



Available Online at www.hithaldia.in/locate/ECCN
All Rights Reserved

ORIGINAL CONTRIBUTION

Edge AI-Powered Predictive Water-Quality Monitoring and Early Warning for Smart Aquaculture

¹Rishu Raj, ²Pryobroto Sarkar, ³Ritwik Kapat, ⁴Riya Panda, ⁵Debjyoti Adhikari, ⁶Ritwik Haldar

^{1,2,3,4}UG Students, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

⁶Assistant Professor, Dept. of Information Technology, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

⁶Associate Professor, Dept. of Electronics and Communication Engineering, Haldia Institute of Technology, Haldia, Purba Medinipur, West Bengal, India

ABSTRACT

Water quality in aquaculture ponds varies with environmental conditions, biological activity, and daily cycles, so periodic manual measurement is inherently reactive and detects unfavourable conditions only after they have developed. This work presents an Edge Artificial Intelligence (Edge AI) predictive monitoring system for smart fish ponds that forecasts water temperature and pH and provides early risk indication without cloud-based inference. A real-world 24-hour pond dataset is processed by converting timestamps into a decimal hour-of-day feature. Decision Tree Regression and a Long Short-Term Memory (LSTM) network are trained and compared: the Decision Tree uses hour-of-day as its predictor, while the LSTM uses normalised temperature sequences as a temporal benchmark. For temperature, both models reproduce the diurnal pattern with almost identical accuracy, the Decision Tree achieving a coefficient of determination of 0.9989 and the LSTM 0.9987, with the LSTM marginally lower in absolute error. The Decision Tree is therefore selected not on accuracy but on deployability, since it trains in under a second, occupies a few tens of kilobytes, and executes on a microcontroller, whereas the LSTM could not be deployed on the target device. The selected models are converted to C++ using micromlgen and deployed on an ESP32, which synchronises time over NTP, performs local inference in under one millisecond, and displays four-hour forecasts on a 16 × 2 I2C LCD. A threshold-based mechanism issues early warnings when predicted values fall outside predefined safe ranges. The prototype demonstrates a compact TinyML approach to predictive aquaculture monitoring on inexpensive embedded hardware.

KEYWORDS: Edge AI, Smart Aquaculture, Decision Tree Regression, LSTM, TinyML, Water Quality Monitoring.

2. INTRODUCTION

Aquaculture plays an important role in modern food production, and maintaining suitable water-quality conditions is essential for healthy fish growth and pond productivity. Temperature and pH are two of the most informative indicators of pond condition, and both vary with environmental conditions, biological activity, feeding, rainfall, and the daily solar cycle. Conventional monitoring generally relies on periodic manual measurement, which can miss rapid changes and provides information only after an undesirable condition has already developed. A monitoring scheme that anticipates the near-term evolution of

these parameters would allow corrective action to be taken before fish are placed under stress.

This work proposes a low-cost Edge AI predictive monitoring system for smart fish ponds based on the ESP32 platform. The system analyses real-world pond temperature and pH data and uses machine-learning models to forecast future water-quality conditions. Decision Tree Regression is developed for temperature and pH prediction, while an LSTM network is implemented as a temporal benchmark. The selected Decision Tree models are converted into embedded C++ using micromlgen and deployed directly on the ESP32, enabling local inference without continuous cloud-based machine-learning processing.

The system generates four-hour forecasts and displays the predicted values on a 16×2 I2C LCD. A threshold-based early-warning mechanism identifies predicted temperature and pH values that fall outside predefined safe ranges, so that potential risks are indicated before they occur. It should be noted that the available dataset contains no labelled anomalous events; the implementation is therefore correctly described as a predictive early-warning system rather than a labelled anomaly detector. The primary objective of this work is to demonstrate a compact and practical TinyML approach to predictive aquaculture monitoring using inexpensive embedded hardware.

The remainder of this paper is organised as follows. Section II reviews related work in smart aquaculture, water-quality prediction, and TinyML. Section III describes the proposed system architecture, the dataset, and the machine-learning models. Section IV presents the experimental evaluation and the model comparison. Section V concludes the paper and sets out its limitations and future scope.

