International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 03, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

Regression-PID: Bare-Metal Predictive Temperature Control via Multiple Linear Regression on an ESP32-Based IoT Egg Incubator

1st Zainal Arifin, 2nd Siti Rokhmah, 3rd Tino Feri Efendi
[1,2,3] *Department of Informatics, Institut Teknologi Bisnis AAS Indonesia*
Sukoharjo, Indonesia.
[1] zainalarifin.edu@gmail.com, [2] elfathiey@gmail.com, [3] tinoferi8@gmail.com

Abstract **-** The success of egg hatching depends on the stability of the incubation temperature within a highly strict tolerance (±0.3°C). Conventional PID controllers in egg incubators are reactive, correcting temperature only after an error is detected, which makes them prone to overshoot during warm-up. This study proposes Regression-PID — a PID controller augmented with a Multiple Linear Regression (MLR) predictive model deployed as bare-metal arithmetic in ESP32 firmware, without any machine learning framework. Trained offline using Ordinary Least Squares on historical temperature and duty cycle data, the optimal window W = 5 yields 11 coefficients ($R^2 = 0.7634$, RMSE = 0.1512°C) stored in 44 bytes of flash. The 30-second-ahead temperature prediction drives the proportional and derivative terms; the integral term uses actual temperature to guarantee steady-state error elimination. A comparative experiment was performed on an ESP32-based IoT egg incubator with real-time MQTT telemetry to a cloud backend. Regression-PID reduces overshoot by 68.1% (0.44 vs. 1.38°C), ISE by 89.7% (0.71 vs. 6.92 °C²·s), IAE by 74.6%, and steady-state standard deviation by 73.9% (0.014 vs. 0.053°C); both modes maintained the ±0.3°C tolerance band 100% of the time. Computational overhead is only +12.17 µs per cycle (0.0012% of the 1-second period) with deterministic latency. Performance differences are confirmed by Mann-Whitney U ($p = 0.0009$, $r = 0.777$) and Wilcoxon Signed-Rank ($p < 0.0001$). These results demonstrate that linear regression is sufficient for predictive thermal control in quasi-linear systems, with minimal complexity and no compromise to real-time feasibility.

Keywords : IoT egg incubator, predictive control, multiple linear regression, ESP32, bare-metal deployment.

I. INTRODUCTION

The poultry sector is one of the key pillars of food security in Indonesia. According to data from Statistics Indonesia (Badan Pusat Statistik), the broiler chicken population in 2023 reached more than 3.1 billion heads, with demand continuing to rise alongside population growth [1]. In the poultry production chain, egg incubation is a critical stage that determines the success of the entire production cycle. Hatchability rate depends heavily on the incubator system's ability to maintain optimal environmental conditions throughout the 21-day incubation period [2].

Incubation temperature is the parameter most influential to embryo development. During the first 18 days of incubation, chicken embryos are poikilothermic — body temperature follows ambient temperature directly, so metabolic rate and organ development are entirely determined by the incubator temperature [2]. The optimal temperature range for chicken egg incubation is 37.5–37.8°C [2], [3]. A deviation as small as ±1°C from this optimal range has been shown to significantly affect hatch weight, chick quality, and hatchability rate [2]. Exposure to temperatures above 38.5°C accelerates metabolism excessively, depletes glycogen reserves, and reduces hatchability, while temperatures below 37°C slow embryo development and extend incubation duration [2]. Therefore, the temperature control system of an incubator must be capable of maintaining temperature within a very tight tolerance, ideally ±0.3°C from the setpoint.

Various control methods have been applied to egg incubator systems. The simplest approach is On-Off control (bang-bang controller), in which the heater is switched on when temperature drops below a lower threshold and switched off when it exceeds an upper limit. Peprah et al. [4] implemented relay-based On-Off control on a solar-powered egg incubator with remote monitoring via GSM/IoT, in which the heating element is activated when temperature drops below the setpoint and deactivated once it is exceeded; testing on quail eggs and guinea fowl eggs (not broiler chicken) recorded hatchability rates of 95.45–97.14% and 95.2%, respectively. However, On-Off control inherently produces periodic temperature oscillations of ±1–2°C [5], which can potentially affect embryo quality under suboptimal conditions. Septian et al. [5] demonstrated through simulation that PID (Proportional-Integral-Derivative) control yields far better temperature stability than On-Off, with virtually no oscillation at steady state and lower energy consumption due to the smooth PWM signal. Prabowo et al. [6] developed an ESP32-based incubator with PID control and IoT connectivity via Firebase, achieving temperature stability at 38°C with an average data transmission latency of 31.3 ms. Pandey et al. [7] also used ESP32 with PID and Arduino IoT Cloud, maintaining 37.5°C ±0.3°C but achieving only 62.5% hatchability — influenced by sensor calibration and humidity management. Simangunsong et al. [8] explored a fuzzy logic alternative, implementing Fuzzy Mamdani on an ESP32 with an SHT30 sensor and IoT web-based monitoring, achieving 71.4% hatchability with a mean temperature sensor error of only 0.78%.

Alongside the development of control methods, integration of Internet of Things (IoT) technology in incubator systems has become a significant trend. Maaño et al. [9] developed SmartHatch — an Arduino MKR1000-based monitoring system connected to Arduino IoT Cloud —

achieving 95.24% hatchability with an internal temperature standard deviation of only 0.39°C. The MQTT (Message Queuing Telemetry Transport) protocol has been proven effective for IoT communication in the livestock sector. Ko and Lee [10] built a three-tier MQTT architecture on a smart livestock farm using ESP32, recommending QoS 1 as the optimal balance between reliability (99.2% data collection rate) and latency (150.2 ± 6.4 ms). Has et al. [11] analyzed MQTT efficiency for agricultural IoT and found that protocol overhead at QoS 0 adds only 936 bytes per transaction, making it highly suitable for microcontroller devices with limited bandwidth.

Although conventional control methods have demonstrated success, all such approaches are reactive — control actions are taken only after an error is detected, not before a deviation occurs. Predictive control approaches leveraging neural network models have proven capable of overcoming this limitation. Almachi et al. [12] showed that PID augmented with real-time gain updates from an artificial neural network (ANN) can outperform conventional fixed-gain PID in high-temperature industrial furnace control: the ANN-assisted PID approach reduced mean absolute error to 5.08°C, decreased overshoot from 53% to 41%, and shortened settling time by 20% compared to the best fixed-gain PID configuration. Xie et al. [13] developed Model Predictive Control (MPC) based on NNARX (Neural Nonlinear AutoRegressive eXogenous) for an industrial temperature control unit and demonstrated a 10.3% reduction in total operating cost compared to a conventional PI controller. However, both solutions require computation beyond the embedded controller — the ANN in Almachi et al.'s approach runs on a host computer via a LabVIEW interface, while MPC requires a nonlinear optimization solver (IPOPT) — so neither can run directly on a resource-constrained microcontroller. Lightweight machine learning approaches such as TensorFlow Lite Micro enable model deployment on microcontrollers, but still require framework integration that adds firmware complexity, memory consumption, and external library dependencies [14].

Notably, the thermal dynamics of an egg incubator represent a relatively simple system — temperature changes slowly with a long time constant (on the order of hundreds of seconds) and operates at a single fixed operating point (37.5°C). In such quasi-linear systems, Multiple Linear Regression (MLR) has the potential to provide sufficiently accurate predictions without requiring complex neural network architectures. The regression coefficients produced by Ordinary Least Squares (OLS) can be deployed directly as multiply-add arithmetic operations in firmware — without an ML framework, without external library dependencies, and without significant memory overhead. This approach offers an implementation pathway that is far simpler and more easily reproducible by practitioners and farmers with limited technical resources.

