

ESP32 CLIENT-SERVER ROOM MONITORING & WEATHER DISPLAY SYSTEM

PROJECT INTRODUCTION







This project implements a real-time room environment monitoring system using two ESP32 microcontrollers communicating via Wi-Fi on a local network. It integrates sensor data collection, wireless data transmission, time synchronization, weather data fetching from an online API, and a multi-row LED matrix display for visualization.

The Server ESP32 collects temperature, humidity, and air quality data from sensors, packages this data, and transmits it to the Client ESP32 via Wi-Fi.

The Client ESP32 receives this sensor data, displays it on a dual 8x32 LED matrix (top and bottom rows), and simultaneously fetches weather updates from an external API to display the outdoor weather conditions. It also hosts a simple web UI on the client for remote monitoring via a browser.

The design emphasizes wireless communication, real-time updates, intuitive LED matrix visualization, and internet connectivity for extended data.

COMPONENTS USED

Component	Description	Icon
ESP32 Microcontroller	Dual-core MCU with Wi-Fi & Bluetooth	
DHT22 Sensor	Temperature and Humidity sensor	
MQ-135 / Equivalent	Air Quality sensor (detects CO2, gases, etc.)	
MD_MAX72XX LED Matrix Modules	8x32 dot matrix LED displays controlled via SPI	
Power Supply	5V DC supply for ESP32 and sensors	
Wi-Fi Network	Local 2.4 GHz Wi-Fi Router	



FUNCTIONAL OVERVIEW

SERVER ESP32

- Reads the sensor data's:
- Temperature (°C)
- Humidity (%)
- Air Quality Index (AQI)
- Packages these as integer values.
- Waits for Wi-Fi connection.
- Listens for TCP client connection.
- Sends sensor data over Wi-Fi periodically.
- Manages connection reliability and reconnection logic.

CLIENT ESP32

- Connects to the same Wi-Fi network.
- Connects to the Server via TCP/IP.
- Receives sensor data and stores it locally.
- Displays sensor data on the bottom row of the dual LED matrix in a readable format all time.
- Fetches weather data for a predefined city from OpenWeatherMap API every 3 minutes.
- Displays weather info on the top row scrolling text.
- Hosts a lightweight web server accessible via browser for real-time data monitoring.
- Implements time synchronization via NTP for time-stamped operations.
- Handles reconnection to Wi-Fi or Server if connection drops.
- Monitors data timeout to restart ESP if needed.
- Alternates between scrolling weather text and static display of sensor titles.

🔍 DETAILED FEATURES

Feature	Server Side	Client Side
Sensor Data Collection	Yes (Temperature, Humidity, AQI)	No
Data Transmission	Sends TCP packets to client	Receives TCP packets
Wi-Fi Connectivity	Connects to Wi-Fi network	Connects to same Wi-Fi network
DOT Matrix Display	N/A	2 x 8x32 LED matrix modules
Weather Data Fetch	N/A	Fetches via HTTPS from OpenWeatherMap API
Time Synchronization	N/A	NTP based sync for accurate time
Web Server For Ui	N/A	Yes, shows sensor and weather data
Reconnecting Handling	Yes, manages Wi-Fi and TCP reconnect	Yes, manages Wi-Fi and TCP reconnect
Watchdog Restart	N/A	Restart if no data for 5 minutes

✅ ADVANTAGES

- **Real-time monitoring:** Sensor values update continuously and are displayed immediately, providing instant insights into environmental conditions.
- **Wireless communication:** Eliminates the need for cumbersome cables between the sensor node and the display unit, enhancing flexibility in deployment.
- **User-friendly visualization:** The dual LED matrix offers a bright, clear, and dynamic data display, ensuring information is easily digestible at a glance.
- **Internet integration:** Fetches outdoor weather data from an external API, enriching the system's utility by providing a comprehensive view of both indoor and outdoor environments.
- **Remote monitoring:** The embedded web server provides accessible real-time data monitoring from any device connected to the local area network, promoting convenience and accessibility.
- **Robust connection handling:** Automated Wi-Fi and TCP reconnection logic, coupled with a watchdog timer, ensures high system reliability and uninterrupted operation even in dynamic network environments.
- **Modular design:** The clear separation of server and client functionalities allows for independent development, testing, expansion, or replacement of each module, simplifying maintenance and future upgrades.



