitHub repository containing code for connecting an ESP8266 with OpenWeatherMap (OWM) API and displaying data on an I2C LCD. Available at: https://github.com/Wunderwaffez/esp8266-owm-i2c

YouTube Video Tutorial - A YouTube video tutorial demonstrating the implementation of a weather station using Arduino and NodeMCU. Available at: https://www.youtube.com/watch?v=4b_yrbPLiLo&ab_channel=CreativeStuff

Instructables Tutorial - A detailed tutorial on how to make an Arduino weather station. Available at: https://www.instructables.com/How-to-Make-an-Arduino-Weather-Station/

Hackster.io Tutorial - A tutorial on creating a weather station using Arduino and NodeMCU. Available at: https://www.hackster.io/tarantula3/weather-station-using-arduino-and-nodemcu-d2b9d3

Export to PDF