2. LITERATURE REVIEW

Boyd and Tucker [1] established the importance of maintaining suitable water-quality conditions in aquaculture ponds and documented the physical and biological processes that drive the diurnal variation of temperature, pH, and dissolved oxygen. Gubbi et al. [2] set out the architectural elements of the Internet of Things that underpin distributed environmental sensing, and Olanubi et al. [3] applied these principles directly to aquaculture through an IoT-based intelligent water-quality management system. Geetha and Gouthami [4] demonstrated an IoT-enabled real-time water-quality monitoring system, confirming the practicality of continuous measurement in place of periodic sampling.

A second group of studies applies machine learning to the prediction of water-quality parameters. Zhang et al. [5] combined kernel principal component analysis with a recurrent network to predict the trend of dissolved oxygen, and Zambrano et al. [6] showed that classical machine-learning models can predict manually measured water-quality parameters in fish farming with useful accuracy. Li et al. [7] applied a gated recurrent unit network to dissolved-oxygen prediction in a fishery pond and argued

explicitly for the lower parameter count and training cost of the gated recurrent unit relative to the LSTM, while Chen et al. [8] combined an LSTM with an attention mechanism for river water-quality prediction. These works establish that the temporal structure of pond data is learnable, but they assume computation on a server or a personal computer.

A third group addresses the deployment of such models on constrained hardware. Banbury et al. [9] characterised the benchmarking and resource challenges of TinyML on microcontroller-class devices. Kotsiantis [10] reviewed the Decision Tree as an efficient and interpretable learning method whose inference reduces to a sequence of comparisons, which makes it well suited to embedded execution, while Hochreiter and Schmidhuber [11] introduced the LSTM architecture that has since become the standard baseline for temporal prediction. Salerno [12] provides micromlgen, which converts trained scikit-learn estimators into self-contained C++ classes, and Espressif Systems [13] documents the ESP32 as a low-cost platform combining a dual-core processor with integrated Wi-Fi.

Sowmiya et al. [14] brought these strands together in a TinyML-based reconfigurable IoT platform for brackish-water aquaculture monitoring. Most existing systems, however, concentrate either on measurement and transmission or on prediction performed off the device, and comparatively few compare a deployable model against a recurrent benchmark on the same data before moving inference onto the microcontroller itself. The proposed system addresses this gap by training a Decision Tree and an LSTM benchmark on the same pond record, quantifying the difference between an hour-of-day regressor and a one-step-ahead recurrent predictor, and deploying the selected model directly on an ESP32 to generate four-hour temperature and pH forecasts with local early-warning capability.

3. PROPOSED SYSTEM

3.1. System Architecture

The proposed system comprises two connected stages. The first is an offline machine-learning pipeline executed on a personal computer: it loads the pond dataset, converts timestamps to decimal hours, performs exploratory analysis, trains the Decision Tree and LSTM models, evaluates their

performance, and exports the selected Decision Tree models. The second is the edge-deployment stage, in which the generated C++ model header is compiled into the ESP32 firmware. The device obtains the current time through the Network Time Protocol (NTP), predicts the next four hourly values, and displays them on the LCD. The complete workflow is shown in Fig. 1.

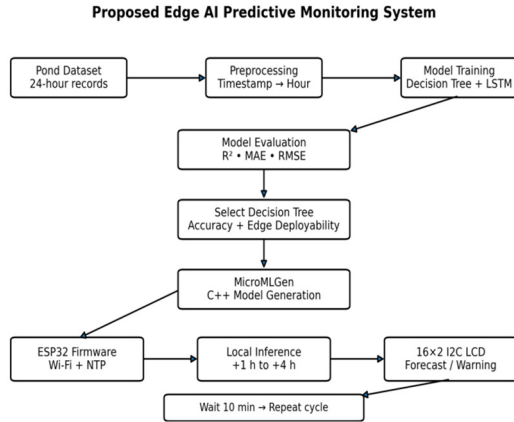


Figure 1: End-to-end Edge AI workflow, from dataset preparation and model training to continuous four-hour forecasting and warning on the ESP32.

3.2. Dataset and Feature Engineering

The dataset was collected from an active fish pond over a continuous 24-hour period and contains timestamp, temperature ($^{\circ}\text{C}$), and pH fields. Each timestamp is converted into a decimal hour feature computed from its hour, minute, and second components, so that the time of day is represented as a single continuous variable. Hour-of-day is used as the principal predictor for the Decision Tree models, on the basis that both parameters are driven primarily by the solar cycle.