FUTURE SCOPE

Feature	Description	Benefits
Multiple sensor integration	Add more sensors (e.g., CO, VOC, light, noise)	More comprehensive environment monitoring
MQTT protocol support	Use MQTT for more scalable pub/sub communication	Easier multi-client support and cloud integration
Mobile app integration	Create an app to display data and control ESP remotely	Enhanced accessibility
Cloud storage & analytics	Store historic data on cloud, apply AI for pattern detection	Predictive analytics, alerts
Voice assistant control	Add Alexa or Google Assistant integration	Hands-free control
Energy efficiency	Optimize power consumption for battery or solar operation	Longer runtime, eco-friendly
Enhanced UI on webserver	Add graphs, charts, and controls on the web interface	Better user interaction



SUMMARY TABLE

Aspect	Description
Hardware	ESP32, Sensors (DHT22, MQ-135), LED Matrices
Software	Arduino framework, WiFi, HTTPClient, MD_Parola
Communication	Wi-Fi TCP socket between ESP32 devices
Display	Dual 8x32 LED matrix showing sensor and weather data
Data Update Rate	Sensor data ~100ms interval; Weather data ~3 min
Web Interface	Local network access via browser showing live data
Reliability	Auto reconnect and watchdog restart
Scope for Extensibility	Adding sensors, protocols, remote apps, cloud services

This project successfully establishes an **ESP32-based Client-Server Room Monitoring & Weather Display System**, demonstrating a robust and versatile application of IoT principles.

Hardware Foundation: The system is built upon two versatile ESP32 microcontrollers, integrated with essential environmental sensors (DHT22 for temperature/humidity, MQ-135 for air quality), and communicates with MD_MAX72XX LED matrix displays for vivid data visualization. A stable 5V DC power supply and a 2.4 GHz Wi-Fi network underpin the entire operation.

- **Software Architecture:** Developed using the Arduino framework, the software leverages WiFi and HTTPClient libraries for network communication and the MD_Parola library for efficient LED matrix control. The codebase is structured to ensure reliability and responsiveness.
- **Seamless Communication:** The core of the system's wireless capability relies on a Wi-Fi TCP socket connection established between the Server ESP32 (data source) and the Client ESP32 (data consumer and display). This ensures reliable and real-time data flow within the local network.
- **Dynamic Display:** Sensor data, including temperature, humidity, and air quality, is dynamically presented on the bottom row of a dual 8x32 LED matrix. Simultaneously, real-time weather information fetched from the OpenWeatherMap API is elegantly scrolled across the top row, offering a comprehensive environmental overview.
- **Real-time Data Updates:** The system is engineered for responsiveness, with sensor data updating approximately every 100 milliseconds. Outdoor weather conditions are refreshed every 3 minutes, balancing timely information with efficient API usage.
- **Web-Based Monitoring:** A lightweight web server hosted on the Client ESP32 allows users to access live sensor and weather data via any web browser on the local network, providing convenient remote monitoring capabilities.
- **Enhanced Reliability:** Critical to its performance, the system incorporates intelligent auto-reconnection mechanisms for both Wi-Fi and TCP connections. A watchdog timer ensures robust operation by initiating a restart if data flow ceases for a predefined duration, preventing system hangs.
- **Extensibility for Growth:** Designed with future enhancements in mind, the architecture inherently supports extensive expansion. This includes the straightforward integration of additional sensors, adoption of more advanced communication protocols like MQTT, development of dedicated mobile applications, and potential integration with cloud storage and analytics platforms.

CONCLUSION

This project stands as a compelling demonstration of a **powerful, scalable, and highly functional IoT room monitoring system**. It adeptly combines local sensor data acquisition with dynamic internet data fetching, making it an exceptionally versatile solution for diverse applications, from home automation and environmental control to educational prototypes and smart office environments.

The architectural decision to cleanly separate the data provider (Server ESP32) from the data consumer and display unit (Client ESP32) is a key strength. This modularity not only simplifies the development and debugging processes but also significantly enhances maintainability and provides a clear pathway for future upgrades and modifications without impacting the entire system.

Moreover, the successful implementation of key embedded programming and networking skills — including real-time wireless TCP communication, efficient LED matrix visualization, time synchronization via NTP, robust connection handling, and seamless integration with external web APIs — underscores a solid foundation in modern IoT development practices. This project serves as an excellent portfolio piece, showcasing a practical, real-world application of embedded systems knowledge and a strong understanding of networked device interactions. Its robust design and clear future scope position it as a valuable asset for further development in the ever-evolving landscape of the Internet of Things.

DEVELOPER INFORMATION

Name: Aniket Chowdhury [Hashtag]

Email: micro.aniket@example.com

GitHub: <https://github.com/itzzhashtag>

Instagram: https://instagram.com/itzz_hashtag

LinkedIn: <https://www.linkedin.com/in/itzz-hashtag/>