Journal of Soft Computing Exploration
Homepage: htttps://shmpublisher.com/index.php/joscex
p-ISSN: 2746-7686
e-ISSN: 2746-0991

.

# Lightweight discrete q-learning for self-tuning PID on ESP32: Robustness evaluation and cross-volume adaptation in egg incubators

**Tino Feri Efendi[1], Zainal Arifin[2]**
[1,2]Department of Informatics, Institut Teknologi Bisnis AAS Indonesia, Indonesia

**Article Info**

***Article history:***

Received Mar 10, 2026
Revised Apr 03, 2026
Accepted May 15, 2026

***Keywords:***

Adaptive PID control
Discrete q-learning
Edge-AI
Egg incubator
Self-tuning
Thermal system

**ABSTRACT**

Temperature stability is the most crucial factor in the success of the egg incubation process. The use of conventional Proportional–Integral–Derivative (PID) control with static Ziegler–Nichols tuning often fails to adapt to external disturbances and thermal dynamics, leading to temperature overshoot that can be fatal to embryo survival. This study proposes the implementation of an adaptive PID controller using a Discrete Q-Learning method based on Edge-AI on an ESP32 microcontroller. Experimental results under standard conditions show that the Q-Learning method successfully reduces overshoot by up to 81.8%, limiting the temperature spike to only 0.2°C above the target of 38.0°C, and accelerating the stabilization time by 76.9% with a reduction in IAE of 52.5%. In the dynamic disturbance rejection test, the adaptive system validated the algorithm's robustness against dynamic disturbances. Furthermore, cross-environment adaptation evaluation by reducing the incubator volume by 50% demonstrates the agent's autonomous adaptation capability, eliminating overshoot entirely (0.000°C) without parameter recalibration and reducing IAE by 55.1% compared to static PID. This study concludes that the implementation of Q-Learning on low-cost hardware produces a robust, precise, and autonomously adaptive thermal control system for agricultural technology applications.



***Corresponding Author:***

Tino Feri Efendi,
Department of Informatics,
Institut Teknologi Bisnis AAS Indonesia,
Jl, Slamet Riyadi No. 361 Windan, Makamhaji, Kartasura, Sukoharjo, Jawa Tengah, Indonesia
Email: tinoferi8@gmail.com
https://doi.org/10.52465/joscex.v7i2.8

## 1. INTRODUCTION

Stable temperature is the most crucial factor in the success of the egg incubation process in artificial incubators. Temperature fluctuations that exceed tolerance limits, particularly initial temperature overshoot, can cause high levels of thermal stress and potentially lead to early embryo mortality [1]–[4]. In practice, temperature control in egg incubators has generally shifted from simple ON/OFF methods to conventional Proportional–Integral–Derivative (PID) control using static tuning methods such as Ziegler–Nichols [5]. Although it provides a smoother response, thermal systems possess nonlinear characteristics and significant

dead-time. This causes static PID controllers to frequently fail in adapting to external environmental disturbances [6], [7] or changes in thermal load capacity, ultimately leading to overall system instability.

Various efforts have been made to overcome the limitations of static PID through integration of intelligent computational techniques. The Particle Swarm Optimization (PSO) method has been proven capable of minimizing overshoot in industrial heating furnace systems [8], however this approach is offline – parameters are optimized once before the system operates and cannot adapt to changes in system dynamics in real-time. Artificial Neural Network and Fuzzy Logic have also been widely implemented for dynamic PID tuning in vacuum pressure stabilization systems [9], second-order mechanical systems [10], [11] and even temperature control in oyster mushroom incubators [12], [13]. Although effective in well-characterized environments, these methods have distinct fundamental limitations: Neural Networks require large historical training datasets and computationally intensive offline training; rule-based Fuzzy systems require deep expert knowledge for design and must be manually redesigned when system characteristics change; while Deep Reinforcement Learning (DRL) such as DQN requires millions of training episodes in simulation environments and computational infrastructure far exceeding the capacity of low-power microcontrollers [7], [14], [15]. These limitations collectively make all three approaches impractical for direct implementation on embedded hardware such as ESP32 operating in remote environments without cloud computing connectivity.

There exists a significant research gap at the intersection of three simultaneously unmet needs: (1) an adaptive control system capable of operating online without historical training data, (2) computational and memory loads light enough to run directly on low-cost microcontrollers, and (3) the ability to adapt to changes in environmental characteristics without manual recalibration procedures. Existing research generally satisfies only one or two of these three needs simultaneously. Therefore, this study proposes the application of *Discrete Q-Learning* as a PID parameter self-tuning mechanism [16], [17] that satisfies all three needs simultaneously: learning online through direct interaction with the physical system, operating within a very compact memory footprint (Q-Table 15×7 = 105 float values), and capable of autonomously adapting to changes in the incubator's thermal capacity without operator intervention.

The selection of Discrete Q-Learning as the self-tuning method in this study is based on a comparative analysis of available alternative methods. Table 1 summarizes the comparison of technical characteristics of five relevant intelligent computational approaches based on the most critical criteria for this embedded implementation context.

