e-ISSN: 2614-8404
p-ISSN:2776-3234

Deployment of a Machine Learning Based Indoor Air Quality Predictor on NodeMCU ESP32 Using EloquentTinyML

Yaddarabullah[1*)], Umar Al Faruq[2]
[1]Department of Informatics, Universitas Trilogi
[2]Department of Information System, Universitas Trilogi
Email: [1]yaddarabullah@trilogi.ac.id, [2]faruq@trilogi.ac.id

Abstract Indoor air quality monitoring increasingly requires local intelligence because indoor exposure conditions can change faster than centralized systems can respond. However, many machine learning based indoor air quality predictors remain difficult to deploy on microcontrollers due to memory limits, computation constraints, network dependence, and the lack of a reproducible edge deployment workflow. This study develops an end-to-end TinyML framework for deploying a lightweight indoor air quality predictor on a NodeMCU ESP32 using EloquentTinyML and TensorFlow Lite Micro. A synthetic IAQ dataset was generated from eight environmental variables, namely temperature, humidity, PM2.5, PM10, NO2, SO2, CO, and room type, and the indoor air quality score was derived from pollutant weighted features before balancing comfort labels using SMOTE. The proposed compact MLP contains eight inputs, one hidden layer with twelve neurons, ReLU activation, dropout regularization, and a single regression output. Five-fold validation produced an average root mean square error of 5.794, mean absolute error of 4.394, and R^2 of 0.877, while the converted model required only 121 trainable parameters. These results indicate that compact TinyML deployment can provide a feasible proof-of-concept for local indoor air quality estimation, although physical sensor validation remains necessary.

Keywords - *Indoor air quality, TinyML, NodeMCU ESP32, EloquentTinyML, TensorFlow Lite Micro, embedded machine learning, SMOTE*

I. INTRODUCTION

Indoor air quality (IAQ) is a critical component of healthy learning spaces, offices, laboratories, and public facilities because indoor exposure to particulate matter, gaseous pollutants, temperature, and humidity directly affects comfort, cognitive performance, respiratory risk, and the perceived safety of occupants. As indoor environments are increasingly instrumented with low-cost sensors, IAQ monitoring is shifting from occasional measurement to continuous observation, enabling data-driven estimation of air condition and air comfort states at finer temporal resolution [1]-[3].

Despite this progress, IAQ prediction systems face a fundamental system constraint when they are moved from laboratory computers or cloud platforms to embedded devices. Microcontrollers such as the NodeMCU ESP32 provide affordability, portability, and wireless connectivity, but they have limited RAM, flash memory, computational throughput, and power budget. Therefore, an IAQ predictor that performs well on a computer may not be directly suitable for on-device inference unless the model is compact, the runtime is compatible with microcontroller execution, and the deployment pipeline is reproducible.

Recent literature has addressed different parts of this problem. Low-cost IAQ sensing studies emphasize sensor calibration, bias correction, and data reliability in indoor or urban environments [2], [3], [15]. IoT-based air quality studies generally focus on data acquisition, dashboard visualization, or server-side analytics, whereas TinyML studies demonstrate that neural networks can be compressed and executed on highly constrained devices using TensorFlow Lite Micro or similar runtimes [4]-[6]. In parallel, embedded AI platforms such as Edge Impulse and MLPerf Tiny have shown that systematic benchmarking, model conversion, and deployment automation are important for bringing machine learning closer to physical sensing devices [11], [12].

However, existing methods still show several limitations for practical IAQ deployment. Cloud-based prediction introduces network dependence, possible latency, and additional communication energy. High-capacity machine learning models such as ensemble regressors can produce accurate estimates but are often less convenient to convert into a microcontroller-native inference graph. Conversely, extremely small models can be deployed more easily but may lose predictive flexibility when sensor patterns become nonlinear or when environmental context needs to be included in the input representation.

The research gap addressed in this study is therefore not only the prediction of IAQ score, but the construction of a complete and transparent path from IAQ data generation, preprocessing, model training, model conversion, C-array embedding, and firmware-level inference on NodeMCU ESP32. This gap is particularly relevant for educational and early-stage research settings where physical sensor data may not yet be available, but a reproducible proof-of-concept is needed before field instrumentation and long-term validation are conducted.

