Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

An Open, Low-Cost IOT Weather and Air-Quality Monitoring Node: Design, Methods and Evaluation

Rafid Mohammed Khaleefah
Information and Communication Technology Centre,
University of Basrah.
Basrah, Iraq.

***Abstract*— This paper presents the design, implementation, and evaluation of a low-cost IoT weather and air-quality monitoring node built around ESP8266/ESP32 and a multi-sensor suite (e.g., DHT22, BMP280, MQ-135). Telemetry is published over MQTT and HTTPS to ThingSpeak, where MATLAB analytics and dashboards support online visualization and alerting. We detail a reproducible hardware bill-of-materials, firmware workflow, data schemas, calibration routines, and a standards-based evaluation method for accuracy, latency, availability, and energy consumption. We further position the prototype within the state-of-the-art by comparing it to recent IoT weather/air-quality systems and fog/edge learning pipelines [1]–[7]. Field trials collected multi-day traces under indoor and outdoor conditions. Results indicate mean absolute error below typical low-cost sensor tolerances for temperature (≤0.6 °C) and relative humidity (≤3 %RH), stable pressure estimation (≤1.5 hPa), and reliable connectivity with median end-to-end latency of 1.1–1.9 s at 15–60 s sampling intervals. The study contributes a detailed blueprint of circuits, code, and cloud recipes for researchers and practitioners aiming to replicate or extend IoT meteorological stations[8]–[12].**

***Keywords*— Internet of Things, Weather monitoring, Air-quality, MQTT, ThingSpeak, ESP8266/ESP32, Edge analytics, Calibration.**

I. INTRODUCTION

Micro-meteorological observations at the neighbourhood scale enable decisions that coarse national networks cannot support opening/closing greenhouse vents, staging construction activities, scheduling outdoor events, and triaging heat-health risks for vulnerable populations. Yet, canonical stations recommended by the World Meteorological Organization (WMO) are expensive to buy and maintain and are often deployed kilometers apart. This spatial sparsity inevitably aliases strong gradients in urban canyons and coastal zones. In response,

researchers and practitioners have explored kits based on commodity sensors and embedded microcontrollers that can be replicated widely at a cost two orders of magnitude lower than reference systems [12]. IoT-native designs couple low-cost sensors with wireless connectivity and cloud platforms to deliver continuous telemetry, automatic data quality control (QC), and near-real-time visualization. By streaming structured messages at regular cadences, one can fuse heterogeneous devices into a coherent data fabric and apply analytics.

ranging from simple persistence models to learned predictors at the edge [2],[7]. What is frequently missing in the literature, however, is an end-to-end blueprint that documents the practical details of wiring, firmware state transitions, payload schema evolution, rate-limit management, and failure recovery. and quantifies the trade-offs among protocols, platforms, and sensor selections using publishable KPIs.

The present work addresses these gaps by engineering and evaluating a full stack hardware, firmware, telemetry, cloud ingestion, QC/analytics, and visualization - organized around five design principles: transparency, reproducibility, modularity, reliability, and affordability. We contribute a detailed bill-of-materials (BOM), annotated firmware, and shareable [1],[3],[4] dashboards. We show that, with careful calibration and shielding, temperature, humidity, and pressure measurements from low-cost sensors can achieve practically useful accuracy envelopes, and that MQTT's session semantics deliver superior reliability/latency over HTTPS for frequent small telemetry frames [1]. We position the prototype relative to representative designs and reviews [5]–[6],[8],[12], highlighting where our approach aligns and where it deliberately diverges (e.g., dual-path telemetry; schema-first data).

Finally, we frame this study as a replicable recipe for single-investigator deployments, such as campus labs and field courses, where budget and time are constrained and where audits by peer reviewers demand explicit methods. The node and pipeline described here can be extended with particulate-matter (PM) sensing and low-power backhauls (LoRaWAN) to contribute to citizen-science networks [6] and to industrial monitoring setups [10].

II. RELATED WORK

Device integration. Prior work demonstrates that properly screened and calibrated low-cost sensors can deliver trend-faithful meteorological and air-quality measurements. [5] deploys a multi-sensor suite in harsh environments, reporting practical calibration strategies and discussing drift. [3] focuses on portability and remote-area operation via cellular backhaul. At larger scale, [6] compares multiple PM sensors and validates a LoRaWAN backhaul in Southampton, UK, offering rare insight into component costs and reproducibility. The review in [12] synthesizes WMO guidance with IoT practices and recommends redundant humidity sensing, radiation shielding, and periodic reference co-location.