Table 1. Comparison of self-tuning PID method characteristics on constrained hardware

| Criteria | PSO | Neural Network | Fuzzy Adaptive PID | DQN / Deep RL | Discrete Q-Learning (This Study) |
|---|---|---|---|---|---|
| Training Data Requirement | Large historical dataset | Large dataset + labels | Manual expert rules [18]–[20] | Millions of simulation episodes | Not required (online learning) |
| Offline Training | Mandatory | Mandatory | Manual design | Mandatory (simulation) | Not required |
| RAM Memory | High (particle population) | Very high (network weights) | Medium (rule base) | Very high (DNN model) | Very low (15×7 = 105 floats) |
| ESP32 Compatibility | Difficult | Impractical | Possible | Not feasible | Proven operational |
| Online Adaptation | No | Limited | Static after design [18]–[20] | Yes (requires GPU) [7], [21] | Fully online |
| Recalibration when Environment Changes | Requires re-run | Requires retraining | Requires rule base redesign | Requires retraining | Automatic adaptation |
| Policy Interpretability | Low | Very low (black box) | High (rule base) | Very low (black box) | High (Q-Table inspectable) |

Based on the analysis in Table 1, Discrete Q-Learning was selected because it satisfies all criteria simultaneously. Unlike PSO and Neural Networks that require historical training data and offline training processes, the Q-Learning agent learns online through direct interaction with the physical environment — making it model-free and independent of historical dataset availability. Compared to Fuzzy Adaptive PID, the Q-Learning approach does not require expert knowledge to design rule bases and can automatically discover optimal policies through exploration. Compared to Deep Reinforcement Learning (DRL/DQN) methods, the tabular representation of Q-Learning with 15 states and 7 actions produces a Q-Table matrix of only 105 float values (420 bytes), which can be stored entirely within the ESP32 microcontroller's RAM with a very comfortable margin. The interpretability advantage of the policy — where each Q-Table entry can be inspected and verified — also makes Discrete Q-Learning more suitable for safety-critical applications such as embryo incubation, where system behavior must be predictable and auditable.

Considering the broad scope of intelligent system engineering, the scope of this research is limited to the implementation of Discrete Q-Learning for updating proportional, integral, and derivative parameters. The experiment is restricted to controlling an Alternating Current (AC) heating element using a zero-cross dimmer

actuator with feedback from an SHT31 temperature sensor. The system operates within incubator volume variations up to a maximum of 64 liters and does not incorporate humidity regulation into the intelligent control computation.

## 2. METHOD

### Hardware Architecture and Communication System

This study employs an Edge Computing architecture in which artificial intelligence processing is executed locally on a dual-core ESP32 microcontroller. Temperature sensing is performed using a high-precision SHT31 sensor that communicates via the I2C protocol [22], [23]. As the thermal actuator, the heating element is controlled using a Zero-Cross AC Dimmer module, which enables proportional power regulation. The system is integrated with an Internet of Things (IoT)–based data pipeline, where telemetry data are transmitted every second via the MQTT protocol to a server [24], [25], stored in a time-series database, and visualized through Grafana.

### Dual-Loop Control Strategy

The control system is designed using a dual-loop architecture to accommodate the difference in time dynamics between the thermal actuator and the machine learning process. The fast loop is executed every second to perform real-time Proportional–Integral–Derivative (PID) calculations in order to regulate the dimmer power percentage based on the current parameter values. Meanwhile, the slow loop is executed every three minutes as the Reinforcement Learning (RL) agent. This thermal interval allows the thermodynamic environment sufficient time to respond to power changes before the agent evaluates the outcome of its actions.

### Discrete Q-Learning Formulation

The intelligent agent in this study is modeled using the Discrete Q-Learning method with components of a Markov Decision Process (MDP), which include state, action, and reward. The state space is formulated through the fuzzification of two main variables, namely temperature error and change in error. The temperature error variable is divided into five levels ranging from very cold to very hot, while the change in error is divided into three levels: decreasing, stable, and increasing. The combination of these variables results in 15 discrete states, which significantly simplifies the computational burden.

Table 2 presents the complete mapping of all 15 discrete states formed from the combination of fuzzification of two input variables. The temperature error level is divided into five levels using thresholds of ±0.5°C and ±2.0°C, while the error rate-of-change (Δerror) is divided into three levels using a threshold of ±0.1°C. Each state is represented as a single integer using the formula: ID = (error_level × 3) + delta_level, allowing the entire representation to be stored in an array of 15 elements that is memory-efficient.

Table 2. State space definition

| **15 Discrete States** | | | | |
|---|---|---|---|---|
| **State ID** | **Error Level** | **ΔError Level** | **Temperature Condition** | **Physical Meaning** |
| **0** | Very Cold ($e > 2°C$) | Decreasing ($\Delta e < -0.1$) | ≪ Target | Temperature far below target and still decreasing |
| **1** | Very Cold ($e > 2°C$) | Stable ($\lvert\Delta e\rvert \leq 0.1$) | ≪ Target | Temperature far below target, rate stable |
| **2** | Very Cold ($e > 2°C$) | Increasing ($\Delta e > 0.1$) | ≪ Target | Temperature far below target but rising |
| **3** | Cold ($0.5 < e \leq 2°C$) | Decreasing | < Target | Temperature slightly below target and decreasing |
| **4** | Cold ($0.5 < e \leq 2°C$) | Stable | < Target | Temperature approaching target, rate stable |
| **5** | Cold ($0.5 < e \leq 2°C$) | Increasing | < Target | Temperature approaching target and rising |
| 6 | On Target ($-0.5 \leq e \leq 0.5$) | Decreasing | ≈ Target | Temperature in target zone but decreasing |
| **7** | **On Target ($-0.5 \leq e \leq 0.5$)** | **Stable** | **≈ Target** | **IDEAL CONDITION: temperature stable at target** |
| 8 | On Target ($-0.5 \leq e \leq 0.5$) | Increasing | ≈ Target | Temperature in target zone but increasing |

| | | | | |
|---|---|---|---|---|
| **9** | Hot (−2 < e < −0.5) | Decreasing | > Target | Temperature above target but decreasing |
| **10** | Hot (−2 < e < −0.5) | Stable | > Target | Temperature above target, rate stable |
| **11** | Hot (−2 < e < −0.5) | Increasing | > Target | Temperature above target and still rising |
| 12 | Very Hot (e < −2°C) | Decreasing | ≫ Target | Large overshoot but decreasing |
| 13 | Very Hot (e < −2°C) | Stable | ≫ Target | Large overshoot, stable rate |
| **14** | **Very Hot (e < −2°C)** | **Increasing** | **≫ Target** | **CRITICAL: overshoot and still rising** |

Note: State ID = (error level × 3) + delta level.