This study positions a compact multilayer perceptron (MLP) as a TinyML-oriented regression model. The MLP is selected not because it is expected to outperform every baseline on a synthetic formula-based dataset, but because it can be represented as a TensorFlow Lite Micro model, embedded into Arduino firmware, and executed through EloquentTinyML while still retaining nonlinear modeling capacity for future real sensor data. This positioning makes the proposed method a

e-ISSN: 2614-8404
p-ISSN:2776-3234

deployment framework rather than merely an offline regression experiment.

The proposed framework integrates synthetic IAQ dataset generation, comfort-label balancing using SMOTE, feature standardization, compact MLP training, cross-validation, TensorFlow Lite conversion, C-array model embedding, and serial-monitor feedback on predicted IAQ, comfort level, inference time, free heap memory, and estimated CPU usage. The main contributions are an end-to-end TinyML deployment workflow for IAQ prediction, an analysis of the accuracy-efficiency trade-off between compact and larger models, an ablation study of preprocessing and input components, and an embedded implementation structure that can be extended to calibrated physical sensors.

The significance of this work lies in its role as a bridge between IAQ machine learning and microcontroller-based environmental intelligence. By demonstrating what can and cannot be claimed from a synthetic proof-of-concept, the study provides a careful foundation for future field deployment, sensor calibration, and long-term indoor exposure monitoring. The rest of this paper is organized as follows. Section II describes the methodology, dataset, preprocessing, model, training configuration, and evaluation metrics. Section III presents the results. Section IV discusses the findings, novelty, practical meaning, and limitations. Section V concludes the paper and outlines future work.

II. RESEARCH METHODOLOGY

A. Flowchart of Methodology

The methodology was designed as a traceable engineering pipeline rather than a single offline prediction experiment. Figure 1 summarizes the workflow from controlled dataset generation to embedded deployment. Each step has a direct implementation artifact: a synthetic dataset, a preprocessed training matrix, a compact model, cross-validation metrics, a TensorFlow Lite file, C-array model files, and Arduino firmware for on-device inference.

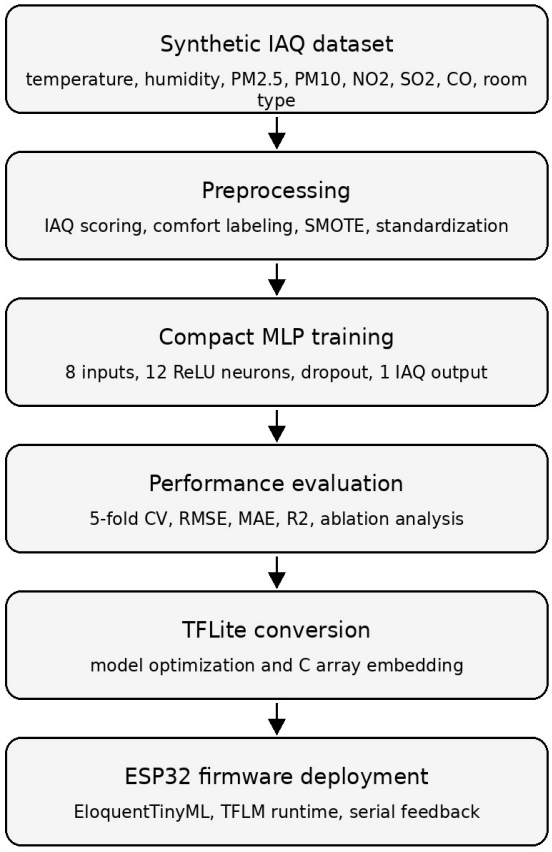

Figure. 1 Methodology flowchart of the indoor air quality TinyML deployment pipeline.

B. Dataset

The dataset was generated synthetically to reproduce the structure of an IAQ sensing problem before physical sensors are installed. The input variables were temperature, relative humidity, PM2.5, PM10, NO2, SO2, CO, and room type. The first two variables represent thermal comfort, five variables represent particulate and gaseous pollutants, and room type represents a simple spatial-context variable. The IAQ score was computed using a pollutant-weighted rule and clipped to the interval between 0 and 100 as shown in Equation (1).

$$IAQ = \text{clip}(100 - (0.5\,PM2.5 + 0.3\,PM10 + 0.7\,NO2 + 3\,CO), 0, 100) \quad (1)$$