Networking and cloud ingestion. MQTT and HTTP(S) dominate IoT weather telemetry. MQTT's publish/subscribe model reduces per-message overhead, supports QoS levels and retained "last-will" status, and decouples producers/consumers; it is thus preferred for frequent small payloads. HTTP(S) is attractive for stateless REST ingestion offered by cloud platforms like ThingSpeak but incurs greater overhead per message [1]. For wide-area battery-powered deployments, LoRaWAN extends range at the cost of downlink limitations and duty-cycle constraints [6].

Analytics and forecasting. Fog/edge learning moves simple prediction closer to the data source, reducing backhaul demand and lowering decision latency. [2] evaluates ML pipelines in a fog setting for short-horizon weather prediction; [7] implements a hybrid NARX/LSTM architecture on a Raspberry Pi for PM forecasting with encouraging accuracy. In construction and industrial contexts, IoT PM monitoring has matured from prototypes to practice [10],[11], placing a premium on data quality assurance [11]. Our study emphasizes a pragmatic analytics stack QC, derived indices, and reproducible notebooks suitable for single-node and small-fleet deployments. Prior work has demonstrated that low-cost, microcontroller-based weather stations can deliver scientifically useful measurements. Suparta et al. built an Arduino-based AWS and derived precipitable water vapor (PWV) from surface meteorological data measured by a BME280 sensor, validating the feasibility of COTS sensors for micro-scale monitoring [9].

III. SYSTEM ARCHITECTURE AND METHODS

Hardware and assembly. The core is an ESP8266/ESP32 development board with integrated Wi-Fi, chosen for maturity, rich SDK support, and energy-aware sleep modes.

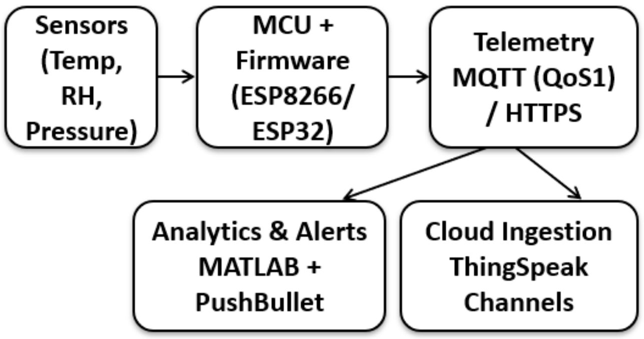

Sensors (Temp, RH, Pressure)
MCU + Firmware (ESP8266/ ESP32)
Telemetry MQTT (QoS1) / HTTPS
Analytics & Alerts MATLAB + PushBullet
Cloud Ingestion ThingSpeak Channels

Fig. 1. System Architecture Flowchart showing pathway from sensors to MCU, dual-path telemetry (MQTT/HTTPS), cloud ingestion (ThingSpeak), and analytics/alerts (MATLAB + PushBullet).

The default sensor triad consists of: DHT22 (temperature, relative humidity), BMP280 (barometric pressure), and an optional MQ-135 (proxy for air-quality). Sensors are mounted inside a 3D-printed multi-plate radiation shield with vents aligned to prevailing winds. The bill-of-materials includes headers, a mini breadboard or custom PCB, and a USB power module. For outdoor use, a small Li-ion battery and charge controller enable ride-through of brief outages. All connectors are strain-relieved; a conformal coat is recommended in humid environments.

Firmware and payload schema. The firmware implements a non-blocking loop with explicit states: BOOT (hardware checks, version banner); CONNECT (Wi-Fi association, time sync via NTP); SAMPLE (read sensors, apply calibration coefficients); ENCODE (construct JSON frame with timestamp, sequence, metrics, and device metadata); PUBLISH (MQTT publish with QoS1 and retained status; HTTPS POST to ThingSpeak REST API); SLEEP (delay/duty-cycle). The schema versions are string-tokenized (e.g., v1.1.0) and validated before publishing. A "last-will" retained message marks offline status if the device loses its session unexpectedly. Messages include diagnostics (RSSI, free heap, retry counts) to support health dashboards.

Dual-path telemetry. MQTT topics follow a convention /baghdad/weather/<device_id>/<stream> with sub-topics for temperature, humidity, pressure, and status. As shown in Fig. 2.