Table 3. Action space definition

| 7 Discrete Actions | | | | | |
|---|---|---|---|---|---|
| **ID** | **Action Description** | **ΔKp** | **ΔKi** | **ΔKd** | **Parameter Bounds After Action** |
| **0** | Increase Kp (Accelerate Response) | **+1.0** | 0 | 0 | 0 ≤ Kp ≤ 60.0; Ki unchanged; Kd unchanged |
| **1** | Decrease Kp (Slow Response) | **−1.0** | 0 | 0 | 0 ≤ Kp ≤ 60.0; Ki unchanged; Kd unchanged |
| **2** | Increase Ki (Reduce SS Error) | 0 | **+0.01** | 0 | Kp unchanged; 0 ≤ Ki ≤ 2.0; Kd unchanged |
| **3** | Decrease Ki (Reduce Windup) | 0 | **−0.01** | 0 | Kp unchanged; 0 ≤ Ki ≤ 2.0; Kd unchanged |
| **4** | Increase Kd (Strengthen Damping) | 0 | 0 | **+0.5** | Kp unchanged; Ki unchanged; 0 ≤ Kd ≤ 40.0 |
| **5** | Decrease Kd (Reduce Noise Sensitivity) | 0 | 0 | **−0.5** | Kp unchanged; Ki unchanged; 0 ≤ Kd ≤ 40.0 |
| **6** | Hold (Maintain Current Parameters) | 0 | 0 | 0 | All parameters unchanged in this cycle |

Note: The upper bound of Kd (40.0) is set independently from the empirical baseline (15.0) and original ZN value (833.3), providing 2.67× exploration space above the empirical value while maintaining system stability. The Hold action (ID=6) is critical for stabilization when the agent determines current parameters are already optimal.

For the action space, the agent has seven discrete actions designed to gradually modify the PID parameters. These actions include increasing or decreasing the proportional, integral, and derivative parameters, as well as one action to maintain the current parameter values. All parameters are constrained by upper and lower bounds to prevent extreme system instability.

The reward function is specifically designed to minimize error, prevent overshoot, and suppress oscillations in the heating actuator. The mathematical formulation of the reward function is presented in Equation 1.

$$R = -|e(t)| - (\lambda_1 \times P_{overshoot}) - (\lambda_2 \times |\Delta e(t)|) \quad (1)$$

Description:

$R$ = Total reward received by the agent
$e(t)$ = Current temperature error relative to the target
$\lambda_1$ = Penalty weight when overshoot occurs (temperature exceeds the target)
$P_{\_overshoot}$ = Active penalty value when the actual temperature exceeds the upper limit
$\lambda_2$ = Penalty weight for oscillation or change in error
$\Delta e(t)$ = Difference between the current error and the error in the previous cycle

The determination of penalty constant values in the reward function is based on critical biological and actuator system characteristics. The value $\lambda_1 = 10.0$ for the overshoot penalty is set asymmetrically much larger than the base error weight. This reflects the asymmetric biological consequences: in the context of egg incubation, a temperature increase of even 1°C above the setpoint of 38.0°C during critical periods can cause irreversible thermal stress on the embryo [1]–[4]. With a penalty weight 10 times the base error, the agent is trained to prioritize overshoot prevention above all else, even at the cost of slower rise time. This behavior manifests in the consistent early braking strategy observed in the experimental data of Scenarios 1 and 3.

The value $\lambda_2 = 5.0$ for the oscillation penalty is set at half the overshoot penalty value based on actuator lifespan considerations. The AC heating actuator controlled through the Zero-Cross Dimmer module is inductive; excessively frequent and drastic power changes (high oscillation) can cause current spikes that accelerate component degradation. The penalty proportional to $|\Delta e(t)|$ encourages the agent to choose a smooth and gradual temperature approach trajectory. Both constant values were determined through empirical iteration during preliminary testing sessions, and validated by observing agent behavior in comparative scenarios.

The agent's learning process is carried out by updating the values in the experience matrix (Q-Table) during each slow-loop cycle. This value update is calculated based on the Bellman Equation [26], [27], as presented in Equation 2.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max Q(s',a') - Q(s,a)] \quad (2)$$

Description:
$Q(s,a)$ = Q-value for the current state (s) and action (a).
$\alpha$ = Learning rate that determines the update of new information.
$R$ = Actual reward obtained from the environment.
$\gamma$ = Discount factor used to weigh the value of future rewards.
$Q(s',a')$ = The highest estimated Q-value in the next state (s′).

The exploration–exploitation policy is regulated using the Epsilon-Greedy method to ensure that the agent does not rely solely on past experiences but also explores new parameter combinations that may yield more optimal results. The Q-Table matrix is periodically stored in the microcontroller's non-volatile memory using a wear-leveling mechanism to prevent the agent from losing its learned knowledge when the system is restarted or experiences power loss.