3.3. Decision Tree Regression

Two Decision Tree Regression models are trained separately, one for temperature and one for pH. The scikit-learn Decision Tree Regressor is configured with `max_depth=10` and `random_state=42`, and the dataset is divided into 80% training and 20% testing partitions. The model recursively partitions the hour feature and returns the learned target value at the corresponding leaf, so that inference at run time reduces to a short sequence of numerical comparisons.

3.4. LSTM Benchmark

The LSTM model uses a sliding window of 20 normalised temperature values to predict the next

value. Its architecture, summarised in Table 1, contains an LSTM layer with 64 units, a second LSTM layer with 32 units, and a single-unit linear dense output layer. The model is trained with the Adam optimiser and mean squared error loss for 20 epochs, with a batch size of 64 and a 20% validation split [15]. It should be noted that the LSTM performs one-step-ahead prediction from a window of past temperature values, whereas the Decision Tree maps hour-of-day directly to the target; the two are therefore compared as alternative modelling strategies on the same record rather than on an identical prediction task.

Table 1: LSTM benchmark model architecture.

Layer	Type	Units	Output shape
1	LSTM	64	(batch, 20, 64)
2	LSTM	32	(batch, 32)
3	Dense	1	(batch, 1)

3.5. Model Evaluation Metrics

The models are evaluated using the coefficient of determination (R^2), the mean absolute error (MAE), and the root mean squared error (RMSE). R^2 measures the proportion of variance explained, MAE reports the mean magnitude of the prediction error in the units of the measured parameter, and RMSE weights larger errors more heavily and is therefore the more sensitive indicator of occasional large deviations.

3.6. Edge Deployment

Micromlgen converts the trained scikit-learn Decision Tree estimators into self-contained C++ classes exposing a `predict()` method [12]. The temperature and pH models are stored in `dt_models.h` as `TempPredictor` and `pHPredictor`. The ESP32 firmware is written in Arduino C++ and combines Wi-Fi connectivity, NTP time synchronisation, the embedded prediction classes, and the `LiquidCrystal_I2C` library. The LCD is addressed over I2C at address 0x27, with GPIO 21 used for SDA and GPIO 22 for SCL, as illustrated by the reference wiring in Fig. 2. In each cycle the firmware calculates the next four target hours, performs local inference, and presents the forecast in 12-hour AM/PM format. Each forecast frame is displayed for three seconds, followed by a ten-minute pause before the next cycle begins.

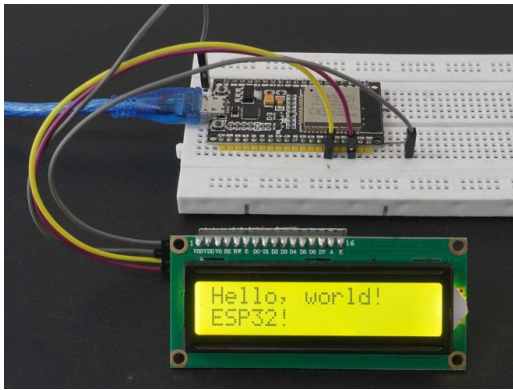


Figure 2: Reference wiring of the ESP32 interfaced with a 16 × 2 I2C LCD (pin assignment after [13]).

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

The machine-learning experiments were performed on a Windows 11 computer with an Intel Core i5 processor and 8 GB of RAM, using Python 3.10 with Pandas, NumPy, scikit-learn, TensorFlow, Matplotlib, Seaborn, and micromlgen [15], [16]. The ESP32 firmware was developed and flashed using Arduino IDE 2.x.

4.2. Exploratory Analysis

The recorded data show the diurnal behaviour expected of a shallow pond. Temperature rises from approximately 24.5 °C in the early morning to slightly above 31 °C in the afternoon, and pH varies across a narrower band of approximately 7.05 to 7.80, rising during the period of peak photosynthetic activity and falling thereafter.