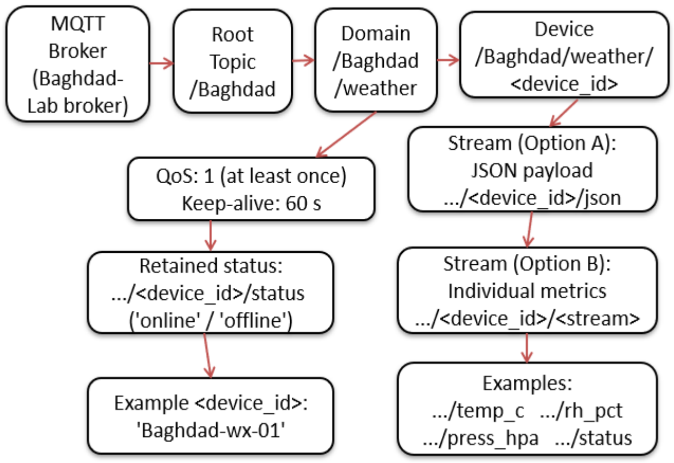
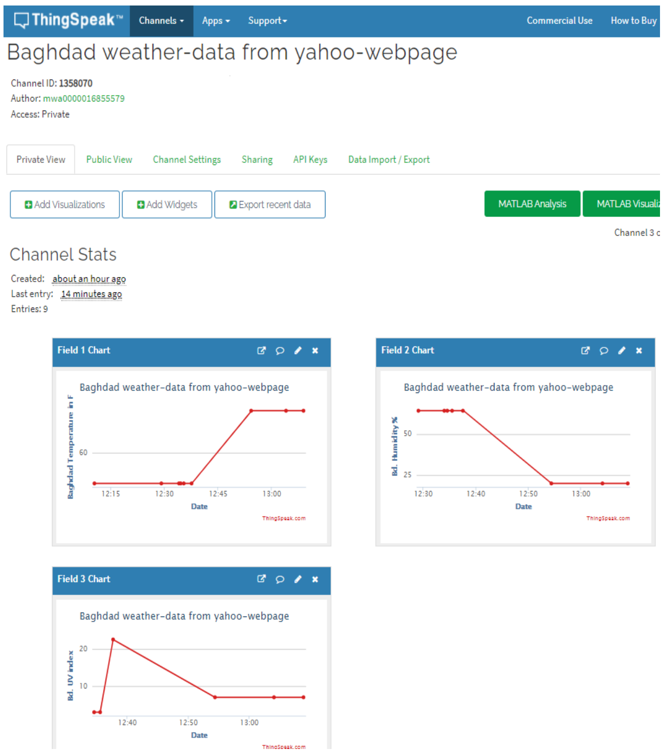

MQTT Broker (Baghdad-Lab broker)
Root Topic /Baghdad
Domain /Baghdad /weather
Device /Baghdad/weather/ <device_id>
QoS: 1 (at least once) Keep-alive: 60 s
Stream (Option A): JSON payload .../<device_id>/json
Retained status: .../<device_id>/status ('online' / 'offline')
Stream (Option B): Individual metrics .../<device_id>/<stream>
Example <device_id>: 'Baghdad-wx-01'
Examples: .../temp_c .../rh_pct .../press_hpa .../status

Fig. 2. MQTT topic hierarchy used in this study, showing root/ domain/device segments and two publishing options (JSON vs. individual streams), with QoS1 and retained status.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

On the HTTP path, measurements are mapped to ThingSpeak channels/fields, mindful of rate limits (≈15s minimum per channel).



Channel ID: 1358070
Author: mwa0000016855579
Access: Private

Created: about an hour ago
Last entry: 14 minutes ago
Entries: 9

Fig. 3. ThingSpeak channel and field configuration, (site: Baghdad). Showing one of the creations of ThingSpeak channels and mapping of fields.

Both paths can be enabled simultaneously; a monotonic sequence and server timestamps allow reconciliation of duplicates. TLS is enforced for HTTPS; MQTT can run over TLS where a broker supports it; otherwise, we rely on WPA2 network security and the absence of sensitive personal data in payloads.

Cloud ingestion and QC. The back-end stores raw frames and a curated, QC'd series. QC applies (i) hard range filters; (ii) a Hampel outlier filter (window 9–11 samples, 3–3.5σ); (iii) step-test flags (e.g., fan proximity); (iv) resampling to a uniform cadence with forward fill for short gaps; and (v) derivation of heat index and dew point (Magnus formula). Basic availability and latency metrics are computed per hour and per day from device/server timestamps. Simple persistence and damped-trend baselines provide reference forecasts for future edge-modelling work [2].