Table 1. Descriptive characteristics of the synthetic IAQ dataset

Feature	Mean	Std.	Min.	Max.
temp	26.135	2.965	16.276	37.558
hum	49.936	10.054	19.805	89.262
PM2.5	29.280	14.995	-17.651	76.694
PM10	64.866	31.003	-45.651	170.872
NO2	22.025	7.908	-9.379	49.019
SO2	6.923	2.978	-3.127	17.133
CO	1.529	0.800	-0.966	4.243
room_type	2.500	1.141	1.000	4.000
IAQ	45.895	13.469	4.052	86.362

Table 1 shows that the synthetic generator creates a wide but controlled environmental space. The pollutant variables have sufficient variance to produce a broad IAQ range, while the temperature and humidity variables preserve the comfort context used in the labeling rule. Some generated pollutant values are negative because unconstrained normal distributions were used

e-ISSN: 2614-8404
p-ISSN:2776-3234

in the original material; this is acceptable for pipeline testing but should be replaced by bounded or sensor-calibrated distributions in field validation.

C. Preprocessing

Preprocessing consisted of comfort label construction, class balancing, feature scaling, and train-test partitioning through cross-validation. Comfort labels were generated from the combination of IAQ score, temperature, and humidity. Class 0 represents very comfortable air, class 1 comfortable air, class 2 slightly uncomfortable air, and class 3 uncomfortable air. Because the deterministic rule produced highly imbalanced classes, SMOTE was applied to generate synthetic minority samples in the feature space before regression training [9].

Table 2. Comfort-label distribution before and after SMOTE

Class	Meaning	Before	After
0	Very comfortable	7	1103
1	Comfortable	167	1103
2	Slightly uncomfortable	723	1103
3	Uncomfortable	1103	1103

Table 2 indicates that the original dataset was dominated by the uncomfortable class, while the very comfortable class was extremely rare. SMOTE equalized all comfort labels to 1,103 samples each. The technical implication is that the model is trained on a more balanced representation of comfort states, although the synthetic minority samples must later be checked against realistic sensor behavior.

D. Proposed Model

The proposed model is a compact MLP designed for TensorFlow Lite Micro compatibility. It accepts eight environmental inputs and produces one continuous IAQ score. The hidden layer contains twelve neurons with ReLU activation, followed by dropout regularization in the Keras implementation and a linear output neuron. This architecture intentionally prioritizes a small memory footprint and simple inference graph over maximum offline accuracy.

Table 3 clarifies the main architectural advantage of the proposed model. With only 121 parameters, the model can be stored as a compact C array and executed with a small tensor arena. The trade-off is that the hidden representation is intentionally narrow, which can reduce accuracy compared with larger neural networks or ensemble models on the training computer.

Table 3. Proposed compact MLP architecture

Layer	Configuration	Output
Input layer	8 environmental variables	8
Dense hidden layer	12 neurons, ReLU	12
Dropout	Rate 0.1 during training	12
Output layer	1 linear neuron for IAQ regression	1
Trainable parameters	Approximately 121 weights and biases	-

E. Training Configuration

Training was performed using five-fold cross-validation with shuffled folds and a fixed random seed to improve reproducibility. StandardScaler was used before MLP training so that features with different physical units did not dominate the optimization process. For comparison, several conventional regressors were evaluated on the same balanced dataset, including linear regression, random forest, gradient boosting, XGBoost, a compact MLP, and a larger MLP. The comparison is used to interpret accuracy-efficiency trade-offs rather than to claim that the smallest model is always the most accurate model.

Table 4. Training and deployment configuration

Component	Configuration
Validation strategy	5-fold cross-validation, shuffled, random_state = 42
Training target	Continuous IAQ score
Primary model	Compact MLP with 12 hidden neurons
Optimizer/loss	Adam optimizer and MSE loss in the original Keras workflow
Feature scaling	StandardScaler fitted within each training fold
Deployment conversion	Keras model to TensorFlow Lite, then C array
Runtime target	EloquentTinyML with TensorFlow Lite Micro on ESP32

F. Performance Evaluation

Model accuracy was evaluated using RMSE, MAE, and coefficient of determination. RMSE emphasizes larger prediction errors, MAE provides an interpretable average absolute deviation in IAQ-score units, and R2 measures the proportion of target variance explained by the model. For the embedded stage, the firmware design also reports predicted IAQ, comfort level, inference time, free heap memory, and estimated CPU usage through the serial monitor. In this paper, embedded metrics are treated as implementation outputs because the uploaded material still uses simulated sensor input rather than calibrated physical sensors.