The correlation structure of the dataset is shown in Fig. 3. Temperature and pH are very strongly correlated with one another ($r = 0.97$), which is consistent with both being driven by the same photosynthetic and thermal cycle. The linear correlation between hour-of-day and each parameter is, by contrast, weak ($r = 0.25$ for temperature and $r = 0.39$ for pH). This apparent inconsistency is a property of the Pearson coefficient rather than of the data: the diurnal profile rises and then falls within the observation window, so the relationship between hour and parameter is strong but non-monotonic, and a linear measure of association necessarily understates it. This observation is important to the design of the system, because it rules out simple linear regression on hour-of-day and motivates the use of a model capable of representing non-monotonic structure, such as a regression tree.

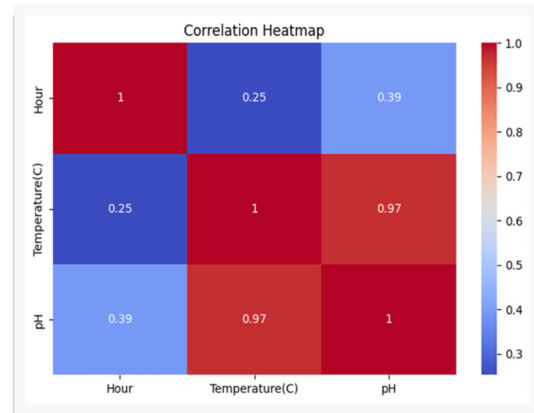


Figure 3: Pearson correlation heatmap for hour-of-day, temperature, and pH.

The frequency distributions of the two parameters are shown in Fig. 4. Neither is normally distributed. The temperature distribution spans approximately 24.5 °C to 31.5 °C and is bimodal with a pronounced mode near 31 °C, reflecting the extended period the pond spends near its afternoon maximum. The pH distribution spans approximately 7.05 to 7.80 and is left-skewed, with the greatest density towards the upper end of the range and a longer tail towards lower values. No extreme outliers are present in either parameter, indicating that the recorded data are free of gross sensor faults over the observation period.

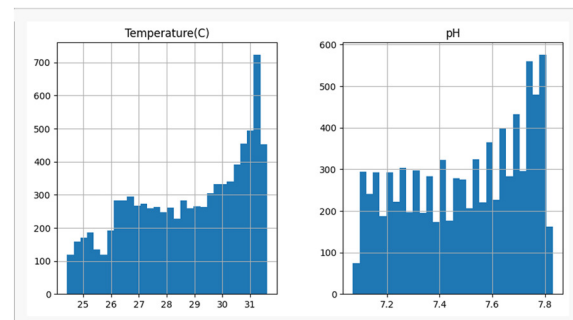


Figure 4: Frequency distributions of temperature and pH over the 24-hour observation period.

4.3. Prediction Performance

Table 2 reports the performance of the trained models on the held-out test partition. Both models explain almost all of the variance in the measured temperature, and the difference in their absolute errors is small in relation to the range over which the parameter varies during the observation window. The Decision Tree pH model achieves an R^2 of 0.9850 with an MAE of 0.020 and an RMSE of 0.030 pH units, slightly below its

temperature counterpart, which is consistent with the narrower dynamic range of the pH signal.

Table 2. Model performance on the held-out test partition. Temperature errors are expressed in °C and pH errors in pH units.

Model / target	R ²	MAE	RMSE
Decision Tree – Temperature	0.9989	0.057	0.069
LSTM – Temperature	0.9987	0.053	0.063
Decision Tree – pH	0.9850	0.020	0.030

4.4. Model Comparison

Fig. 5 compares the coefficient of determination achieved by the two models for temperature prediction. The scores are effectively indistinguishable, at 0.9989 for the Decision Tree and 0.9987 for the LSTM, indicating that both architectures capture the diurnal structure of the data equally well. The result is unsurprising: the underlying signal is a smooth, strongly periodic function of time of day, and both a sufficiently deep regression tree and a recurrent network are able to represent it closely.

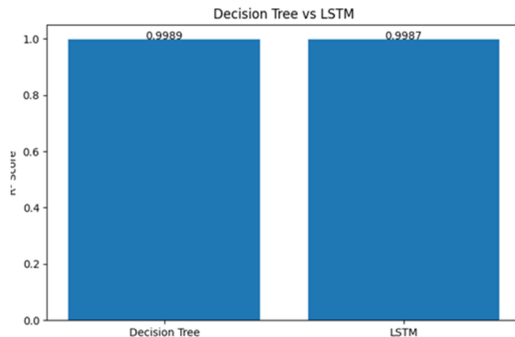


Figure 5: Coefficient of determination for the Decision Tree and the LSTM benchmark.