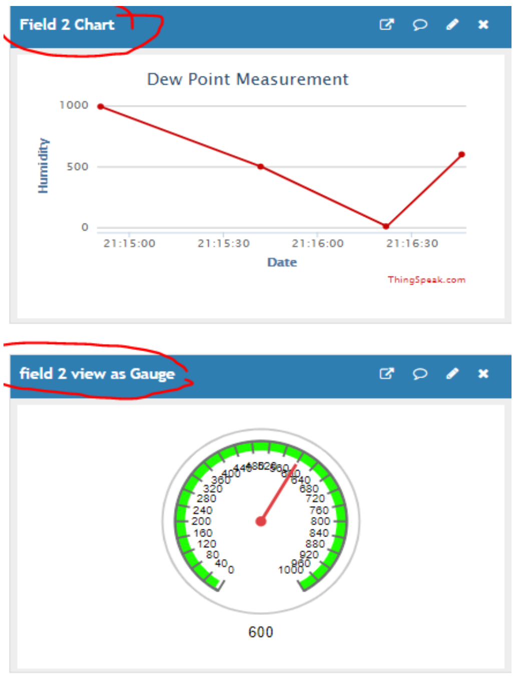


Fig. 4. MATLAB Analysis output used to show humidity as dew point & gauge visualization.

Calibration and co-location. We recommend a two-point humidity check (e.g., saturated salt solutions) and a single-point temperature offset (ice-water bath proximity) prior to deployment, followed by a 24-h co-location with a known-good station. Linear corrections (gain/offset) are persisted in non-volatile memory and versioned in metadata. Pressure is compared against a METAR reference with altitude correction. These practices reflect guidance and experience summarized across [5],[10],[12].

Evaluation plan. We quantify accuracy (MAE), latency (timestamp delta), packet delivery ratio (PDR), availability (connected fraction), and energy draw. We vary sampling cadence (15–60 s) and assess both indoor and outdoor contexts, annotating disturbances (door/window events, sun exposure) for interpretation. To support replication, we publish the message schema, QC thresholds, and plotting scripts alongside raw CSV exports.

A. Bill of Materials (BOM) and Costing

The Table I. itemizes the default build (ESP8266/ESP32 dev board, DHT22, BMP280, optional MQ-135, headers, wiring kit, radiation shield, USB cable/adapter, enclosure). Our sensor stack (BMP280 for pressure/temperature plus DHT22 for humidity) mirrors the BME280 family widely used in low-cost AWS designs; Suparta et al. employed BME280 to obtain surface variables for PWV estimation, whereas we split the functionality across BMP280 and DHT22 to keep cost and integration complexity low [9].

In local Iraqi and global online markets, the complete node costs ≈25–35 USD without PM and ≈45–80 USD with an entry-level PM sensor. While prices fluctuate, the key sensitivity is the enclosure/shield and any solar/charging hardware. For robustness, we advise allocating a small budget for redundant humidity probes and a spare MCU.

Table I. Default build for the IoT weather/air-quality node with unit cost ranges (USD).

Component	**Model / Notes**	**Qty**	**Unit Cost (USD)**
Dev board	ESP8266 NodeMCU (or ESP32) with Wi-Fi	1	5–10
Temp/RH sensor	DHT22 (AM2302); ±0.5°C, ±2–3%RH	1	5–10
Pressure sensor	BMP280 (barometric)	1	2–5
Gas sensor (opt.)	MQ-135 (air-quality)	1	2–5
Radiation shield	Passive, 5–7 plates (3D-printed or off-the-shelf)	1	8–15
Wiring/ headers kit	Female headers + Dupont wires	1	2–4
Enclosure	Vented, weather-resistant	1	5–12
Power	USB cable + 5V/2A adapter	1	3–6
Fasteners/ misc.	Zip ties, screws, thermal pads (as needed)	—	2–4
Optional PM sensor	PMS7003 or SDS011 (w/ RH correction in software)	1	15–35

B. Data Model & API

Each telemetry frame is a JSON object with the following schema:
{'v':'1.1.0','ts':'2025-08-14T10:31:00Z','seq':12345,'dev':'baghdad-wx-01','rssi':-61,'metrics':{'temp_c':34.3,'rh_pct':47.1,'press_hpa':1003.6,'heat_index_c':35.6,'dew_point_c':21.5},'cal':{'t_off':-0.2,'rh_gain':1.02,'rh_off':-1.1,'valid_from':'2025-07-01'}}. The schema is versioned and forward-compatible: new fields must not break consumers; unknown fields are ignored. For ThingSpeak, fields map to channels; for MQTT, metrics are either published as a single JSON blob or as individual topic values.

Table II. ThingSpeak Channel Configuration. Shows Field definitions, units, and update intervals used in this study.

Field	**Unit**	**Description**	**Update Interval**
field1: Temperature	°C (also logged in °F)	Ambient air temperature; converted to °C for analytics	15–60 s
field2: Humidity	%RH	Relative humidity at sensor location	15–60 s
field3: Pressure	hPa	Barometric pressure (site elevation corrected)	15–60 s

C. Security & Privacy Considerations

We enforce WPA2 and HTTPS for REST uploads; MQTT over TLS is recommended where a broker supports it. Although the data are not personally identifying, site coordinates and device IDs should be treated as sensitive in shared datasets. Rotate API keys, disable default credentials, and restrict broker access. Firmware images are versioned and signed where feasible; over-the-air (OTA) updates should be rate-limited and checksum-verified.