Based on this background, this study proposes a predictive temperature control system based on linear regression in an IoT egg incubator using an ESP32 microcontroller without any machine learning framework. The MLR model is trained offline using historical internal temperature and heater duty cycle data, and the resulting coefficients are embedded directly into the firmware as a constant float array. A 30-second temperature change prediction is used to modify the proportional and derivative actions of the PID controller, while the integral action remains based on the actual temperature reading to guarantee steady-state error elimination. The system is equipped with real-time IoT monitoring via MQTT protocol to a cloud broker, operating in a non-blocking manner without interfering with the control cycle.

The main contributions of this study include: 1. A Multiple Linear Regression approach as a temperature predictor deployed directly as arithmetic operations in ESP32 firmware, without dependency on any machine learning framework. 2. Integration of predictive control with real-time IoT monitoring via MQTT on a single microcontroller. 3. A comparative evaluation between conventional PID and Regression-PID with non-parametric statistical testing (Mann-Whitney U and Wilcoxon Signed-Rank). 4. Measurement and analysis of computational latency through control computation time (lat_total_us) and MQTT publish time (lat_pub_us), quantifying the additional overhead of the predictive approach compared to conventional PID and confirming its real-time feasibility.

II. RESEARCH METHODS

2.1. System Architecture

The egg incubator system developed in this study is built on an ESP32 DevKit V1 microcontroller running Arduino framework-based firmware in the PlatformIO environment. The system architecture consists of three functional layers: sensor acquisition, control loop, and IoT communication. An SHT31 sensor (Sensirion, I2C address 0x44) is mounted inside the incubator chamber to measure internal temperature and relative humidity. This sensor has a typical accuracy of ±0.2°C over the 0–90°C range with repeatability of ±0.04°C in high-precision mode [15], making it highly suitable for detecting temperature variations within the incubation operating range (37–39°C). Empirical offset calibration is applied in the firmware to correct sensor reading deviation against a reference thermometer, and the correction value is stored persistently in the ESP32 Flash NVS (Non-Volatile Storage).

The heating element — a defrost heater — is controlled via a RobotDyn AC Light Dimmer module operating with zero-crossing detection at the 50 Hz mains frequency. This mechanism enables proportional heating power adjustment through a duty cycle of 0–100%, which is far smoother than binary switching systems based on relays. The zero-crossing signal is received on GPIO 27 as an interrupt, and TRIAC triggering is performed on GPIO 14 with a time delay calculated from the desired duty cycle percentage. Air circulation inside the incubator chamber is maintained by a 12V DC fan operating continuously while the system is active — the fan is connected directly to the power supply without microcontroller control, so heat distribution remains constantly uniform. A door sensor on GPIO 13 functions as

a safety interlock that automatically cuts heater power when the incubator door is opened.

A local user interface is provided through a 16×2 LCD display (I2C, address 0x27) showing the active operating mode, current temperature, setpoint, and duty cycle percentage. Mode selection is performed via a rotary encoder (GPIO 25/33/32), and the selected mode configuration is stored in Flash NVS to persist after reboot. Network connectivity is provided through the ESP32's integrated Wi-Fi module, connecting the device to an MQTT broker for telemetry and remote monitoring. The main control cycle runs every 1 second, while MLR model sampling and MQTT telemetry transmission occur every 5 seconds — an interval selected based on the relatively long time constant characteristic of the incubator thermal system (on the order of hundreds of seconds), allowing meaningful temperature changes to be detected between samples without overloading the microcontroller computation.

Table I. Hardware component specifications of the IoT egg incubator system.

COMPONENT	SPECIFICATION	ROLE
ESP32 DevKit V1	Xtensa LX6 dual-core 240 MHz, 520 KB SRAM, Wi-Fi 802.11 b/g/n	Main microcontroller
SHT31 (Sensirion)	I2C 0x44, accuracy ±0.2°C, repeatability ±0.04°C	Internal temperature and humidity sensor
Heating Element (Defrost Heater)	Resistive element, TRIAC-controlled	Primary heat source
AC Light Dimmer (RobotDyn)	Zero-crossing 50 Hz, TRIAC, PWM 0–100%	Proportional heating power control
DC Fan	12V, continuous operation	Air circulation
Door sensor	GPIO 13, active-low	Heater safety interlock
16×2 LCD	I2C 0x27	Local status display

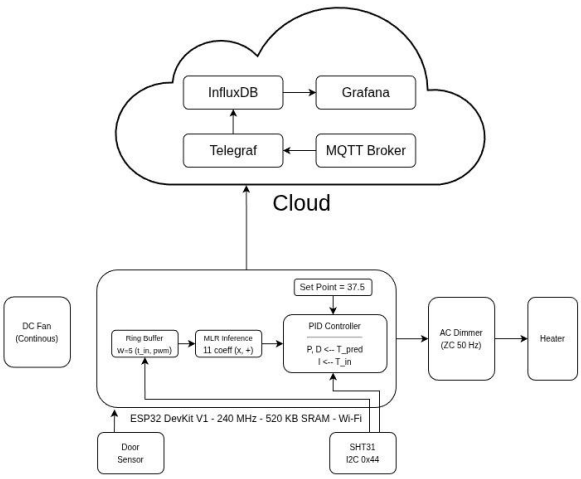

Figure 1. System architecture of the Regression-PID-based predictive temperature control system for an IoT egg incubator using ESP32.

The firmware provides two main operating modes used in this study: Conventional PID Mode and Regression-PID Mode. Both modes use identical hardware components, so any observed performance difference is attributable entirely to the difference in control algorithm.

2.2. Data Collection

The data used to train the linear regression model was collected independently and exclusively for this study, to avoid cross-study bias and ensure the external validity of the model. The acquisition process was performed using a Data Collection mode in the same firmware, in which the ESP32 runs PID control actively throughout the recording session following a structured protocol: a heating phase (cold start) from room temperature to the setpoint, 10-minute steady-state intervals per cycle, four door-opening disturbance events (door events) of ±30 seconds each with the heater automatically cut off by a firmware safety interlock, and PID recovery phases after each disturbance. The duty cycle variation that arises organically from the PID dynamics — ranging from 0% (door open) to 100% (initial heating), with an average of ≈27% at steady state — provides sufficient operational condition coverage for MLR model training.

Every 5 seconds, one data sample is recorded and transmitted via MQTT to an InfluxDB database on a cloud server, covering the three main variables listed in the following table.

Table II. Variables recorded every 5 seconds during the training data collection session.

Variable	Symbol	Unit	Source
Timestamp	*ts*	Unix seconds	NTP
Internal incubator temperature	*t_in*	°C	SHT31 (0x44)
Heater duty cycle	*pwm*	% (0–100)	*Firmware*

A total of 1,910 raw samples were collected in a single session lasting approximately 2.65 hours (9,546 seconds). Although only a single session, the data covers all relevant operational conditions: cold start from room temperature ≈30°C (148 samples), steady state near the 37.5°C setpoint (1,169 samples), door-opening disturbances including preparation and waiting phases (105 samples), and post-disturbance recovery (488 samples). The diverse condition coverage within a single recording session is sufficient to train the regression model across the relevant operating range — given that the incubator thermal system is quasi-linear and its dynamics are repetitive (stationary) around a single operating point.

Data preprocessing was performed offline in a Python environment. Data was sorted by timestamp, then one row with an invalid temperature reading ($t\_in \leq 10°C$) was removed — this threshold was chosen to eliminate sensor readings during device anomalies without discarding cold start phase data whose initial temperature was ≈30°C. The remaining data was then divided into continuous segments: any gap of more than 10 seconds between samples is treated as a segment boundary so that the sliding window is never constructed across discontinuous segments. From these continuous segments, feature samples were formed consisting of W consecutive historical values of t_in and

pwm, with the target being Δt = t_in[k + H] − t_in[k] — the temperature change over a horizon of H = 6 steps (30 seconds) ahead. All samples were then randomly split into a training set (80%) and test set (20%) using train_test_split with shuffle=True and random_state=42; randomization was performed at the window level so that temporal ordering within each window is preserved as input features, and the distribution of operational conditions is represented proportionally in both subsets.