All Q-Learning agent hyperparameters used in this study are summarized in Table 4 along with the basis and justification for each value. These values are hardcoded directly into the ESP32 firmware and verified through a series of preliminary experiments before the main testing scenarios were executed.

Table 4. Q-learning agent hyperparameters

| Hyperparameter | Symbol / Value | Source | Justification |
|---|---|---|---|
| **Learning Rate** | **α = 0.1** | Empirical / Literature | Conservative value commonly recommended for real-time physical systems. Prevents aggressive Q-Table updates from transient sensor disturbances while enabling sufficiently fast learning within the thermal time scale of minutes. |
| **Discount Factor** | **γ = 0.9** | Empirical / Literature | Value approaching 1 indicates the agent prioritizes long-term rewards (sustained temperature stability) over short-term gains. Appropriate for slow thermal systems requiring control consistency. |
| **Exploration Rate** | **ε = 0.2** | Empirical | 20% exploration vs. 80% exploitation ratio balances discovery of new policies and utilization of proven ones. Keeps the agent adaptive to environmental changes without causing destabilization. |
| **Number of States** | **15 discrete states** | Design (5×3) | Combination of 5 temperature error levels × 3 error change levels. Intentionally limited so Q-Table size (15×7 = 105 float values) fits comfortably within ESP32 microcontroller RAM. |
| **Number of Actions** | **7 discrete actions** | Design | Three action pairs (±Kp, ±Ki, ±Kd) plus one Hold action. Provides sufficiently fine-grained control without exploding the search space. |
| **RL Cycle Interval** | **Every 3 minutes** | Empirical (thermal) | Sufficient time for the thermal system (time constant T=747.7 s ≈ 12 min) to respond to parameter changes before the agent evaluates action outcomes. |
| **NVS Save Interval** | **Every 10 updates (~30 min)** | Embedded design | Wear-leveling mechanism to preserve ESP32 flash memory lifespan. Q-Table saved to Non-Volatile Storage (NVS) periodically, not every update, to minimize flash write cycles. |

The convergence behavior of the Q-Table in this system has unique characteristics because the learning process takes place online in real thermal time scale. Each RL cycle lasts 3 minutes, so in a 12-minute test session, the agent only executes 4 Q-Table update actions. Although the number of cycles per session is limited, the memory persistence mechanism through the ESP32's built-in Non-Volatile Storage (NVS) enables experience accumulation across sessions without data loss due to restart or power outage. The Q-Table is saved to NVS every 10 updates (~30 minutes of operation) using a wear-leveling mechanism to preserve the flash memory lifespan.

Practical convergence indication in this system is observed from the consistency of actions chosen by the agent for the same state in consecutive cycles. In Scenario 3 (32L incubator), the agent consistently chose the Increase Kp action in the first three cycles (state: Very Cold/Increasing and Cold/Increasing) and switched to Decrease Kp in the fourth cycle when the state shifted to Cold/Increasing approaching target. This policy consistency on a Q-Table starting from zero demonstrates that the algorithm can build adequate value representations even in very short learning sessions, which is a fundamental advantage of the Discrete Q-Learning approach with a compact state space (15 states) compared to deep learning methods requiring thousands of training episodes.

## 3. RESULTS AND DISCUSSIONS

### Curve Extraction and Baseline Parameter Determination

Before conducting the performance comparison, the baseline parameters for the static PID controller were determined using the Ziegler–Nichols open-loop reaction curve method. The system was given a test input in the form of a constant heating power of 50% to extract the thermodynamic characteristics of the 64-liter incubator.

The selection of 50% power as the step test operating point is based on three considerations. First, 50% is the midpoint of the actuator's working range (0–100%) that represents the nominal operating condition of the system, so the extracted thermodynamic characteristics (L, T, K) reflect system behavior under actual working conditions. Second, power below 30% risks producing a response that is too slow and susceptible to environmental disturbances during testing, while power above 70% risks reaching thermal saturation and exceeding the safety cut-off limit at 40.0°C. Third, selecting the same operating point (50%) for establishing PID baseline parameters and for the Q-Learning agent's initial condition ensures comparison consistency: the agent begins learning from a reference point identical to the condition where ZN parameters were optimized, so the Q-Table convergence process can start from a well-characterized basis.

Changes in the operating point would theoretically shift the L, T, and K values because the system's thermal characteristics are nonlinear, particularly in the low temperature range where heater efficiency is higher. However, in the context of this study, such variation does not significantly affect Q-Table convergence because the agent does not depend on a mathematical model of the system — the agent learns directly from real thermal responses through an actual temperature error-based reward mechanism (model-free learning).

Based on the transient response test results, a dead time (L) of 181.1 seconds and a time constant (T) of 747.7 seconds were obtained, with a process gain (K) value of 0.538.