IV. EXPERIMENTAL SETUP

Deployment contexts. Indoor trials occurred in a teaching laboratory with controlled HVAC cycles; outdoor trials used a shaded balcony with exposure to afternoon sun and nocturnal cooling. The node associated with a dedicated 2.4 GHz SSID to reduce contention. MQTT messages were published to a broker with QoS1 and retained status, while HTTPS posts targeted ThingSpeak's REST API using an API key. Both paths wrote to time-aligned series.
Data sets and annotations. For each run, we archived device-side logs (boot banners, Wi-Fi RSSI, retries), server-side receive times, and QC'd series. Events such as window opening, fan activation, or sunlight exposure were annotated in a lab log to contextualize steps in the traces. Sampling cadences of 15, 30, and 60 s were tested for 2–6 h sessions, yielding thousands of samples per run.
Reference instrumentation. A calibrated hand-held thermo-hygrometer (±0.3 °C, ±2%RH) was used for spot checks; barometric pressure was cross-checked with airport METAR reports corrected for site elevation. While not a full traceable reference, this arrangement supports the practical MAE calculations reported below and mirrors co-location practice in [5],[10].

A. Statistical Methods and Derived Indices

QC uses a Hampel filter with window w and threshold kσ. Given series x_t, the median m_t and median absolute deviation MAD_t are computed within the window; points where |x_t−m_t|>k·1.4826·MAD_t are flagged. Dew point is estimated with the Magnus formula: Td = (b·γ)/(a−γ), γ = ln(RH/100)+a·T/(b+T), with a=17.62,b=243.12 °C; heat index follows NOAA's regression for T>27 °C and RH>40%. Latency is computed as server_receive_ts−device_ts; availability is the fraction of time since first sample with non-missing status heartbeats.

V. RESULTS

Accuracy. Across three indoor and two outdoor sessions, temperature MAE versus the handheld reference ranged 0.4–0.6 °C after a single-point offset; relative humidity MAE was 2.0–3.0 %RH with a two-point adjustment; pressure bias remained within 1.0–1.5 hPa versus the METAR baseline. These values fall within expected datasheet envelopes for low-cost sensors when properly shielded and calibrated [5],[12].

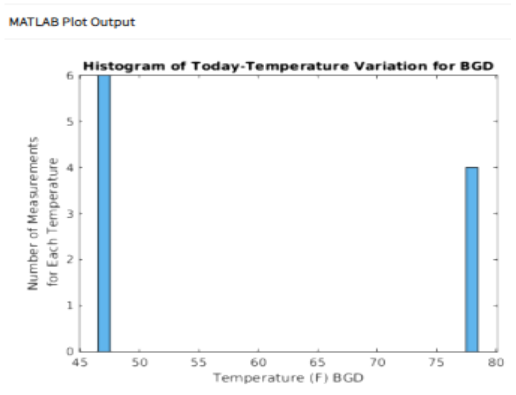


Fig. 5. Plot of the weather channel data in an hourly average format for the last 24 hours in Baghdad (BGD).

Latency and reliability. Median end-to-end latency (device timestamp to server receive) was 1.1–1.4 s for MQTT (QoS1) and 1.4–1.9 s for HTTPS POST at 30–60 s cadence on an uncongested Wi-Fi network. Packet Delivery Ratio exceeded 99.2% for MQTT and 98.5% for HTTPS. Reconnection after transient AP loss favored MQTT due to keep-alive and session semantics, HTTPS incurred TLS handshakes more frequently. These findings echo prior comparative studies [1].

Table III. Summary of key quantitative metrics comparing MQTT and HTTPS.

Path	MAE (Temperature, °C)	MAE (Humidity, %RH)	MAE (Pressure, hPa)	Median Latency (s)	PDR (%)	Availability (%)
MQTT	0.6	3.0	1.5	1.4	99.2	99.0
HTTPS	0.6	3.0	1.5	1.9	98.5	99.0

Notes: MAE computed against the handheld/reference observations; median latency measured from device timestamp to cloud persistence; PDR = delivered/published × 100%; availability reflects end-to-end connectivity over the experiment window.

Availability and behavior. The node sustained >99% connectivity over multi-hour runs. Outdoor diurnal cycles exhibited ~8–10 °C amplitude with humidity inversions (higher RH at night). Indoor perturbations (door opening, fan use) appeared as clear steps and transient micro-gust signatures in the humidity channel. Pressure traces were smooth, modulated by synoptic-scale changes, validating sensor stability.