2.3. Multiple Linear Regression Model

The predictive model used in this study is Multiple Linear Regression (MLR), which predicts the change in internal temperature (Δt) over a 30-second prediction horizon (6 steps × 5 seconds per step). The model formulation is defined as follows:

$$\Delta t = \beta_0 + \sum_{i=0}^{W-1} \beta_i \cdot t_{in}[k-i] + \sum_{j=0}^{W-1} \gamma_j \cdot pwm[k-j] \quad (1)$$

where β₀ is the *intercept*, β_i is the coefficient for the internal temperature history at step (k–i), γ_j is the coefficient for the duty cycle history at step (k–j), and W is the sliding window size. The total number of model coefficients is $2W + 1$.

The choice of Δt — rather than absolute temperature — as the target variable is a critical design decision. At steady state, the expected value of Δt approaches zero, so the model is isolated from absolute setpoint shifts and focused on predicting thermal deviations. The estimated future temperature is then computed as:

$$T_{pred} = t_{in}[k] + \Delta t \quad (2)$$

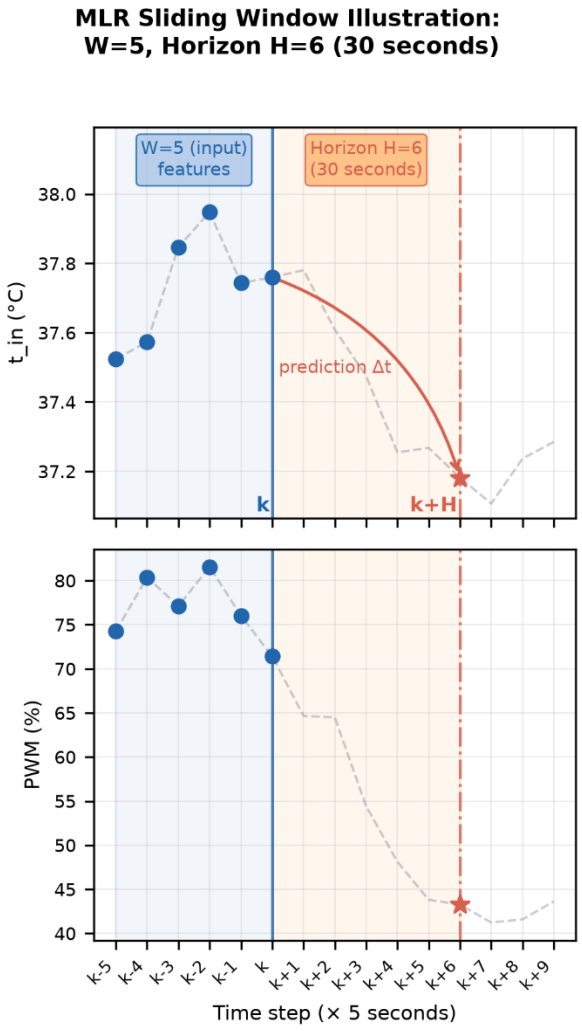

Figure 2. Illustration of the sliding window mechanism in the MLR model. The blue area shows $W = 5$ historical samples forming the input feature vector; the orange area shows the prediction horizon $H = 6$ steps (30 seconds). The red point *() is the target Δt** being predicted.

To determine the optimal sliding window size, a systematic experiment was conducted with variation $W \in \{5, 10, 15, 20\}$ samples, representing historical data of 25 to 100 seconds. Each configuration was trained using *Ordinary Least Squares* (OLS) in a Python environment (scikit-learn library) and evaluated based on two key metrics on validation data: **RMSE** (*Root Mean Square Error*), which measures the typical magnitude of prediction error, and **R²** (coefficient of determination), which measures the proportion of data variance explained by the model. The comparison of four configurations is presented in Figure 3.

International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 03, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

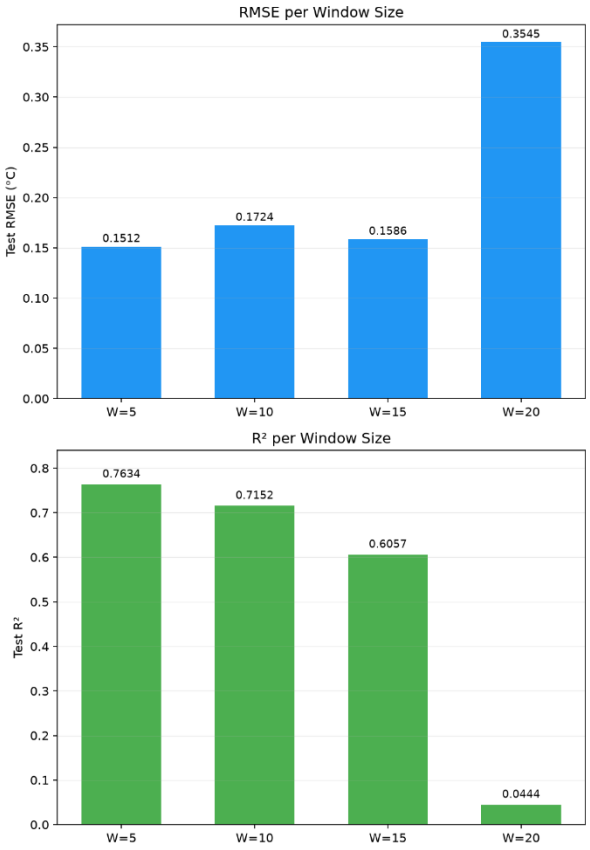

Figure 3. Comparison of RMSE and R² on test data for four window size variations. *W* = 5 yields the best generalization; *W* = 20 shows *overfitting* with R² test ≈ 0.04.

The window size configuration with the lowest validation RMSE was selected as the final model. The trained model coefficients were then exported as a float constant array and embedded directly into the ESP32 firmware code. The inference process on the microcontroller consists solely of $2W + 1$ multiply-add operations — without requiring a machine learning framework, without external library dependencies, and without dynamic memory allocation [14], [16]. The model memory footprint does not exceed $(2W + 1) \times 4$ bytes (float32 format), which for $W = 5$ (the selected final model) equals **44 bytes** — far below the ESP32 memory limit.

2.4. PID Integration

PID controller gain parameters were derived through systematic *plant* dynamics identification. An open-loop step test was executed to analyze the incubator's response around the 37.5°C operating point. The response data was estimated using a *First-Order Plus Dead Time* (FOPDT) model:

$$G(s) = \frac{K \cdot e^{-Ls}}{\tau s + 1} \quad (3)$$

The identified plant parameters were then used to compute PID gains via the SIMC (*Skogestad Internal Model Control*) method [17]. SIMC was selected for its advantage in lag-dominant processes ($\tau >> L$), where classical methods such as Ziegler-Nichols produce overly aggressive tuning and standard IMC ($\tau_i = \tau$) produces excessively slow steady-state error elimination. The SIMC formulation defines the restriction $\tau_i = \min(\tau, 4(\lambda + L))$ with coefficient λ as the sole tuning parameter controlling controller aggressiveness.

This study compares two control configurations using identical PID gains:

Conventional PID. Proportional, integral, and derivative actions are all calculated from the actual temperature reading (*t_in*):

$$u(t) = K_p \cdot e(t) + K_i \int e(t)\, dt + K_d \cdot \frac{de(t)}{dt} \quad (4)$$

where error *e*(*t*) = *T_setpoint* − *t_in*(*t*). The controller is reactive — control action is taken only after an error is detected.