Fig. 6 compares the two models on absolute error. Here the LSTM is marginally the more accurate of the two, with an MAE of 0.053 against 0.057 and an RMSE of 0.063 against 0.069 for the Decision Tree. The difference is 7% in MAE and 9% in RMSE, corresponding to between four and six thousandths of a degree Celsius, and is therefore not practically significant for pond management.

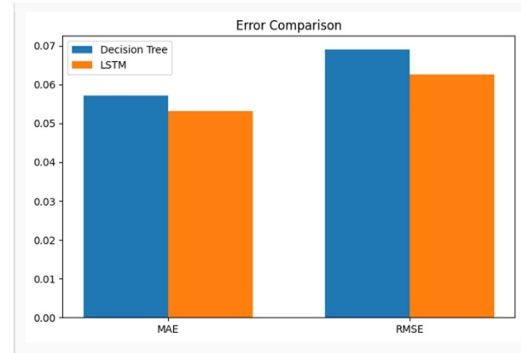


Figure 6: Mean absolute error and root mean square error for the Decision Tree and the LSTM benchmark.

The selection of the Decision Tree for deployment therefore does not rest on predictive accuracy, and Table 3 sets out the basis on which it was made. At practically equivalent accuracy, the Decision Tree trains in under a second rather than several minutes, compiles to a model of a few tens of kilobytes, and executes on the ESP32 in under one millisecond per prediction. The LSTM was not deployed on the target device: it requires a floating-point tensor runtime and a persistent 20-sample input window, and micromlgen provides no conversion path for recurrent models, so no equivalent single-header C++ export was available. The appropriate conclusion is that for a signal of this regularity the additional representational capacity of the recurrent model buys no useful accuracy, and the simpler model should be preferred because it is the one that can actually run at the edge.

Table 3: Comparison of the Decision Tree and the LSTM benchmark.

Metric	Decision Tree	LSTM
R ² (temperature)	0.9989	0.9987
MAE	0.057	0.053
RMSE	0.069	0.063
Training time	< 1 s	Minutes
Model size	≈ 10–50 KB	Not measured
ESP32 inference	< 1 ms	Not deployed
Edge deployment	Full TinyML	Not feasible

4.5. Real-Time ESP32 Evaluation

During testing, the ESP32 connected to the configured Wi-Fi network, displayed a connection message, synchronised its clock over NTP, and reported that the Decision Tree models were ready. At a test time of 19:45 Indian Standard Time (IST), the device generated forecasts for the following one, two, three, and four hours. The predicted values followed the

expected evening decline in both temperature and pH, confirming correct operation of the embedded inference loop and of the time-handling logic that maps the current NTP time onto the hour feature expected by the models.

4.6. Predictive Risk Indication

Safe operating ranges of 26–32 °C for temperature and 6.5–8.5 for pH are defined for the cultivated species. A predicted value outside the safe range raises an advisory warning on the LCD, and a value beyond the critical boundaries listed in Table 4 raises a critical warning. Sustained temperatures below 24 °C or above 34 °C are associated with thermal stress and reduced dissolved-oxygen solubility, while pH values below 6.0 or above 9.0 are associated with osmoregulatory failure. The observed minimum of approximately 24.5 °C lies below the 26 °C safe lower bound but above the 24 °C critical bound, so early-morning forecasts on this record raise an advisory rather than a critical warning. Because the mechanism acts on predicted rather than measured values, a warning is raised in advance of the condition arising, which is the principal operational advantage of the predictive approach over threshold checking applied to live readings.

Table 4. Safe ranges and critical boundaries used for predictive risk indication.

Parameter	Safe range	Critical below	Critical above
Temperature	26–32 °C	< 24 °C	> 34 °C
pH	6.5–8.5	< 6.0	> 9.0

5. CONCLUSION

This paper presented the design and evaluation of a low-cost Edge AI predictive monitoring system for smart aquaculture. Decision Tree Regression and an LSTM benchmark were trained on a 24-hour pond dataset and compared on the same pond record, though on different input

representations, and the selected Decision Tree models were deployed on an ESP32 using micromlgen to produce four-hour temperature and pH forecasts with threshold-based risk indication. The two models proved almost identical in accuracy, with the LSTM marginally lower in absolute error, so the Decision Tree was selected on the grounds of embedded deployability rather than predictive performance. The result is consistent with a broader design expectation for TinyML applications, which further work will need to confirm: where the target signal is strongly periodic and low-dimensional, a compact interpretable model may deliver accuracy comparable to that of a recurrent network at a small fraction of its computational cost.