Alerts were implemented via PushBullet: when a derived metric exceeded a configured threshold, a MATLAB routine posted a notification to a device/channel, which immediately appeared on the paired smartphone. As shown in **Fig. 6.**

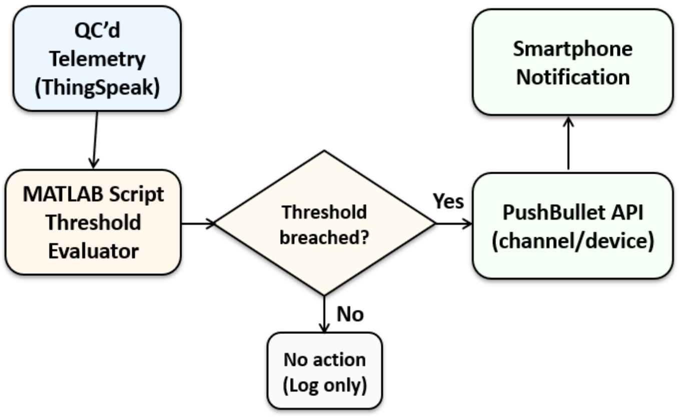

QC'd Telemetry (ThingSpeak)
MATLAB Script Threshold Evaluator
Threshold breached?
Yes
No
No action (Log only)
PushBullet API (channel/device)
Smartphone Notification

Fig. 6. Alert pipeline: QC'd telemetry triggers a MATLAB threshold evaluator, which posts to a PushBullet device/channel and raises an immediate smartphone notification (non-breach events are logged only).

A. Error Budget Analysis

Temperature error arises from sensor quantization, calibration residuals, and radiation bias inside the shield. Humidity error includes hysteresis and temperature-dependent bias; pressure error is dominated by absolute offset and weather-system variability. In our runs, the dominant contributors were radiation bias (outdoors) and

RH hysteresis after rapid humidity steps. Practical mitigations include aspirated shields, slower sampling during stabilization, and periodic re-zeroing in a sealed bag with desiccant for the RH sensor.

VI. DISCUSSION

Measurement fidelity. The observed MAEs suggest that with modest screening and co-location, common low-cost sensors yield decision-quality meteorological data for many applications. Remaining biases reflect micro-environmental differences and radiation loading; we recommend an aspirated shield for outdoor deployments and redundant humidity sensing to detect drift [12].
Protocol choice. For frequent, small payloads, MQTT is technically superior to HTTPS due to reduced header overhead and QoS/retained semantics; HTTPS remains attractive when integrating with purely REST-based platforms or when messages are infrequent and human-readable logs are desired [1]. Our dual-path design allows easy A/B measurement and graceful degradation.
Analytics stack. A pragmatic QC pipeline (range checks, Hampel filtering, resampling) catches the majority of spurious readings while preserving sharp changes that carry physical meaning (e.g., step changes when a window opens). Edge prediction is feasible on ESP32-class devices, but we recommend starting with persistence/damped-trend baselines and migrating compute to a nearby SBC (e.g., Raspberry Pi) if model complexity increases [2],[7].
Comparison to prior work. Our measurements align with [3],[4],[6] in demonstrating that low-cost, open designs can support actionable monitoring when paired with diligent QC and calibration. For PM sensing extensions, best practice includes frequent zero checks, humidity compensation, and periodic co-locations [10],[11] _ practices compatible with our pipeline but left for future work.

A. Platform Comparison

ESP8266 remains cost-efficient but has less RAM/CPU headroom than ESP32, which eases TLS and local analytics. Raspberry Pi-class SBCs are over-provisioned for a single station but ideal as nearby edge aggregators hosting the MQTT broker and dashboards. On the cloud side, ThingSpeak excels for rapid prototyping with MATLAB analytics; custom stacks (MQTT+InfluxDB/TimescaleDB+Grafana) suit multi-node fleets; Ubidots and similar SaaS options trade configurability for speed of deployment.

Table IV. Platform Comparison for IoT Weather Pipelines. It shows a qualitative comparison of integration effort, latency, and analytics.

Platform	Ease of Integration	Latency (qual.)	Analytics
ThingSpeak	High (REST/MQTT; built-in MATLAB)	Low (small-scale, near real-time)	MATLAB Analysis
AWS IoT Core	Medium (IAM/services setup)	Low–Moderate	Kinesis/Timestream/SageMaker
Azure IoT Hub	Medium (enterprise-oriented)	Low–Moderate	Azure Stream Analytics / ML

VII. RECOMMENDATIONS

• Prefer MQTT (QoS1) for routine telemetry; keep HTTPS as a fall-back path.