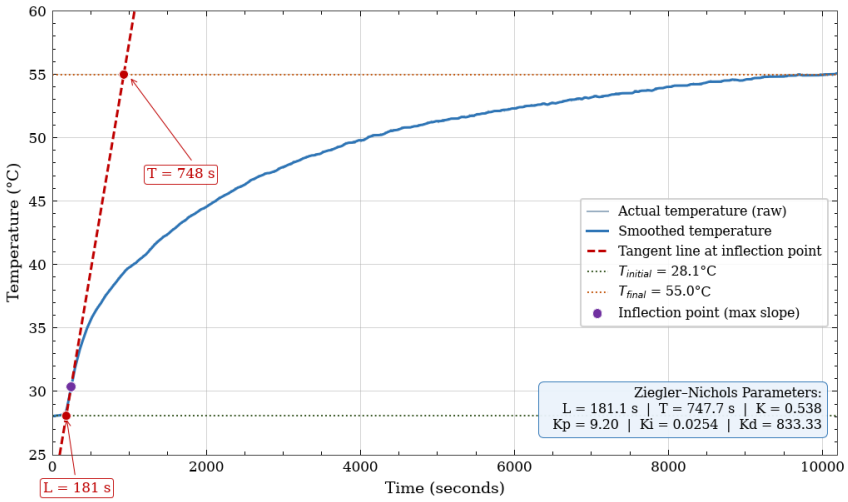


Figure 1. Open-loop reaction curve of the 64-liter incubator at constant 50% heating power: measurement of dead time (L = 181.1 s), time constant (T = 747.7 s), and process gain (K = 0.538) as the basis for Ziegler-Nichols tuning.

Using the Ziegler–Nichols formulation for a full PID controller, the initial mathematical calculation produced proportional parameter Kp=9.20, integral parameter Ki=0.025, and derivative parameter Kd=833.3. However, in real thermal system implementations, excessively large derivative values are highly sensitive to sensor reading noise (derivative kick) [28], [29], which may lead to actuator saturation. Therefore, an empirical adjustment was performed by reducing the Kd value to 15.0. The resulting parameter set (Kp=9.20;Ki=0.025;Kd=15.0) was then fixed as the baseline static PID configuration for all subsequent comparative testing scenarios.

It is important to distinguish the three Kd values that appear in this study. The original Ziegler-Nichols Kd value (833.3) is the result of pure mathematical calculation that is not directly implemented because it would cause derivative kick in the thermal system with sensor noise. The empirical Kd value (15.0) is the value carefully selected for use as the static PID baseline parameter — the smallest value that still provides adequate oscillation damping without actuator saturation. The upper bound of the Q-Learning agent's exploration for Kd is set at 40.0, a value independently determined based on stability testing, providing an exploration space 2.67 times above the empirical value but still far below the range that could cause instability. This bound allows the agent to find a more optimal Kd value than the empirical baseline if the environmental thermodynamic conditions require it.

**Transient Response Test under Standard Conditions**

The transient response (step response) test was conducted to compare the performance of the conventional Ziegler–Nichols-based PID controller with the adaptive Q-Learning-based PID controller. The system was started from room temperature to reach the predetermined target value of 38.0°C within a standard incubator volume of 64 liters.

In the test using the conventional PID controller, the system demonstrated a relatively responsive initial target-reaching time of 10.67 minutes. However, the conventional controller produced a temperature overshoot that exceeded the target limit, reaching a peak temperature of 39.1°C. In the context of egg incubation, an increase of 1.1°C above the ideal temperature poses a potentially fatal risk to embryo survival. Furthermore, the conventional system required a considerably long time, approximately 30.25 minutes, to dampen oscillations and reach a stable condition within the ±0.5°C tolerance band (see Figure 2).

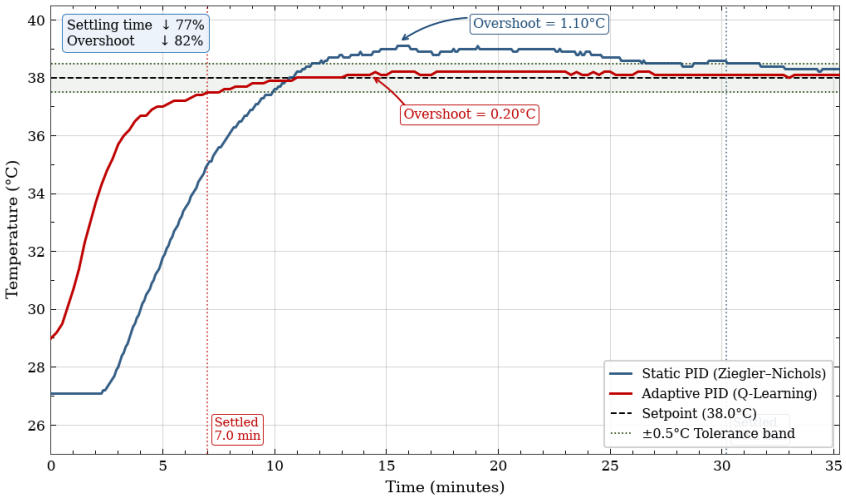


Figure 2. Comparison of transient responses between the conventional PID controller and the q-learning adaptive PID controller.

In contrast, the implementation of the Q-Learning agent demonstrates a significant improvement in performance in suppressing temperature overshoot. This adaptive controller reaches the target temperature within 11.00 minutes, slightly slower than the conventional method because the artificial intelligence agent performs early braking when it detects that the temperature is approaching the target. This decision results from the reward function, which assigns a high penalty for temperature overshoot. As a result, the overshoot is drastically reduced by up to 81.8%, with the peak temperature reaching only 38.2°C. The adaptive system also exhibits superior stability, as it is able to settle within the stable zone in only 7.00 minutes, or 76.9% faster than the conventional method. Cumulative accuracy also improved significantly with a 52.5% reduction in Integral Absolute Error (IAE) and a 66.7% reduction in steady-state ripple. A complete quantitative comparison of all performance metrics is presented in Table 5.