III. RESULTS

A. Dataset Characteristics

The synthetic dataset contains 2,000 initial samples and 4,412 samples after SMOTE balancing. The key dataset characteristic is the strong imbalance produced by the comfort rule: only seven initial samples belonged to the very comfortable class, while 1,103 samples belonged to the uncomfortable class. This pattern is technically meaningful because IAQ comfort rules often create rare boundary conditions, and a model trained without balancing tends to underrepresent those rare conditions.

The IAQ target had a mean of 45.895, a standard deviation of 13.469, and a range from 4.052 to 86.362. This distribution shows that the generated environment is dominated by moderate to low IAQ values rather than very clean air. In the research context, this makes the dataset useful for testing whether a lightweight model can estimate degraded air conditions, but it also shows why field validation is required before the system can be used for health-related decisions.

Table 5 should be interpreted with the characteristics of the synthetic target in mind. Linear regression achieved nearly perfect reconstruction because the IAQ score was generated from a linear pollutant-weighted formula with clipping. This result is an internal consistency check for the synthetic generator rather than evidence that a linear model will be sufficient for real indoor environments. Tree ensembles and boosting models

e-ISSN: 2614-8404
p-ISSN:2776-3234

produced strong accuracy, but they are less straightforward to deploy through TensorFlow Lite Micro and EloquentTinyML than a compact neural network.

B. Model Performance Comparison

Table 5. Model performance comparison

Model	RMSE	MAE	R2	Complexity	TinyML readiness
Linear regression	0.000	0.000	1.000	9	Very high
Random forest	2.254	1.459	0.981	Tree ensemble	Low
Gradient boosting	2.064	1.484	0.984	Tree ensemble	Low
XGBoost	2.405	1.769	0.979	Tree ensemble	Low
Larger MLP, 32-16 neurons	0.301	0.157	1.000	833	Medium
Compact MLP, 12 neurons (proposed)	5.794	4.394	0.877	121	High

The proposed compact MLP produced an average RMSE of 5.794, MAE of 4.394, and R2 of 0.877. Its main advantage is not absolute dominance in offline accuracy, but the balance between acceptable regression performance, small parameter count, nonlinear capacity, and direct compatibility with the TFLite-based deployment path. The larger MLP achieved substantially higher accuracy, but it increased parameter count from 121 to about 833, illustrating the trade-off between predictive capacity and embedded simplicity.

C. Analysis of Key Advantage of the Proposed Model

Table 6 links the quantitative model design to the practical IAQ context. A model with 121 parameters is small enough for embedded experimentation, while the output structure remains useful for occupants because the IAQ score is translated into a comfort label. This design also supports future extension: once calibrated sensor data are available, the same deployment path can be reused with updated training data and a revised model.

Table 6. Key advantages of the proposed model

Aspect	Finding	Technical implication
Model compactness	121 parameters	Small memory footprint and simple C-array storage
Deployment path	Keras to TFLite to model_data.cc/h	Traceable conversion from Python to Arduino
Runtime compatibility	EloquentTinyML and TFLM	Microcontroller-oriented inference interface
Output interpretability	IAQ score and comfort label	Combines numerical prediction and human-readable feedback
Extensibility	Eight environmental inputs	Can be connected to real IAQ sensors in future work

D. Ablation Study Results

Table 7 provides a deeper explanation of how preprocessing and input selection affect the compact model. Removing SMOTE reduced R2 from 0.877 to 0.608, showing that balancing the comfort-conditioned samples helps the model learn across a broader IAQ-comfort space. Removing feature scaling also degraded performance, confirming that optimization becomes less stable when variables with different units and magnitudes are provided directly to a small MLP.

Table 7. Ablation study

Configuration	RMSE	MAE	R2
Full compact MLP with SMOTE and scaling	5.794	4.394	0.877
Without SMOTE	8.392	6.853	0.608
Without contextual room_type	6.753	5.080	0.834
Pollutant-only inputs	4.558	3.392	0.924
Without feature scaling	6.354	4.565	0.852

The pollutant-only configuration achieved the highest ablation R2 because the synthetic IAQ formula depends directly on PM2.5, PM10, NO2, and CO, while temperature, humidity, SO2, and room type do not directly enter Equation (1). This finding is not a reason to remove contextual variables from the final framework. Instead, it reveals the trade-off between formula-specific accuracy and context-aware deployment. In real IAQ systems, thermal variables and room context are important for comfort interpretation, sensor placement, and model generalization.