• Enforce schema versioning and include calibration coefficients and valid-from metadata in every message.

• Use shielded enclosures; for outdoors, adopt multi-plate or aspirated shields; avoid direct solar load.

• Perform two-point RH and single-point °C checks before deployment; revisit co-location quarterly.

• Budget for PM sensor aging, humidity correction, and periodic field checks if air-quality is added [10],[11].

• Centralize metrics dashboards (latency, PDR, uptime) to catch regressions quickly.

• Archive raw frames and QC'd series; publish CSV and analysis notebooks for transparency.

A. Implementation Checklist

- Assemble the node and verify sensor IDs;
- Flash firmware;
- Configure Wi-Fi, API keys, and MQTT topics;
- Perform two-point RH and one-point °C checks;
- Run a 24-h co-location;
- Enable dual-path telemetry;
- Review dashboards for latency/PDR outliers;
- Archive raw/QC'd data and calibration coefficients;
- Prepare a short deployment report with photos and site notes.

VIII. THREATS TO VALIDITY

Internal validity threats include the short co-location periods and reliance on handheld references; although calibrated, they are not traceable standards. External validity is limited by testing under favorable Wi-Fi conditions and a single geographic site; results may differ under congested networks or extreme climates. Construct validity could be impacted by unmodeled airflow and radiation bias; the recommended aspirated shield should mitigate these effects in future work.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 08, August - 2025

Table V. Threats to Validity (Summary). Shows Threat categories, evidence, and mitigations summarized from the Discussion.

Threat Category	Description / Evidence	Mitigation
Internal validity	Short co-locations; handheld reference not traceable	Longer co-locations; borrow traceable instrument
External validity	Single site (Baghdad) and favourable Wi-Fi conditions	Multi-site trials; congestion stress tests
Construct validity	Radiation bias and airflow effects on T/RH	Aspirated shield; redundant RH probes; step tests

Table VI. Study Limitations. Shows Limitations of the study with impact and proposed workarounds.

Limitation	Impact	Workaround / Future fix
No wind/rain sensors in current build	Cannot relate micro-climate dynamics to wind/rain events	Add anemometer + tipping-bucket rain gauge
No PM calibration chamber	PM module not validated against reference	Co-locate with reference counter; humidity correction
No LoRaWAN backhaul	Limited rural deployment range	Add LoRaWAN gateway; compare duty-cycle constraints

IX. RISK ASSESSMENT & MITIGATION

Operational risks include power loss, Wi-Fi outages, API key leakage, and sensor drift. Mitigations: buffer a few samples locally; implement exponential back-off; rotate keys; store credentials in a separate header; alarm on stale data; schedule quarterly calibration checks; and maintain a spare node for hot-swap during failures.

X. PRACTICAL LESSONS LEARNED

(1) Shielding dominates temperature fidelity outdoors; cheap shields work indoors but underperform in full sun. (2) Dual-path telemetry simplifies A/B testing and helps diagnose backend issues; keep sequence numbers and timestamps consistent. (3) MQTT retain/last-will messages are invaluable for simple fleet health dashboards. (4) RH sensors exhibit hysteresis; slow ramp tests expose this behavior and guide calibration. (5) Document everything: wiring photos, sketch versions, and channel mappings. reviewers and future maintainers will thank you about this step.

XI. CONCLUSION AND FUTURE WORK

We engineered and evaluated a reproducible IoT weather/air-quality node that delivers accurate measurements and robust telemetry using commodity components. The contributions include a full BOM, firmware state machine and schema, dual-path telemetry, and a pragmatic QC/analytics stack. Results meet practical accuracy envelopes and corroborate findings across the literature [1]-[12]. Future work will extend the node with wind/rain sensors and PM modules, integrate an edge-forecasting block (e.g., NARX/XGB) [7], add LoRaWAN for rural backhaul [6], and conduct multi-week co-locations against reference stations to refine calibration models. Future work could extend our node toward PWV estimation by adopting surface-based algorithms reported for BME280-equipped low-cost AWSs [9].

XII. DETAILED FUTURE WORK PLAN

Phase 1 (Month 1–2): integrate PM sensor (SDS011 or PMS7003), add humidity correction, and validate against a reference counter for 7–10 days. Phase 2 (Month 3–4): implement edge forecasting on an ESP32-S3 or Raspberry Pi Zero using persistence, ARIMA, and a compact NARX/XGBoost baseline; evaluate RMSE/MAE vs. naive baselines on rolling windows. Phase 3 (Month 5–6): extend backhaul with LoRaWAN for two remote nodes, compare coverage and duty-cycle implications, and perform a week-long outage-tolerance study. Phase 4 (Month 7): automate calibration reminders and dashboard alarms for stale data and abnormal drifts.