Regression-PID. Proportional and derivative actions are calculated from the predictive temperature (*T_pred*) from the MLR model, while the integral action remains based on actual temperature (*t_in*):

$$u(t) = K_p \cdot e_{pred}(t) + K_i \int e_{actual}(t)\, dt + K_d \cdot \frac{de_{pred}(t)}{dt} \quad (5)$$

where *e_pred*(*t*) = *T_setpoint* − *T_pred*(*t*) and *e_actual*(*t*) = *T_setpoint* − *t_in*(*t*). With this architecture, the P and D components provide early "braking" action based on the 30-second temperature prediction — before the actual temperature exceeds the setpoint — while the I component continues to accumulate actual error to guarantee steady-state error elimination toward zero. If the input buffer is not yet full or sensor data is invalid, the system automatically falls back to conventional PID mode (Equation 4).

The use of identical gains in both configurations is a critical experimental design decision. By isolating the tuning parameters, any observed performance difference can be directly attributed to the predictive effect of the MLR model, rather than to differences in controller aggressiveness.

2.5. IoT Monitoring Architecture

The remote monitoring system is implemented using MQTT protocol over the ESP32's integrated Wi-Fi connection, with the architecture ESP32 → Mosquitto Broker → Telegraf → InfluxDB → Grafana running on a cloud Virtual Private Server (VPS). The ESP32 transmits JSON telemetry messages to the topic inkubator/telemetri every 5 seconds (synchronous with the MLR sampling interval), covering actual temperature, duty cycle, predicted temperature, control error, door status, and labels for operating mode and experiment scenario. A separate heartbeat message is published every 30 seconds to the topic inkubator/status for early detection of offline conditions or memory anomalies. QoS 0 protocol was selected to minimize latency and network overhead (936 bytes per transaction) [11], given that occasional loss of a single telemetry sample is not safety-critical.

All IoT communication is non-blocking — the control loop operates entirely locally on the ESP32 without depending on network stability. If the Wi-Fi or MQTT connection is interrupted, heater actuation continues normally and data will be forwarded once the connection is

restored. The ctrl_mode, session_id, and scenario labels are configured as tags in InfluxDB to enable data segmentation per control mode and experiment scenario without full-table scans.

To quantify the computational impact of adding MLR, three latency components are measured directly using micros() instructions and included in every telemetry payload; the third component (end-to-end latency) is computed on the backend side as the difference between the MQTT message reception time and the timestamp recorded when the sample was captured on the ESP32.

Table III. Latency components measured in the ESP32 firmware and cloud backend.

SYMBOL	DATA COLUMN	DEFINITION	MEASUREMENT METHOD
t_computation	lat_total_us	Control computation time — includes MLR inference (REG mode) and PID calculation	micros() at the start and end of the control computation block
t_mqtt	lat_pub_us	Time to publish the JSON payload to the MQTT broker	micros() before and after mqtt_publish()
t_e2e	lat_monitor_ms	End-to-end latency — the difference between the backend's MQTT message reception time and the device timestamp (timestamp_unix) when the sample was recorded on the ESP32	Backend time.Now() minus timestamp_unix × 1000

MLR inference overhead is derived indirectly as the difference in t_computation between REG-PID and conventional PID modes, not as a separate per-component measurement.

2.6. Experimental Protocol

To ensure an objective comparison between conventional PID and Regression-PID, all operational parameters were held constant: setpoint 37.5°C, identical PID gains (from SIMC identification), and equivalent environmental conditions across all sessions. Both modes were run alternately on the same hardware to avoid unit-to-unit variation.

Each control mode was tested in two independent sessions, for a total of four experimental sessions. Session PID-Run1 lasted 67.1 minutes with an initial temperature of 29.0°C, and session PID-Klasik lasted 88.7 minutes with an initial temperature of 32.1°C. For Regression-PID, session REG-Run1 lasted 70.9 minutes with an initial temperature of 29.4°C, and session REG-Regresi lasted 84.9 minutes with an initial temperature of 32.3°C. Each session included two door-opening events of ±30 seconds each as structured thermal disturbances, yielding a total of eight door events distributed evenly across modes. Session pairs with comparable initial temperatures (Run1 ≈ 29°C for both modes; Klasik/Regresi ≈ 32°C for both modes) enable transient response comparison under equivalent initial conditions.

Control performance was evaluated using five quantitative metrics defined in the following table.

Table IV. Control performance metrics for comparison between conventional PID and Regression-PID.

Metric	Definition
Overshoot (°C)	Maximum positive deviation above the setpoint after cold start
Settling time (minutes)	Time until temperature enters and remains within ±0.3°C of the setpoint
ISE (*Integral Square Error*)	$\int e^2(t)\,dt$ — sensitive to large errors
IAE (*Integral Absolute Error*)	$\int \lvert e(t)\rvert\,dt$ — measures total error accumulation
In-band occupancy (%)	Percentage of time temperature remains within ±0.3°C during steady state

Table V. Computational latency metrics and their reporting format.

METRIC	REPORTING FORMAT
t_computation (lat_total_us)	Mean ± std, max (µs), per mode
MLR inference overhead	Mean *t_computation* difference REG − PID (µs)
t_mqtt (lat_pub_us)	Mean ± std, max (ms), per mode
t_e2e (lat_monitor_ms)	Mean ± std, median, max (ms), per mode

Statistical significance of differences between the two control configurations was tested through a sequential procedure. A **Shapiro-Wilk** normality test was first applied to the ISE distribution per 1-minute time window to determine the appropriate comparative method. If the normality assumption is not met, **Mann-Whitney U** is used as the primary test (independent samples, H_1: ISE_PID > ISE_REG) and **Wilcoxon *Signed-Rank*** as a paired confirmation, with effect size reported using rank-biserial correlation (r) at significance level $\alpha = 0.05$.

III. RESULT AND ANALYSIS

3.1 Linear Regression Model Performance

The MLR model was trained using four window sizes ($W \in \{5, 10, 15, 20\}$) with a fixed prediction horizon H = 6 steps (30 seconds). A total of 1,910 raw samples were collected in DATACOL mode, covering cold start, steady state, and door disturbance phases; one row with an invalid temperature reading was removed during preprocessing, leaving 1,909 samples for use. After sliding window construction and 80:20 splitting with shuffle, a total of 1,899 active samples were obtained (n_train = 1,519, n_test = 380). Performance comparisons for the four configurations are shown in Table VI.

Table VI. Comparison of MLR model performance per window size on training and test data.

Window (*W*)	No. of Coefficients	RMSE Train (°C)	R^2 Train	RMSE Test (°C)	R^2 Test
5	11	0.1826	0.7772	**0.1512**	**0.7634**
10	21	0.1747	0.7866	0.1724	0.7152
15	31	0.1751	0.7873	0.1586	0.6057
20	41	0.1501	0.8081	0.3545	0.0444

W = 5 yields the best generalization with R^2 test = 0.7634 and RMSE test = 0.1512°C. Conversely, *W* = 20 shows clear overfitting: R^2 train is high (0.8081) but R^2 test is only 0.0444, meaning the model has almost no predictive capability on new data. Increasing the number of coefficients from 11 (*W* = 5) to 41 (*W* = 20) does not improve generalization — it degrades it. Model *W* = 5 was selected as the final model because it yields the optimal balance between prediction accuracy, model complexity, and memory efficiency.

The final *W* = 5 model is formulated as:

$$\Delta t = \beta_0 + \sum_{i=0}^{4} \beta_i \cdot t_{in}[k-i] + \sum_{j=0}^{4} \gamma_j \cdot pwm[k-j] \quad (6)$$

$$T_{pred} = t_{in}[k] + \Delta t \quad (7)$$

with 11 *float32* coefficients deployed as a static array in the ESP32 firmware. Total memory consumption is 44 bytes of flash (coefficients) and 40 bytes of RAM (two-channel *ring buffer* × 5 samples), qualifying it as a practically *zero-dependency* model.

