PROJECT ABSTRACT

Node-RED IoT Dashboard for Sensor Networks

Local Node-RED + MQTT stack for a wireless ESP32 sensor network: template node firmware, Mosquitto broker, flow-based dashboard with gauges and charts, threshold alerts and CSV logging — all running on a Raspberry Pi with no cloud needed. Built-to-order student kit.

01. Title

Node-RED IoT Dashboard for Sensor Networks

02. Introduction / Background

IoT coursework usually teaches the pieces in isolation: one lab flashes an ESP32, another plots a chart, a third mentions MQTT. When students then attempt a real sensor network, the pieces do not meet — readings stay trapped in serial monitors, there is no shared dashboard, no alerting and no history. Cloud platforms solve this but add accounts, subscriptions and an internet dependency. Node-RED, the open-source flow-based programming tool from the OpenJS Foundation, offers a middle path: a visual canvas where MQTT messages flow into processing nodes, dashboard widgets, alert nodes and log files, all running locally on a Raspberry Pi.

03. Problem Statement

Students can build single-sensor demos but struggle to assemble a coherent monitoring system: ingesting data from several wireless nodes, visualizing it live, raising alerts on thresholds and keeping history for analysis. Doing this with cloud services introduces cost, accounts and connectivity requirements that complicate college demonstrations. The project addresses this by delivering a complete local stack — sensor nodes, broker, integration flows, dashboard, alerting and historian — that the student deploys, configures and explains end to end.

04. Objectives

- Build ESP32 sensor nodes (DHT22, MQ-series gas, BH1750 light, capacitive soil moisture) with template firmware publishing JSON to an MQTT topic tree.
- Deploy Eclipse Mosquitto and Node-RED on a Raspberry Pi as the local broker and integration runtime.
- Implement Node-RED flows for ingestion, a live dashboard (gauges, charts, status tiles), threshold alerting via email and Telegram, and dated CSV logging.

- Make sampling intervals configurable through retained MQTT topics without reflashing.
- Deliver setup guides, flow exports, report, PPT and viva Q&A.

05. Proposed Methodology

Each ESP32 node reads its sensors on a configured interval and publishes a JSON payload to its topic. The Mosquitto broker on the Pi receives the messages over Wi-Fi. Node-RED subscribes to the topic tree; switch and function nodes route readings by node and sensor type. Dashboard nodes render gauges, charts and status tiles in the browser. Threshold function nodes evaluate limits and fire email/Telegram alerts with node ID, value and timestamp. A file node appends every reading to dated CSV logs, which chart nodes read back for historical views.

06. System Architecture / Working

Node layer: ESP32-WROOM-32 boards running the template firmware (Arduino framework), each with a unique node ID in its topic path. Broker layer: Eclipse Mosquitto (MQTT 3.1.1) on the Raspberry Pi. Integration layer: Node-RED flows — mqtt-in nodes, switch and function nodes for routing and threshold logic, email/Telegram nodes for alerts, file nodes for the CSV historian. Presentation layer: node-red-dashboard widgets (gauges, line charts, status tiles, site view) served from the Pi's address. The entire path is visible on the flow canvas, which makes the data pipeline explainable in a viva.

07. Hardware / Software Requirements

- Raspberry Pi 4 (or a laptop) as deployment host
- Node-RED on Node.js; Eclipse Mosquitto MQTT broker; node-red-dashboard widgets
- 2+ ESP32-WROOM-32 sensor nodes with template firmware
- DHT22 (±0.5 °C, ±2-5 %RH per datasheet), MQ-series gas sensor (indicative), BH1750 light sensor, capacitive soil-moisture sensor
- CSV historian (InfluxDB as documented option); email + Telegram bot alert nodes
- No internet required for core operation; internet only for cloud-backed alerts

08. Expected Outcomes

A working two-to-three-node wireless sensor network with a live Node-RED dashboard, threshold alerts delivered over email/Telegram, and a CSV history of all readings. The dashboard, flows and firmware are demonstrated live; the report documents the MQTT topic design, flow logic and a test run. Scope is a local demo deployment: the dashboard is not a hardened multi-tenant product, the single broker has no failover, and gas readings are indicative rather than calibrated.

09. Applications

- Laboratory / classroom environment monitoring
- Greenhouse and small-farm sensing demonstrations
- Teaching aid for MQTT, flow-based programming and IoT system design
- Building / workshop condition-monitoring demos

10. Future Scope

- InfluxDB + Grafana historian for richer time-series analytics
- Authentication and TLS hardening of broker and dashboard
- LoRa sensor nodes for long-range deployments
- Cloud bridge for remote access alongside the local stack