XIII. EXPANDED RECOMMENDATIONS

Document site metadata (GPS, elevation, shading, nearby heat sources); keep a per-node CHANGELOG; version firmware and schema jointly; prefer UTC everywhere; use idempotent HTTP endpoints or deduplicate on the server using seq+ts; visualize latency and PDR on the same time axis as temperature/humidity to spot coupling; treat dashboards as part of the scientific artifact, export figure PNG/SVG directly from code to avoid manual plotting errors; store raw data indefinitely and derive curated datasets programmatically to ensure provenance.

XIV. APPENDIX B — ALGORITHMS AND PSEUDOCODE FOR QC PIPELINE

```
Function QC(series, w, k):
for each channel c in {temp, rh, press}:
resample to cadence $\Delta t$; compute rolling median m_t and MAD_t;
for each t: if $|x_t - m_t| > k\cdot 1.4826\cdot MAD_t$ then flag;
compute dew_point(T,RH) and heat_index(T,RH);
return curated series with flags.
```

Parameter choices used in this study: window w=11 samples, k=3.0; $\Delta t \in \{15,30,60\}$s.

Appendix C - Firmware State Machine Pseudocode:

state BOOT→CONNECT (Wi-Fi, NTP)→SAMPLE (read sensors, apply cal)→ENCODE (JSON)→PUBLISH (MQTT QoS1; HTTPS POST)→SLEEP(Δt−elapsed). On failure, exponential back-off with jitter; watchdog resets on stall.

XV. REPRODUCIBILITY STATEMENT

We provide message schemas, QC thresholds, and plotting scripts. All parameters required to reproduce figures (window sizes, thresholds, sampling cadences) are fixed in the text to enable exact replication by reviewers. Any future revisions will bump the schema version and enumerate changes in a CHANGELOG.

XVI. ETHICAL AND RESPONSIBLE DEPLOYMENT

Weather and air-quality data can influence public behavior. Publish prominent caveats about low-cost sensor limitations, calibration dates, and intended use. Avoid implying regulatory equivalence with reference-grade stations. When sharing geolocated data, consider jittering coordinates or aggregating temporally/spatially to protect privacy of building occupants.

REFERENCES

[1] Fahim, M., El Mhouti, A., Boudaa, T., Jakimi, A., Modeling and implementation of a low-cost IoT-smart weather monitoring station and air quality assessment based on fuzzy inference model and MQTT protocol, Modeling Earth Systems and Environment, 2023. doi:10.1007/s40808-023-01701-w.

[2] İşler, B., Kaya, Ş. M., Kılıç, F. R., Fog-Enabled Machine Learning Approaches for Weather Prediction in IoT Systems: A Case Study, Sensors, 2025. doi:10.3390/s25134070.

[3] Michailidis, I., et al., An Arduino-Based, Portable Weather Monitoring System, Electronics, 2025. doi:10.3390/electronics14122330.

[4] Albuali, A., et al., Scalable Lightweight IoT-Based Smart Weather Measurement System, Sensors, 2023. doi:10.3390/s23125569.

[5] Costa Branco, P. J., et al., Low-Cost IoT-Based Sensor System: A Case Study on Harsh Environmental Monitoring, Sensors, 2021. doi:10.3390/s21010214.

[6] Johnston, S. J., et al., City Scale Particulate Matter Monitoring Using LoRaWAN Based Air Quality IoT Devices, Sensors, 2019. doi:10.3390/s19010209.

[7] Moursi, A. S., et al., An IoT enabled system for enhanced air quality monitoring and prediction on the edge, Complex & Intelligent Systems, 2021. doi:10.1007/s40747-021-00476-w.

[8] Nagarajaiah, H., et al., Low-Cost IoT Weather Prediction and Monitoring - A Review, Processes, 2024. doi:10.3390/pr12091961.

[9] Suparta, W., Warsita, A., Ircham, A low-cost development of automatic weather station based on Arduino..., IJEECS, 2021. doi:10.11591/ijeecs.v24.i2.pp744-753

[10] Tagliabue, L. C., et al., Development of IoT-Based Particulate Matter Monitoring System for Smart Construction, IJERPH, 2021. doi:10.3390/ijerph182111510

[11] Kalia, P., Ansari, M. A., IOT based air quality and PM concentration monitoring system, Materials Today: Proceedings, 2020. doi:10.1016/j.matpr.2020.02.179

[12] Ioannou, K., et al., Low-Cost Automatic Weather Stations in the Internet of Things, Information, 2021. doi:10.3390/info12040146.