Table 5. Performance metrics comparison

| Scenario 1: Transient Response (64-Liter Incubator) |
|---|

| Performance Metric | Static PID (Ziegler-Nichols) | Adaptive PID (Q-Learning) | Improvement |
|---|---|---|---|
| Peak Temperature (°C) | 39.1 | 38.2 | — |
| Overshoot (°C) | 1.10 | **0.20** | **↓ 81.8%** |
| Rise Time (minutes) | 10.67 | 11.00 | — |
| Settling Time (minutes) | 30.25 | **7.00** | **↓ 76.9%** |
| Steady-State Mean (°C) | 38.022 | 37.968 | — |
| Steady-State Std Dev (°C) | 0.069 | **0.047** | **↓ 32.3%** |
| Steady-State Ripple Peak-to-Peak (°C) | 0.30 | **0.10** | **↓ 66.7%** |
| IAE — Integral Absolute Error (°C·min) | 66.01 | **31.38** | **↓ 52.5%** |

Note: Setpoint 38.0°C, 64-liter incubator. Settling Time measured until temperature enters and remains within ±0.5°C band for minimum 60 seconds. IAE expressed in °C·min. Steady-State computed from last 200 data points.

**Test of Rejection of Dynamic Disturbances**

The robustness of the control algorithm was further evaluated through a disturbance rejection test by fully opening the incubator door for 30 seconds while the system was already in a stable condition. This test aimed to assess how quickly and accurately the system could restore the temperature to the setpoint without triggering further fluctuations. When the door was opened, the incubator's ambient temperature dropped sharply, reaching 36.0°C in the conventional system and 35.5°C in the adaptive system due to differences in the physical room temperature during the test (see Figure 3).

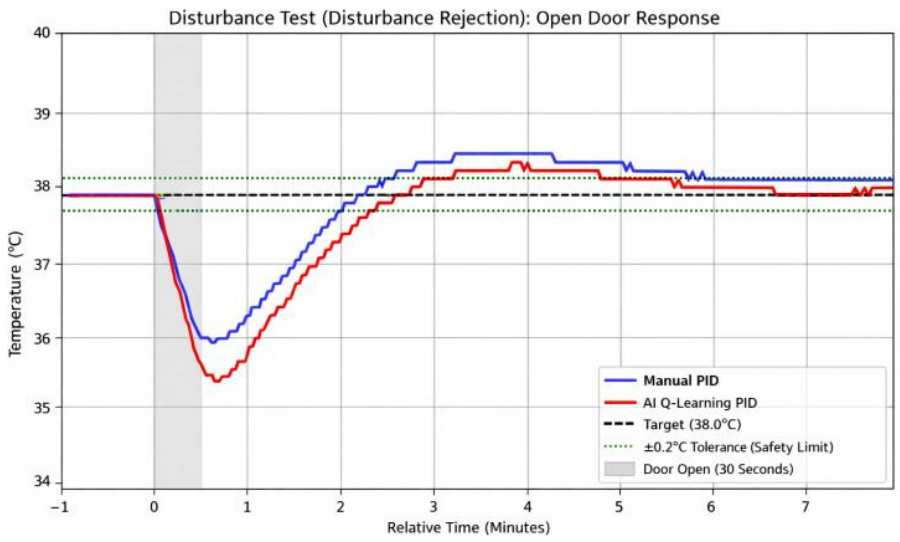


Figure 3. System response to dynamic disturbances
in the form of opening the incubator door for 30 seconds.

During the recovery process, the Q-Learning agent demonstrated its resilience by detecting the large error and automatically increasing the proportional parameter values for a rapid response. Measured from each system's minimum temperature point, the adaptive system required 1.47 minutes to return to the stable zone, while the conventional system required 1.17 minutes. Although absolutely slower, this comparison must be interpreted in context: the RL system faced a larger disturbance (2.5°C drop vs. 2.0°C for conventional PID). Both systems demonstrated comparable recovery capability relative to their respective disturbance scales, with identical post-recovery steady-state ripple at ±0.1°C. More significantly, the adaptive algorithm proved to suppress secondary post-recovery oscillations more effectively, confirming the reliability of the Self-Tuning method in handling real-world environmental anomalies.

**Evaluation of Cross-Environment Adaptation and Parameter Dynamics**

The autonomous adaptation capability of the proposed algorithm was evaluated through a cross-environment test. The hardware was moved to a 32-liter incubator chamber without recalibrating the Ziegler-Nichols parameters or reinitializing the Q-Table matrix – the agent started with an empty Q-Table and learned online without historical data from previous sessions. The 50 percent reduction in air thermal mass caused the temperature to rise much more aggressively. In the comparison test, the conventional PID controller, whose operational parameters were designed for the larger chamber, failed to adapt—reaching the target prematurely in 8 minutes but failing to decelerate, resulting in an overshoot back to 38.3 degrees Celsius (see Figure 4).