E. Additional Metrics Results

Table 8 shows that the proposed compact model is relatively stable across validation folds. RMSE varies only from 5.705 to 5.956, and R2 remains between 0.869 and 0.886. This narrow spread indicates that the result is not dominated by a single favorable split. For an embedded proof-of-concept, such stability is important because the model is expected to operate repeatedly on streaming sensor input rather than on a single static test subset.

Table 8. Fold-level metrics of the proposed compact MLP

Fold	RMSE	MAE	R2
1	5.764	4.322	0.886
2	5.705	4.433	0.877
3	5.739	4.306	0.884
4	5.805	4.388	0.869
5	5.956	4.521	0.870
Average	5.794	4.394	0.877

Figure 2 confirms the same pattern visually. Predictions follow the ideal diagonal trend, although the compact model compresses some high-variance regions because it has only one narrow hidden layer.

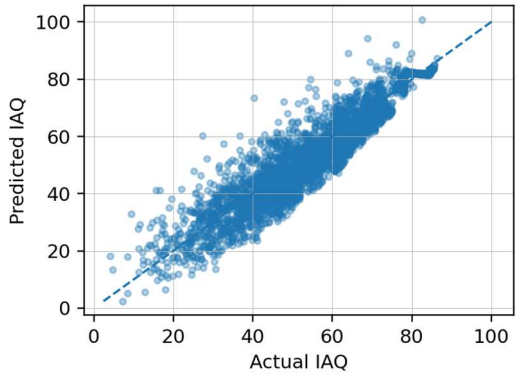

Figure. 2 Actual versus predicted IAQ scores for the compact MLP across validation folds.

e-ISSN: 2614-8404
p-ISSN:2776-3234

This behavior is consistent with the design objective: the model is sufficiently expressive for initial IAQ estimation while remaining small enough for deployment on a microcontroller. Table 9 separates measured training-side metrics from implementation-oriented estimates. The parameter memory estimate is small, but actual flash usage and RAM usage will also depend on the TensorFlow Lite Micro runtime, operator resolver, tensor arena size, and firmware overhead. Therefore, the serial-monitor outputs are necessary in the next physical validation stage to quantify real latency, free heap, CPU load, and power implications on the ESP32 board.

Table 9. Additional implementation-oriented metrics

Metric	Value/status	Interpretation
Trainable parameters	121	Derived from the compact MLP architecture
Approximate float parameter memory	484 bytes	121 parameters multiplied by 4 bytes
Approximate int8 parameter memory	121 bytes	Analytical estimate after full quantization
Firmware serial outputs	IAQ, comfort, inference time, heap, CPU	Implemented as runtime feedback fields
Physical sensor validation	Not yet measured	Required before operational deployment

Figure 3 presents the serial monitor output generated by the ESP32 after the TensorFlow Lite Micro model was loaded and executed through the EloquentTinyML framework. The displayed environmental input consists of eight simulated sensor features, namely temperature, humidity, PM2.5, PM10, NO2, SO2, CO, and room type.

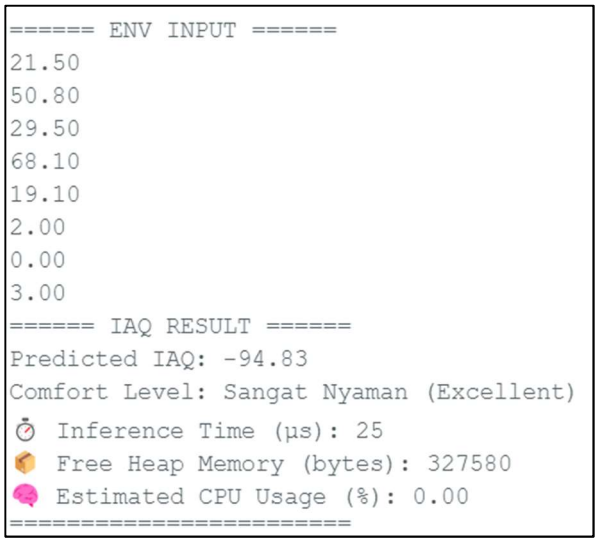

Figure. 3 The Output of Model in ESP 32.