Evaluation of prediction accuracy under actual operating conditions was performed using the t_pred column from IoT telemetry. From 1,862 active samples (t_pred > 0), an overall MAE = 0.1479°C and RMSE = 0.4302°C were obtained. The larger deviation compared to offline evaluation is caused by transient phases and door disturbances that exhibit dynamics outside the training distribution. Under steady-state conditions (t_in > 37°C, door = 0), accuracy improves significantly: the mean T_pred − t_in difference = +0.0053°C with standard deviation 0.1006°C, confirming that the MLR model can accurately track the thermal system dynamics near the incubation operating point. The prediction accuracy visualization is presented in Figure 4.

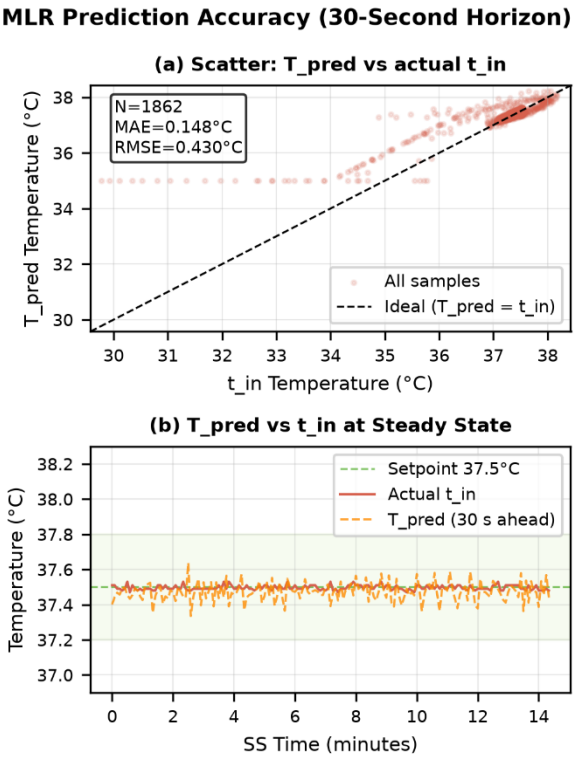

Figure 4. Prediction accuracy of the MLR model (*W* = 5, 30-second horizon) on actual experimental data. (a) Scatter plot of T_pred vs. actual t_in: the majority of points are close to the ideal line (y = x), especially in the steady-state range 37–38°C. (b) Time series of T_pred and t_in during steady state: the prediction follows the actual signal with a mean bias of +0.005°C and std of 0.1006°C.

3.2 Computational Latency Analysis

Two latency components were measured directly in the ESP32 firmware using micros() and included in every telemetry payload: lat_total_us, which records control computation time (including MLR inference in REG-PID mode), and lat_pub_us, which records the time to publish the JSON payload to the MQTT broker.

Table VII. Comparison of control computation latency (lat_total_us) for PID vs. Regression-PID.

Metric	Conventional PID	Regression-PID
Mean (µs)	3.09	15.26
Median (µs)	3.0	15.0
Std (µs)	0.73	0.93
Max (µs)	14.0	23.0
N samples	1,851	1,870
MLR Overhead (REG − PID)	—	**+12.17 µs**

The computational overhead of +12.17 µs represents the mean difference in lat_total_us between REG-PID and conventional PID modes, reflecting the computation cost of 11 multiply-add operations of MLR coefficients in float32 format. This value is equivalent to 0.0012% of the 1-second control cycle period and is far below the 1 ms critical threshold required in real-time embedded control systems [18]. The latency distribution in both modes is extremely

narrow (std < 1 µs), indicating good computational determinism — consistent with the nature of arithmetic computation without dynamic memory allocation, garbage collection, or interpreter overhead commonly found in ML framework-based approaches [14], [16].

MQTT publish latency (lat_pub_us) was 4.68 ± 0.90 ms for PID and 4.66 ± 0.86 ms for REG-PID — practically identical (difference < 0.02 ms), confirming that the addition of MLR has no impact on the IoT communication path.

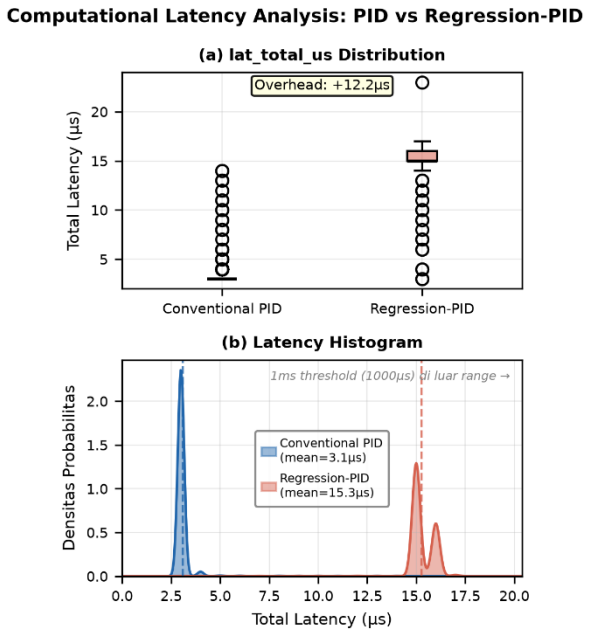

Figure 5. Analysis of control computation latency (lat_total_us). (a) Boxplot showing median PID = 3 µs and REG-PID = 15 µs with very narrow distributions in both modes. (b) KDE curve confirming all samples are far below the 1 ms threshold.

In addition to local computation latency, end-to-end latency (t_e2e) is measured as the difference between the time the ESP32 records a sample (timestamp_unix) and the time the backend service receives the corresponding MQTT message from the broker, computed as lat_monitor_ms. A summary of six statistical measures across all samples per control mode is presented in Table VIII.

Table VIII. Comparison of end-to-end latency (lat_monitor_ms) for PID vs. Regression-PID.

METRIC	CONVENTIONAL PID	REGRESSION-PID
Mean (ms)	598.9	282.3
Median (ms)	598.0	404.0
Std (ms)	850.7	219.6
Min (ms)	76	18
Max (ms)	14,466	1,737
N samples	1,851	1,870

Mean end-to-end latency is substantially larger and more variable than local computation latency — consistent with its nature of spanning the full network path from the ESP32 to the cloud backend, rather than computation confined to the microcontroller alone. Two methodological limitations should be noted: first, the ESP32's timestamp_unix has second-level resolution, introducing quantization of up to ±500 ms into the measured t_e2e values; second, the maximum value for conventional PID (14.466 seconds) is an extreme outlier caused by a transient Wi-Fi/MQTT disruption that pushes the standard deviation (850.7 ms) above the mean, and does not represent typical behavior. This end-to-end latency purely affects the freshness of data on the monitoring dashboard, not the real-time feasibility of the control system itself, which has already been confirmed via lat_total_us in Table VII.

3.3 Control Performance Comparison

Control performance was evaluated using four experimental sessions: two sessions per mode (Conventional PID and Regression-PID), with a total valid data duration of approximately 156 minutes per mode. The analysis covers three phases — transient response from cold start, stability under steady-state conditions, and post-door disturbance recovery — each testing a different aspect of the control system performance.

Transient response comparison was performed on Run 1, which had comparable initial conditions — PID initial temperature = 28.98°C and REG-PID = 29.41°C (difference < 0.5°C). The full warm-up profile is presented in Figure 6, while Table IX summarizes the quantitative metrics.