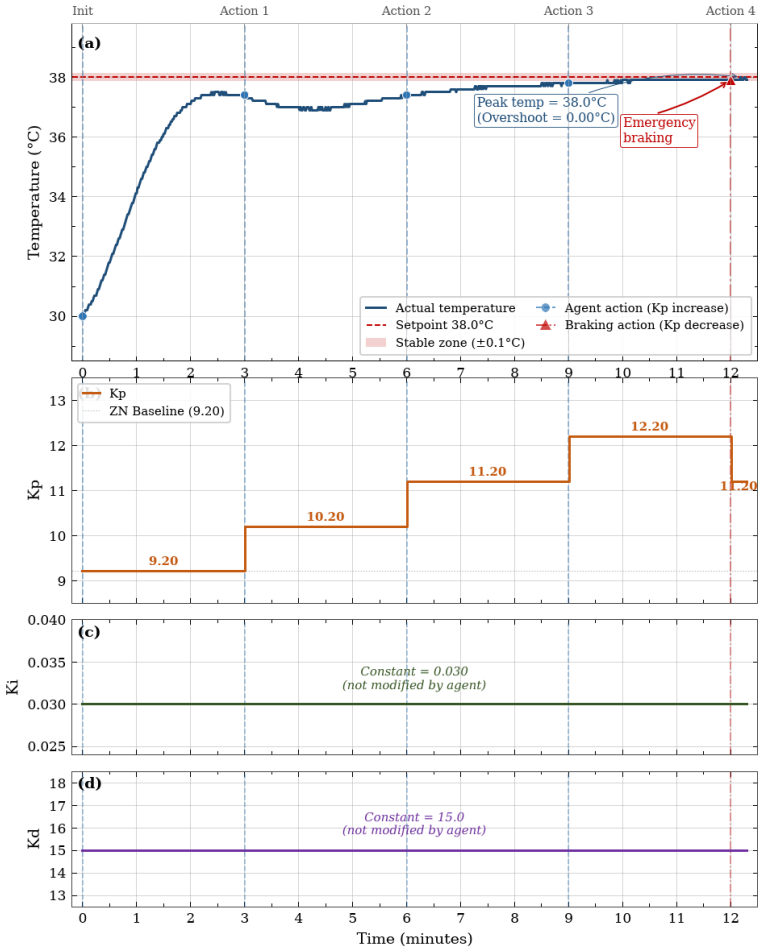


Figure 4. Dynamics of PID parameters by the q-learning agent on the 32-liter incubator: (a) actual temperature and 38.0°C setpoint with ±0.1°C stability zone; (b) proportional parameter Kp; (c) integral parameter Ki; (d) derivative parameter Kd. Vertical dashed lines indicate agent action execution moments every 3 minutes (Actions 1–3: Kp increase; Action 4: emergency braking — Kp decrease). Ki and Kd were maintained constant throughout the session by the agent's optimal policy, reflecting that Kp modification provides the highest control efficiency during the transient phase while minimizing risks of integral windup and derivative kick.

On the other hand, the Q-Learning agent demonstrated highly precise adaptive intelligence. Visualization of the parameter dynamics in Figure 4 and Table 6 shows that in four RL cycles (every 3 minutes), the agent gradually increased the Kp value from 9.20 to 12.20 during the approach phase to accelerate thermal response. Then, precisely when the temperature touched 37.9°C at minute 12, the agent executed a Kp reduction to 11.20 as an emergency braking maneuver to prevent overshoot. As a result, the temperature landed exactly at 38.0°C without exceeding the setpoint at all (overshoot = 0.00°C). Ki and Kd parameters were maintained constant by the agent throughout this session. This behavior reflects the optimal policy learned by the agent; during the transient phase with a large initial error, Kp modification provides the highest control efficiency, while Ki modification risks causing integral windup and Kd modification risks amplifying sensor noise (derivative kick). A complete quantitative metrics comparison is presented in Table 6, while details of each agent action are listed in Table 7.

Table 6. Performance metrics comparison

| **Scenario 3: Cross-Environment Evaluation (32-Liter Incubator, No Recalibration)** |
|---|

| Performance Metric | Static PID (64L Parameters) | Adaptive PID Q-Learning (Fresh Q-Table) | Note |
|---|---|---|---|
| Peak Temperature (°C) | 38.3 | 38.0 | Zero overshoot |
| Overshoot (°C) | 0.30 | **0.00** | **Fully eliminated** |
| Rise Time (minutes) | 8.00 | 12.18 | RL: early braking |
| Settling Time (minutes) | 5.93 | 6.35 | Comparable (<7%) |
| Steady-State Ripple (°C) | 0.10 | 0.10 | Equivalent |
| IAE (°C·min) | 29.09 | **13.05** | **↓ 55.1%** |
| Recalibration Required | Yes (64L param, not recalibrated) | **No (Online learning from scratch)** | **Key advantage** |

Note: Static PID uses parameters Kp=9.20; Ki=0.025; Kd=15.0 (optimized for 64L). Q-Learning agent starts with an empty Q-Table without historical data. Longer RL rise time is a consequence of the early braking strategy to prevent overshoot.

Table 7. PID parameter dynamics by q-learning agent

| 32-Liter Incubator Session | | | | | | |
|---|---|---|---|---|---|---|
| **Minute** | **Agent Action** | **Kp** | **Ki** | **Kd** | **Temp (°C)** | **Interpretation** |
| 0 | Initialization | 9.20 | 0.030 | 15.0 | 30.1 | Initial condition — Ziegler-Nichols baseline parameters |
| 3 | Increase Kp (+1.0) | **10.20** | 0.030 | 15.0 | 37.3 | Accelerate temperature approach to target |
| 6 | Increase Kp (+1.0) | 11.20 | 0.030 | 15.0 | 37.4 | Still in initial approach phase |
| 9 | Increase Kp (+1.0) | **12.20** | 0.030 | 15.0 | 37.8 | Temperature near target, Kp at maximum |
| **12** | **Decrease Kp (−1.0) [EMERGENCY BRAKING]** | 11.20 | 0.030 | 15.0 | 37.9 | **BRAKING: Agent reduces Kp → result: peak temp = 38.0°C (overshoot = 0.00°C)** |

Note: Ki and Kd were maintained constant throughout the session by the agent's decision. During the transient phase with large initial error, Kp modification provides the highest control efficiency. Ki modification risks integral windup; Kd modification risks sensor noise amplification (derivative kick).