The result demonstrates that the embedded firmware was able to pass the input vector to the deployed model, execute on-device inference, and return a predicted IAQ value together with a human-readable comfort label. In this execution cycle, the model produced an IAQ prediction of -94.83 and mapped the condition into “Sangat Nyaman (Excellent)”. The inference time was recorded at 25 microseconds, with 327,580 bytes of free heap memory remaining and an estimated CPU usage of 0.00%. These values indicate that the deployed model can run with very low computational overhead on the ESP32 platform.

However, the negative IAQ value also shows the importance of adding output post-processing, such as clipping the predicted score into the valid IAQ range, before interpreting the result as an environmental condition. Therefore, Figure 3 should be interpreted not as final field validation, but as evidence that the TinyML deployment pipeline, model loading, inference execution, memory monitoring, and serial-based feedback mechanism function correctly on the target embedded device.

IV. DISCUSSION

The central problem addressed in this study is the gap between IAQ prediction as an offline machine learning task and IAQ prediction as a deployable embedded intelligence function. In practical indoor environments, an IAQ prediction system is expected to operate locally, respond with low latency, remain inexpensive, and continue functioning even when network connectivity is limited or unstable. This requirement shifts the evaluation focus from predictive accuracy alone toward a broader embedded-AI perspective that includes model compactness, conversion feasibility, firmware integration, runtime observability, and the semantic validity of the output. Therefore, the proposed framework was not designed merely to optimize an IAQ regression score in Python, but to examine whether a trained model can be transformed into an executable TinyML function on a resource-constrained ESP32 device.

The experimental setup was intentionally organized as a proof-of-concept. Synthetic data were used to control the relationship among pollutant variables, IAQ scores, and comfort labels, while SMOTE was applied to reduce the dominance of majority comfort conditions during model training. Five-fold cross-validation was used to reduce split-specific bias and to obtain a more stable estimate of regression performance. The deployment pathway included TensorFlow Lite conversion, model transformation into C-array files, and Arduino firmware integration using EloquentTinyML with the TensorFlow Lite Micro backend. This workflow represents an important intermediate stage between offline IAQ modeling and physical deployment with calibrated sensors because it verifies whether the model can be embedded, called, and monitored within the target microcontroller environment.

The core quantitative result shows that the compact MLP achieved RMSE 5.794, MAE 4.394, and R2 0.877 on the balanced synthetic dataset. These results indicate that a very small neural architecture is still capable of approximating the pollutant-driven IAQ score under the controlled synthetic formulation. Compared with larger or ensemble-based models, the compact MLP sacrifices part of its offline predictive capacity but offers a more favorable trade-off for embedded deployment. This trade-off is central to TinyML-based IAQ systems, where a model must be sufficiently accurate to support local interpretation but sufficiently small to satisfy memory, latency, and firmware integration constraints. The fold-level stability also suggests that the model does not depend excessively on a single train-test partition, which is important for later streaming inference scenarios.

The model comparison should, however, be interpreted carefully. Because the IAQ score in the synthetic dataset is generated using a deterministic pollutant-weighted formula, linear regression can act as an artificial upper-bound baseline rather than a realistic representation of IAQ prediction in real

e-ISSN: 2614-8404
p-ISSN:2776-3234

buildings. In actual indoor environments, the relationship between pollutant concentration, thermal conditions, room context, ventilation, sensor position, occupant activity, and perceived air comfort is unlikely to remain purely linear. Thus, the strong performance of a linear baseline should not be generalized as evidence that linear modeling is sufficient for real IAQ intelligence. Instead, it highlights the limitation of the current synthetic generator and reinforces the need for physical sensor data, nonlinear environmental dynamics, and temporal validation in future work.

A deeper analytical insight is provided by the ablation study. SMOTE substantially improves the compact MLP, indicating that the distribution of comfort-related conditions affects the ability of a small model to represent less frequent states. Feature scaling also improves model behavior, confirming that preprocessing remains important even when the final target is a microcontroller-based deployment. The pollutant-only input configuration performs better than the full input configuration because the current IAQ score formulation is directly driven by PM2.5, PM10, NO2, and CO. However, this does not imply that temperature, humidity, and room type should be removed from a deployable IAQ-comfort framework. These contextual variables are important for interpreting comfort, supporting future physical validation, and extending the system from simple IAQ score prediction toward more realistic indoor exposure and comfort reasoning.