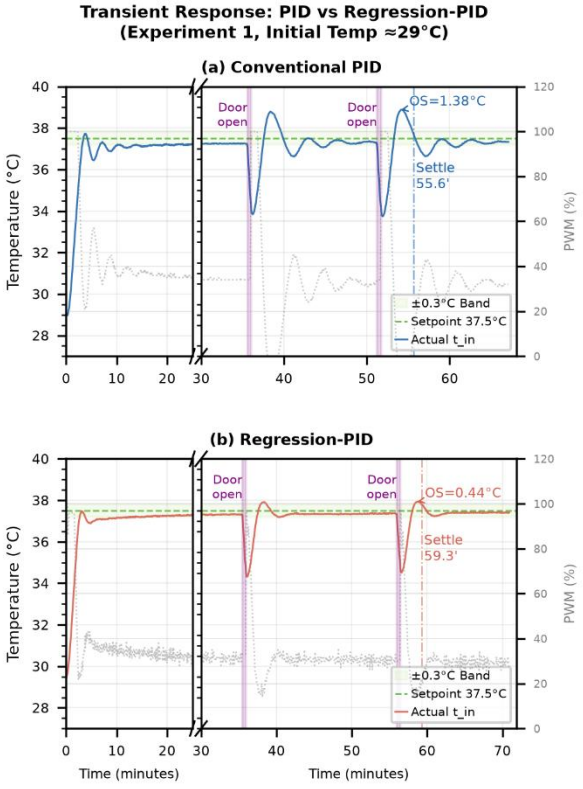

Figure 6. Cold start temperature profile of conventional PID (a) and Regression-PID (b) in Experiment 1 (initial condition ≈29°C). Purple areas = door-open periods; vertical dashed lines = settling time. PWM (%) is shown on the right Y-axis (gray scale).

Table IX. Transient response metrics for Experiment 1 (comparable initial conditions).

Metric	Conventional PID	Regression-PID	Change
Initial temperature (°C)	28.98	29.41	—
Overshoot (°C)	1.38	0.44	↓ **68.1%**
Overshoot (%)	3.68%	1.17%	↓ 68.2%
Settling time (minutes)	55.6	59.3	+6.7%

Regression-PID reduced overshoot by 68.1% (0.44°C vs. 1.38°C) — a direct consequence of the prediction mechanism: P and D components calculated from T_pred anticipate temperature rise before it exceeds the setpoint, so heater power reduction is initiated earlier. In conventional PID, correction only occurs after the error is detected, and the thermal inertia of the system results in larger overshoot. Settling time for REG-PID (59.3 minutes) is slightly longer than PID (55.6 minutes), but the 3.7-minute difference is not practically significant since both modes reach stable conditions within less than 1 hour from cold start.

Under steady-state conditions, the analysis used pooled data from two sessions per mode, with evaluation windows starting 10 minutes after the last disturbance event to eliminate post-disturbance transition periods. Total evaluated durations were PID = 22.5 minutes and REG-PID = 18.9 minutes. Results are summarized in Table X, with time-series visualization in Figure 7 and cumulative metric comparison in Figure 8.

Table X. Steady-state performance metrics from pooled data of two sessions per mode (true steady state).

Metric	Conventional PID	Regression-PID	Change
Mean error (°C)	+0.0821	+0.0251	↓ **69.4%**
Std *t_in* (°C)	0.0534	0.0139	↓ **73.9%**
ISE (°C²·s)	6.92	0.71	↓ **89.7%**
IAE (°C·s)	57.35	14.59	↓ **74.6%**
In-band ±0.3°C (%)	100.0%	100.0%	same

Regression-PID shows consistent superiority across all integral error metrics. ISE that is 89.7% lower reflects a drastic reduction in quadratic deviation, while the 73.9% lower standard deviation of t_in (0.014°C vs. 0.053°C) indicates far greater temperature stability — biologically relevant because excessive fluctuations can reduce hatchability [2]. Nevertheless, both modes maintained temperature within the ±0.3°C band 100% of the time, meaning the difference lies in the degree of stability, not in the ability to maintain the setpoint absolutely.

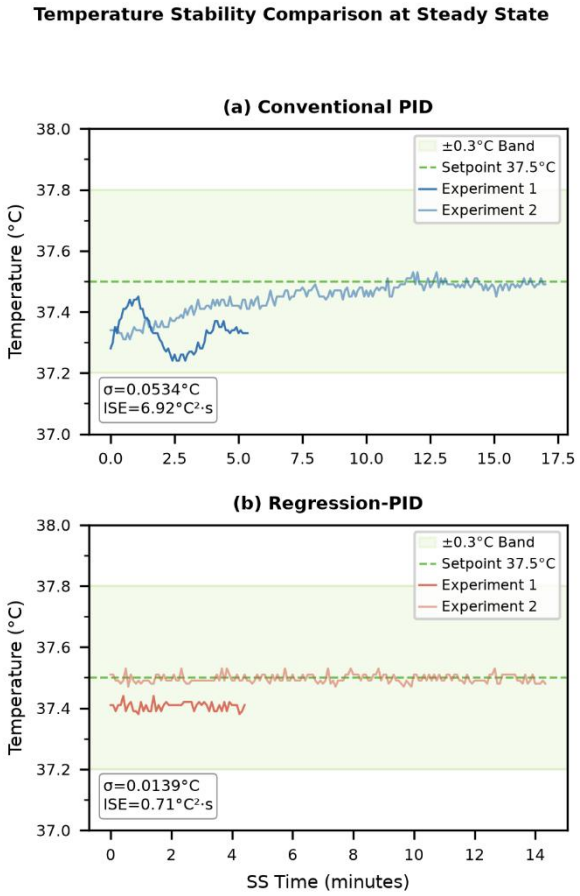

Figure 7. Comparison of temperature signals under steady-state conditions between conventional PID and Regression-PID. The Y-axis is expanded to the ±0.3°C band around the setpoint (37.5°C). PID fluctuations (σ = 0.053°C) are far larger than Regression-PID (σ = 0.014°C). Pooled data from two sessions per mode.

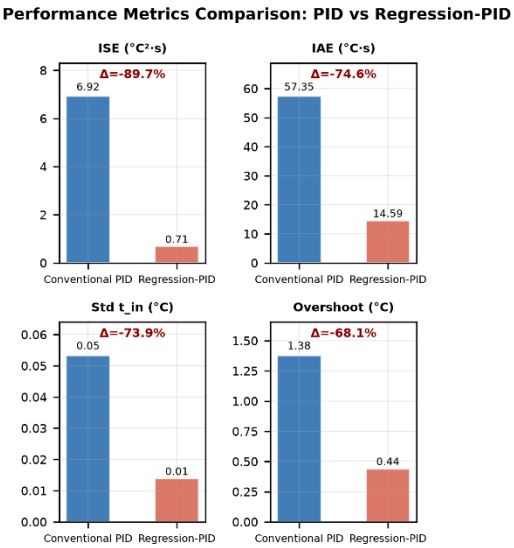

Figure 8. Comparison of cumulative performance metrics for conventional PID vs. Regression-PID. ISE and IAE were calculated from the true steady state period (two sessions pooled); overshoot from Experiment 1. All metrics show significant reductions in Regression-PID (Δ ISE = −89.7%; Δ IAE = −74.6%).

The third aspect evaluated is the response to door disturbances, simulated by opening the incubator door for ±30 seconds twice per session, yielding a total of four events per mode. The recovery profiles aligned at the door-close moment are shown in Figure 9, while Table XI summarizes the quantitative metrics.