Two limitations should be noted. First, the models were trained on a single 24-hour record, and the test partition was drawn at random from within that record; the reported accuracy therefore reflects interpolation within one observed diurnal cycle rather than generalisation to unseen days, and a multi-day evaluation with a chronological train–test split is required before the forecasting accuracy can be regarded as established. Second, the dataset contains no labelled anomalous events, so the early-warning mechanism is validated only against its defined thresholds and not against recorded incidents. Future work will address both points through extended multi-day data collection, direct sensor integration in place of the offline dataset, the inclusion of dissolved oxygen and turbidity as additional parameters, and adaptive on-device retraining as further data accumulate.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Department of Electronics and Communication Engineering, Haldia Institute of Technology, for the laboratory facilities used in this work.

REFERENCES

- [1] C. E. Boyd and C. S. Tucker, *Pond Aquaculture Water Quality Management*. Norwell, MA: Kluwer Academic Publishers, 1998. doi: 10.1007/978-1-4615-5407-3
- [2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013. doi: 10.1016/j.future.2013.01.010

- [3] O. O. Olanubi, T. T. Akano, and O. S. Asaolu, "Design and development of an IoT-based intelligent water quality management system for aquaculture," *Journal of Electrical Systems and Information Technology*, vol. 11, art. no. 15, 2024. doi: 10.1186/s43067-024-00139-z
- [4] S. Geetha and S. Gouthami, "Internet of things enabled real time water quality monitoring system," *Smart Water*, vol. 2, art. no. 1, 2017. doi: 10.1186/s40713-017-0005-y
- [5] Y.-F. Zhang, P. Fitch, and P. J. Thorburn, "Predicting the trend of dissolved oxygen based on the kPCA-RNN model," *Water*, vol. 12, no. 2, art. no. 585, 2020. doi: 10.3390/w12020585
- [6] A. F. Zambrano, L. F. Giraldo, J. Quimbayo, B. Medina, and E. Castillo, "Machine learning for manually-measured water quality prediction in fish farming," *PLOS ONE*, vol. 16, no. 8, art. no. e0256380, 2021. doi: 10.1371/journal.pone.0256380
- [7] W. Li, H. Wu, N. Zhu, Y. Jiang, J. Tan, and Y. Guo, "Prediction of dissolved oxygen in a fishery pond based on gated recurrent unit (GRU)," *Information Processing in Agriculture*, vol. 8, no. 1, pp. 185–193, 2021. doi: 10.1016/j.inpa.2020.02.002
- [8] H. Chen et al., "Water quality prediction based on LSTM and attention mechanism: A case study of the Burnett River, Australia," *Sustainability*, vol. 14, no. 20, art. no. 13231, 2022. doi: 10.3390/su142013231
- [9] C. R. Banbury et al., "Benchmarking TinyML systems: Challenges and direction," arXiv:2003.04821, 2020. [Online]. Available: <https://arxiv.org/abs/2003.04821>
- [10] S. B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261–283, 2013. doi: 10.1007/s10462-011-9272-4
- [11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735
- [12] S. Salerno, *micromlgen: Generate C code for microcontrollers from Python's scikit-learn classifiers*, v1.1.28, 2021. [Online]. Available: <https://github.com/eloquentarduino/micromlgen>
- [13] Espressif Systems, *ESP32 Technical Reference Manual*, v5.8, 2026. [Online]. Available: https://documentation.espressif.com/esp32_technical_reference_manual_en.pdf
- [14] V. Sowmiya, G. R. Kanagachidambaresan, and M. Muralidhar, "Tiny ML-based reconfigurable IoT platform design for brackish water aquaculture monitoring," *Wireless Networks*, vol. 30, no. 9, pp. 7153–7165, 2024. doi: 10.1007/s11276-023-03537-9
- [15] F. Chollet, *Deep Learning with Python*. Shelter Island, NY: Manning Publications, 2017.
- [16] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2019.