The ESP32 output shown in Figure 3 provides an important firmware-level finding. The serial monitor confirms that the deployed model can receive an eight-dimensional environmental input vector, execute inference on the ESP32, and return runtime information such as predicted IAQ, comfort label, inference time, free heap memory, and estimated CPU usage. The recorded inference time of 25 microseconds, remaining heap memory of 327580 bytes, and estimated CPU usage of 0.00% suggest that the deployed model imposes a very small computational burden on the device. This result supports the technical feasibility of executing the compact model within the ESP32 environment. Nevertheless, these metrics should be interpreted as initial runtime indicators rather than final embedded performance claims, because the current test uses simulated inputs and does not yet include sensor acquisition latency, communication overhead, power consumption, long-duration execution, or multi-cycle stability analysis.

More importantly, Figure 3 reveals a critical semantic-validity issue. The model produced a predicted IAQ value of -94.83, while the firmware mapped this output to "Sangat Nyaman (Excellent)." This finding shows that successful model execution on the microcontroller does not automatically guarantee meaningful environmental interpretation. The negative prediction is outside the intended IAQ range and should be treated as an invalid or out-of-range output. The incorrect assignment of an "Excellent" label occurs because the firmware-level label-mapping logic evaluates threshold conditions without first constraining or validating the regression output. This demonstrates the need for post-processing safeguards, including clipping the predicted IAQ score into a valid range, rejecting invalid values, checking input normalization consistency, and separating numerical inference from semantic decision rules. In embedded IAQ systems, such safeguards are not secondary details; they are necessary to prevent misleading comfort feedback.

This finding also clarifies the distinction between deployment feasibility and deployment validity. From a deployment-feasibility perspective, the proposed pipeline successfully moves the trained model from Python into Arduino firmware and executes it on the ESP32 using EloquentTinyML. From a deployment-validity perspective, however, the observed negative IAQ output indicates that further refinement is required before the system can be interpreted as a reliable IAQ decision-support tool. The firmware should include bounded regression output, robust label mapping, sensor plausibility checks, and calibration-aware preprocessing. In addition, the same preprocessing used during model training, particularly feature scaling, must be consistently represented in the embedded inference stage. Without this consistency, a model that performs well during cross-validation may produce unrealistic values when deployed.

Practically, the proposed framework is valuable as an instructional and research prototype for embedded IAQ intelligence. It demonstrates how an IAQ model can move from dataset generation and training into firmware-level inference, how serial output can be used to inspect model behavior, and how runtime observability can be prepared through inference time, memory availability, and CPU usage fields. The framework also exposes the types of engineering issues that emerge only after deployment, such as invalid regression ranges and inappropriate threshold-based label mapping. These findings are useful because they show that embedded AI development requires not only model training but also output validation, runtime monitoring, and firmware-level control logic.

The novelty of this study is positioned at the system level by integrating IAQ regression, comfort-label balancing, compact MLP modeling, TensorFlow Lite conversion, C-array embedding, and EloquentTinyML-based ESP32 deployment into a single reproducible workflow. Rather than proposing a new neural network architecture, this study demonstrates how a lightweight IAQ predictor can be designed, trained, converted, embedded, executed, and evaluated on a resource-constrained microcontroller. This contribution is different from studies that mainly focus on sensor monitoring, cloud-based analytics, or offline prediction accuracy. However, several limitations remain. The dataset is synthetic and does not fully represent real indoor conditions, the data generator may produce unrealistic values, and the embedded test still uses simulated inputs instead of calibrated physical sensors. The negative IAQ value shown in Figure 3 also indicates the need for stronger output validation, such as clipping the predicted score before assigning a comfort label, while the reported CPU usage of 0.00% should be interpreted cautiously because it may be affected by rounding or measurement resolution. Future work should use calibrated indoor sensor data, apply bounded preprocessing, maintain consistent normalization between training and deployment, measure real ESP32 latency and energy consumption, and evaluate long-term stability across different room types.