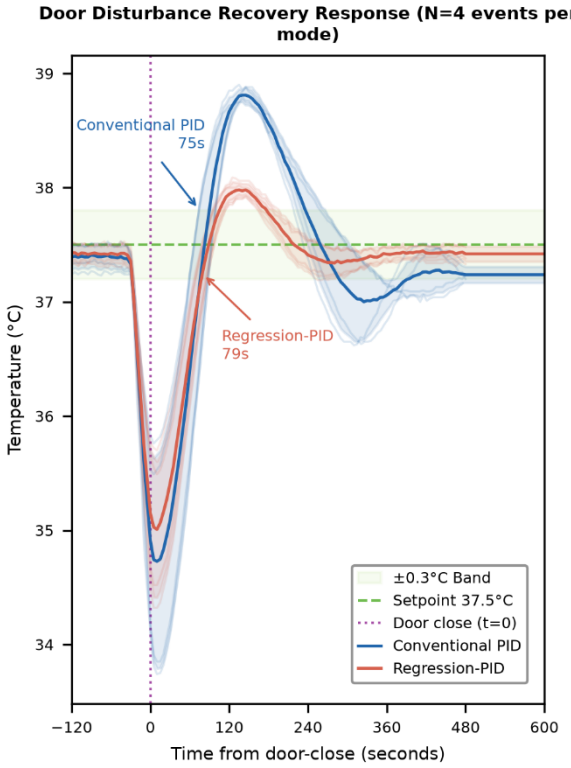

Figure 9. Post-door-disturbance temperature recovery profiles (N = 4 events per mode). The X-axis is normalized at the door-close moment (t = 0 seconds). Both modes show comparable recovery times (~75–79 seconds); the main difference lies in the narrower spread of REG-PID curves.

Table XI. Door disturbance response metrics (N = 4 events per mode, mean ± std).

Metric	Conventional PID	Regression-PID
Mean *undershoot* (°C)	2.59 ± 0.78	2.35 ± 0.57
Mean recovery time (seconds)	75.0 ± 15.0	78.8 ± 8.9

Both modes show comparable recovery times (75.0 vs. 78.8 seconds) with no practically meaningful difference. This is consistent with the prediction horizon limitation: door opening causes a sudden temperature drop that lies entirely outside the training distribution, so T_pred provides no effective anticipation and both modes operate reactively during the recovery phase. REG-PID undershoot is slightly smaller (2.35 vs. 2.59°C) with lower variability (std 0.57 vs. 0.78°C), indicating better response consistency but not greater recovery speed.

3.4 Statistical Significance Testing

Before selecting the comparative test method, the normality of the ISE distribution per 1-minute time window was assessed using the Shapiro-Wilk test. The results show that the ISE distribution does not satisfy the normality assumption for either mode — ISE PID yielded $W = 0.7569$ ($p = 0.0001$) and ISE REG-PID yielded $W = 0.5894$ ($p < 0.0001$). This non-normality is expected given the highly right-skewed nature of the ISE distribution: most windows have ISE close to zero when temperature is within the band, while a small number have high ISE due to post-disturbance transients. Since the normality assumption is not met, parametric tests are not appropriate and non-parametric tests were selected instead. All test results are summarized in Table XII.

Table XII. Summary of normality test results (Shapiro-Wilk) and non-parametric comparative tests (Mann-Whitney U and Wilcoxon Signed-Rank) for the ISE distribution per time window.

Test	Statistic	*p*-value	Conclusion
Shapiro-Wilk ISE PID	W = 0.7569	0.0001	Not normal
Shapiro-Wilk ISE REG	W = 0.5894	< 0.0001	Not normal
Mann-Whitney U (ISE)	U = 373.0	**0.0009**	Reject H_0
Wilcoxon *Signed-Rank*	W = 210.0	**< 0.0001**	Reject H_0

The Mann-Whitney U test compared the ISE distributions between both modes as independent samples (H_1: ISE_PID > ISE_REG, $\alpha = 0.05$). The result $p = 0.0009 < 0.05$ indicates a statistically significant difference, reinforced by a rank-biserial correlation effect size $r = 0.777$ classified as very large (Cohen's scale: > 0.5 = large, > 0.7 = very large). Thus, the difference is not only statistically significant but also practically substantive. The Wilcoxon Signed-Rank test as paired confirmation (N = 20 paired windows) yielded $p < 0.0001$, confirming that Regression-PID consistently produces lower ISE in every paired time window.

3.5 Analysis

Results show that the MLR model with W = 5 (11 coefficients) is sufficiently accurate for use in predictive control, with $R^2 = 0.7634$ on validation data. The adequacy of this simple model is consistent with the quasi-linear nature of the incubator thermal system: within a narrow operating range (±5°C around the 37.5°C setpoint), the relationship between temperature history, heating power, and future temperature change can be well-approximated linearly, as demonstrated by a similar MLR model for indoor temperature prediction of mobile containers achieving an error of only ±7.1% [19]. A similar approach using NNARX for MPC also reported adequate accuracy at short horizons [13], but requires a more complex neural network architecture.

The primary advantage of MLR in this context is the absence of framework dependency: coefficients can be stored as a float32 array and executed with 11 multiply-add operations, without requiring TFLite Micro, CMSIS-NN, or any runtime — fundamentally different from TinyML approaches that require quantization, tensor inference, and dynamic memory allocation [14], [16]. lat_total_us data confirms this advantage quantitatively: the control computation time difference between REG-PID and PID is +12.17 µs (15.26 vs. 3.09 µs, each from N > 1,800 samples),

with standard deviation < 1 µs in both modes — demonstrating computational determinism characteristic of arithmetic operations without runtime overhead.

On the other hand, the findings from the door disturbance phase must be interpreted honestly. The ineffectiveness of REG-PID in accelerating recovery (78.8 vs. 75.0 seconds) is a predictable consequence of the system design: door opening causes a temperature drop at a rate > 5°C/minute, while the MLR model was trained on a Δt distribution dominated by gentle changes (|Δt| < 1°C per 30 seconds). T_pred under these conditions produces out-of-domain estimates, rendering the P and D component contributions unreliable, with the I component from actual t_in maintaining the basic control response. lat_total_us data simultaneously shows that REG-PID continues to execute MLR computation during the disturbance phase at a consistent +12.17 µs overhead — a cost incurred without yielding any predictive benefit. This limitation is inherent to linear regression with bounded domain and is not unique compared to other predictive methods with bounded horizons [12].

From a biological incubation perspective, the Regression-PID performance profile is more conducive to successful hatching. The optimal temperature for chicken embryos lies in the 37.5–37.8°C range [2], and deviations outside this range — especially toward excess heat — can reduce hatchability and damage embryos [3]. PID overshoot of 1.38°C (peak 38.88°C) places the incubator at a critical temperature for several minutes, while REG-PID overshoot of 0.44°C (peak 37.94°C) remains within biological tolerance. The higher steady-state stability (Std = 0.014°C vs. 0.053°C) also supports thermal environment consistency that correlates with better hatch rates [2], [9].

IV. CONCLUSION

This study proposed and evaluated a Regression-PID architecture — a predictive temperature controller based on Multiple Linear Regression (MLR) deployed as bare-metal arithmetic in ESP32 firmware without machine learning framework dependencies. A comparative experiment against conventional PID was conducted on an IoT egg incubator with real-time MQTT telemetry and computational latency measurement via lat_total_us and lat_pub_us.

The MLR model with window size W = 5 (11 coefficients, 44 bytes of flash) proved adequate as a temperature predictor, yielding $R^2 = 0.7634$ and RMSE = 0.1512°C on validation data — confirming that linear regression, without neural network layers, quantization, or a runtime framework, is sufficient for short-horizon thermal prediction in a quasi-linear system. The zero-dependency deployment carries a computational overhead of only +12.17 µs per cycle (0.0012% of the 1-second control period) with fully deterministic latency (std < 1 µs), confirming real-time feasibility without sacrificing embedded determinism [18].

End-to-end latency from the device to the cloud backend is substantially larger and more variable than local computation latency — averaging 598.9 ms (PID) and 282.3 ms (Regression-PID), with a wide spread caused by non-constant Wi-Fi/MQTT network conditions. This characteristic purely affects the freshness of data on the monitoring dashboard and does not impact the real-time feasibility of the local control loop, since the system architecture is designed to be non-blocking, allowing heater actuation to continue independently of network conditions.

Comparative evaluation showed that Regression-PID significantly outperforms conventional PID in both transient and steady-state phases: overshoot reduced by 68.1% (0.44°C vs. 1.38°C), ISE by 89.7% (0.71 vs. 6.92 °C²·s), IAE by 74.6% (14.59 vs. 57.35 °C·s), and steady-state standard deviation by 73.9% (0.014 vs. 0.053°C). Both modes maintained temperature within the ±0.3°C tolerance band 100% of the time, but the far smaller fluctuations of Regression-PID are biologically significant given the optimal embryo temperature range of 37.5–37.8°C [2]. Under door-opening disturbances, recovery times were comparable (75.0 vs. 78.8 seconds), a predictable consequence of the model's domain boundary against large impulsive dynamics. The difference in ISE distribution between the two modes under steady-state conditions was confirmed statistically via Mann-Whitney U ($p = 0.0009$, $r = 0.777$ — very large effect) and Wilcoxon Signed-Rank ($p < 0.0001$).

These findings are subject to several limitations: the experiment was conducted on a single incubator unit, the model assumes a fixed 37.5°C setpoint, and statistical power is constrained by two sessions per mode (N = 20–24 windows). Future work should address multi-setpoint robustness, online coefficient adaptation via recursive least squares, biological validation through active egg incubation trials, and a comparison against FOPDT-based Internal Model Control to quantify the contribution of the data-driven approach relative to a physics-based baseline.

THANK-YOU NOTE

The author expresses gratitude to Institut Teknologi Bisnis AAS Indonesia for support through facilities and the academic environment during this research. Thanks are also extended to the entire academic community of the Informatics Engineering Study Program for guidance and constructive feedback during the research and scientific writing process.

REFERENCES

[1] BPS, "Populasi Unggas Menurut Provinsi dan Jenis Unggas (ekor)." Diakses: 25 Juni 2026. [Daring]. Tersedia pada: https://www.bps.go.id/id/statistics-table/3/Y2tKeVZYUk1UMDVNV1ROcGFXOW1kblZzZUZrMFp6MDkjMw==/populasi-unggas-menurut-provinsi-dan-jenis-unggas--ekor---2023.html?year=2023

[2] S. YALCIN, S. Özkan, dan T. Shah, "Incubation Temperature and Lighting: Effect on Embryonic Development, Post-Hatch Growth, and Adaptive Response," *Front. Physiol.*, vol. 13, no. May, hal. 1–16, Mei 2022, doi: 10.3389/fphys.2022.899977.

[3] E. Iraqi, A. A. Hady, N. Elsayed, H. Khalil, A. El-Saadany, dan K. El-Sabrout, "Effect of thermal manipulation on embryonic development, hatching process, and chick quality under heat-stress conditions,"

Poult. Sci., vol. 103, no. 1, hal. 103257, Jan 2024, doi: 10.1016/j.psj.2023.103257.

[4] F. Peprah, S. Gyamfi, M. Amo-Boateng, E. Buadi, dan M. Obeng, "Design and construction of smart solar powered egg incubator based on GSM/IoT," *Sci. African*, vol. 17, hal. e01326, 2022, doi: 10.1016/j.sciaf.2022.e01326.

[5] F. A. Septian *et al.*, "Simulasi Pengendalian Suhu pada Inkubator Penetasan Telur Ayam Menggunakan Arduino Berbasis PID," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 13, no. 3, hal. 238, Jul 2025, doi: 10.23960/jitet.v13i3.6741.

[6] M. C. A. Prabowo, I. Sayekti, S. Astuti, S. T. Nursaputro, dan S. Supriyati, "Development of an IoT-Based Egg Incubator with PID Control System and Web Application," *JOIV Int. J. Informatics Vis.*, vol. 8, no. 1, hal. 465, Mar 2024, doi: 10.62527/joiv.8.1.2044.

[7] G. Pandey, B. Pokhrel, S. Budhathoki, dan S. K. Mandal, "IoT based PID Control Egg Incubator," *Int. J. Educ. Pract. Eng.*, vol. 2, no. 3, hal. 36–44, Jun 2025, doi: 10.70504/ijepe.v2i3.14313.

[8] W. K. Simangunsong, C. T. H. Manurung, dan I. K. L. N. Suciningtyas, "Rancang Bangun Alat Penetas Telur dengan Sistem Pengendali Suhu dan Kelembapan Menggunakan Metode Fuzzy dan Monitoring Berbasis IoT," *J. Integr.*, vol. 17, no. 1, hal. 70–79, 2025, doi: 10.30871/ji.v17i1.9478.

[9] R. C. Maaño, R. A. Maaño, P. J. De Castro, E. P. Chavez, S. C. De Castro, dan C. D. Maligalig, "SmartHatch: An Internet of Things–Based Temperature and Humidity Monitoring System for Poultry Egg Incubation and Hatchability," in *2023 11th International Conference on Information and Communication Technology (ICoICT)*, IEEE, Agu 2023, hal. 178–183. doi: 10.1109/ICoICT58202.2023.10262810.

[10] K. Il Ko dan M. H. Lee, "MQTT-Based Architecture for Real-Time Data Collection and Anomaly Detection in Smart Livestock Housing," *Sensors*, vol. 25, no. 23, 2025, doi: 10.3390/s25237186.

[11] M. Has, D. Kreković, M. Kušek, dan I. Podnar Žarko, "Efficient Data Management in Agricultural IoT: Compression, Security, and MQTT Protocol Analysis," *Sensors*, vol. 24, no. 11, 2024, doi: 10.3390/s24113517.

[12] J. C. Almachi, R. Vicente, E. Bone, J. Montenegro, E. Cando, dan S. Reina, "Implementation of a Neural Network for Adaptive PID Tuning in a High-Temperature Thermal System," *Energies*, vol. 18, no. 12, hal. 3113, 2025, doi: 10.3390/en18123113.

[13] J. Xie, L. Simpson, J. Asprion, dan R. Scattolini, "A Learning-based Model Predictive Control Scheme with Application to Temperature Control Units *," in *2024 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, Agu 2024, hal. 675–680. doi: 10.1109/CCTA60707.2024.10666571.

[14] R. David *et al.*, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," 2021, doi: 10.48550/arXiv.2010.08678.

[15] A. Sensirion, "Datasheet SHT3x-DIS Humidity and Temperature Sensor," 2022. [Daring]. Tersedia pada: www.sensirion.com

[16] L. Capogrosso, F. Cunico, D. S. Cheng, F. Fummi, dan M. Cristani, "A Machine Learning-Oriented Survey on Tiny Machine Learning," *IEEE Access*, vol. 12, hal. 23406–23426, 2024, doi: 10.1109/ACCESS.2024.3365349.

[17] S. Skogestad, "Simple analytic rules for model reduction and PID controller tuning," *Model. Identif. Control A Nor. Res. Bull.*, vol. 25, no. 2, hal. 85–120, 2004, doi: 10.4173/mic.2004.2.2.

[18] H. Kopetz, "The Real-Time Environment," in *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Cham: Springer International Publishing, 2011, hal. 1–28. doi: 10.1007/978-1-4419-8237-7_1.

[19] Z. Patonai, R. Kicsiny, dan G. Géczi, "Multiple linear regression based model for the indoor temperature of mobile containers," *Heliyon*, vol. 8, no. 12, 2022, doi: 10.1016/j.heliyon.2022.e12098.