V. CONCLUSION

This paper revised and extended the IAQ TinyML deployment study into an academic framework for implementing a machine learning based IAQ predictor on NodeMCU ESP32

e-ISSN: 2614-8404
p-ISSN:2776-3234

using EloquentTinyML. The proposed workflow includes synthetic IAQ dataset generation, comfort-label construction, SMOTE balancing, feature scaling, compact MLP training, five-fold validation, TensorFlow Lite conversion, C-array model embedding, and Arduino firmware integration. The compact MLP achieved an average RMSE of 5.794, MAE of 4.394, and R2 of 0.877 while using only 121 trainable parameters.

The comparison and ablation results show that the model is not the most accurate offline alternative on a deterministic synthetic formula, but it provides a practical balance between regression performance, model compactness, nonlinear extensibility, and microcontroller deployment readiness. The framework is therefore suitable as a proof-of-concept and educational platform for embedded IAQ prediction. Future research should replace synthetic data with calibrated sensor measurements, constrain the data generator to physically realistic pollutant ranges, quantify real ESP32 latency and memory usage, and evaluate model robustness under long-term indoor environmental variation.

REFERENCES

[1] World Health Organization. (2021). WHO global air quality guidelines: Particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide. Geneva: World Health Organization.

[2] Hayward, I., Hirst, E., Stewart, G., & Acton, W. J. F. (2024). Low-cost air quality sensors: Biases, corrections and challenges in their comparability. Atmosphere, 15(12), 1523. DOI: 10.3390/atmos15121523.

[3] Chojer, H., Branco, P. T. B. S., Martins, F. G., Alvim-Ferraz, M. C. M., & Sousa, S. I. V. (2020). Development of low-cost indoor air quality monitoring devices: Recent advancements. Science of the Total Environment, 727, 138385. DOI: 10.1016/j.scitotenv.2020.138385.

[4] Soro, S. (2021). TinyML for ubiquitous edge AI. arXiv preprint arXiv:2102.01255.

[5] Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A comprehensive survey on TinyML. IEEE Access, 11, 96892-96922. DOI: 10.1109/ACCESS.2023.3294111.

[6] David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2021). TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems, 3, 800-811.

[7] EloquentArduino. (2024). EloquentTinyML: An eloquent interface to TensorFlow Lite for Microcontrollers. Arduino Libraries Documentation.

[8] Espressif Systems. (2025). ESP32 Series Datasheet. Shanghai: Espressif Systems.

[9] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321-357.

[10] Warden, P., & Situnayake, D. (2019). TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. Sebastopol, CA: O Reilly Media.

[11] Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., Montino, P., Kanter, D., Ahmed, S., Pau, D., Thakker, U., Torrini, A., Warden, P., Cordaro, J., Di Guglielmo, G., Duarte, J., Gibellini, S., Parekh, V., Tran, H., Tran, N., Wenxu, N., & Xuesong, X. (2021). MLPerf Tiny benchmark. arXiv preprint arXiv:2106.07597.

[12] Hymel, S., Banbury, C., Situnayake, D., Elium, A., Ward, C., Kelcey, M., Baaijens, M., Majchrzycki, M., Plunkett, J., Tischler, D., Grande, A., Moreau, L., Maslov, D., Beavis, A., Jongboom, J., & Reddi, V. J. (2023). Edge Impulse: An MLOps platform for Tiny Machine Learning. Proceedings of Machine Learning and Systems, 5.

[13] Guo, Z., Wang, X., & Ge, L. (2023). Classification prediction model of indoor PM2.5 concentration using CatBoost algorithm. Frontiers in Built Environment, 9, 1207193. DOI: 10.3389/fbuil.2023.1207193.

[14] Yu, W., Nakisa, B., Loke, S. W., Stevanovic, S., Guo, Y., & Rastgoo, M. N. (2024). Indoor PM2.5 forecasting and the association with outdoor air pollution: A modelling study based on sensor data in Australia. arXiv preprint arXiv:2405.07404.

[15] Desouza, P., Kahn, R., Stockman, T., Obermann, W., Crawford, B., Wang, A., Crooks, J., Li, J., & Kinney, P. (2022). Calibrating networks of low-cost air quality sensors. Atmospheric Measurement Techniques, 15, 6309-6328. DOI: 10.5194/amt-15-6309-2022.

[16] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. DOI: 10.1145/2939672.2939785.

[17] Breiman, L. (2001). Random forests. Machine Learning, 45, 5-32. DOI: 10.1023/A:1010933404324.