**Discussion, Scientific Implications, and Approach Limitations**

The results of this study place the Lightweight Discrete Q-Learning approach in a unique position among the machine learning-based adaptive control literature. Unlike RL implementations on thermal systems that generally rely on pre-trained simulations or cloud infrastructure [17], this study demonstrates that a tabular agent with a compact state space (15 states) can build effective control policies in direct learning sessions on low-power physical hardware. Alejandro-Sanjines et al. [30] reported similar advantages of RL approaches for model-free adaptive control, however their implementation uses a DDPG architecture with 300 neurons which is not feasible for running on microcontrollers such as ESP32. This study bridges that gap by sacrificing representation capacity (tabular vs. neural) but gaining advantages in computational efficiency, policy interpretability, and ease of deployment on constrained hardware.

From a practical perspective, the proposed control architecture has direct implications for the small to medium-scale poultry farming industry in developing countries [13], [20], [31], where budget and infrastructure limitations are the main barriers to adopting intelligent control technology. The system's ability to adapt to incubator volume changes (32L vs 64L) without manual recalibration — proven in Scenario 3 — reduces dependence on technical experts and enables non-technical operators to operate the system autonomously. Further cost savings are obtained from using ESP32 as the sole processing unit for PID execution, RL agent, IoT telemetry, and Q-Table storage simultaneously — a functional consolidation that cannot be achieved by Neural Network or DRL-based approaches.

Nevertheless, this study has several limitations that need to be acknowledged transparently. First, the granularity of environmental representation limited to 15 discrete states — although computationally efficient — limits the agent's ability to distinguish system conditions that have similar dynamics but require different responses. In more complex thermal systems or with varying setpoints, this number of states is likely insufficient and would need to be expanded, which directly increases the Q-Table size and memory requirements. An expansion to, for example, 25 states (5 error levels × 5 error change levels) would produce a Q-Table of 175 values — still within ESP32's capacity, but requiring more RL cycles to achieve convergence.

Second, the implemented control system is single-variable — only regulating temperature without considering relative humidity, which is also a critical variable in the egg incubation process [1], [12], [13]. Optimal egg incubators require humidity control in the range of 55–65% RH during the incubation period and 70–80% RH during the last 3 days before hatching. Integration of humidity control into the existing Q-Learning framework would require an expanded action space and potentially require more complex multi-agent or MIMO control architectures.

Third, each test scenario in this study was conducted in a single experimental session without repetition. Result variability due to different environmental conditions (room temperature, ambient humidity)

was not statistically quantified. Future research involving controlled-condition repeated experiments will increase the statistical confidence in the reported performance claims, in line with the recommendations of Cusihuallpa-Huamanttupa et al. [21] for validation of RL-based control systems in real hardware implementations.

Despite these limitations, the implemented dual-loop architecture provides a foundation that can be developed incrementally. Adding humidity control, expanding the state space, or even integrating with model-based methods that utilize the already-identified thermodynamic parameters (L, T, K) as a prior to accelerate Q-Table convergence are promising development directions. This study provides empirical evidence that the Discrete Q-Learning approach on constrained hardware is a feasible and scalable starting point for developing adaptive control systems in IoT-based agricultural environments.

## 4. CONCLUSION

There are at least three aspects we consider to be the contributions of this research. First and foremost, we successfully ran a Q-Learning agent directly on ESP32 — not in a simulator, not in the cloud — and this system learned from scratch in a real physical environment without any historical data whatsoever. This is fundamentally different from PSO, Neural Networks, or DRL which all require a training process before they can be used. Second, we tested this system not only in one condition, but in three complementary scenarios deliberately designed to validate different aspects of system robustness: from initial response, robustness when the door was opened, to transfer to an incubator of different size. The results were consistent across all three. Third, a Q-Table of 420 bytes proved sufficient to control a nonlinear thermal system with significant dead-time. A size this small can be stored in any low-cost flash memory available on resource-constrained microcontrollers.

Data from the three test scenarios show consistent results. In the 64-liter incubator, overshoot that was originally 1.10°C dropped to 0.20°C — and the system reached stable condition 76.9% faster. More striking is the result in the 32-liter incubator: the agent encountering this volume for the first time managed to hold the temperature exactly at 38.0°C without exceeding it at all. Zero overshoot. For a system working with chicken embryos, this is not merely a statistical figure. The door disturbance test was also handled well — although the temperature drop in the RL test was deeper due to different room conditions, recovery still took place in comparable time. All of this was achieved on a low-cost microcontroller operating autonomously, without cloud connectivity or external computing infrastructure.

This research has several limitations that need to be considered. The granularity of environmental representation limited to 15 discrete states limits the agent's scalability in systems with more complex dynamics. The evaluation was conducted on a single-variable (temperature only) control system without involving humidity regulation that is also critical in the incubation process. Furthermore, each test scenario was executed in a single experimental session without statistical repetition, so result variability due to fluctuating environmental conditions has not been quantified.

Several future research directions can be identified based on the findings and existing limitations. One of the most natural steps is expansion to multi-variable control architecture, namely simultaneous temperature and humidity control using a multi-agent Q-Learning framework. This approach is considered capable of significantly increasing the system's relevance for end users. On the other hand, exploration of convergence acceleration techniques is also worthy of prioritization. Q-Table initialization based on domain knowledge or its integration with a thermodynamic model as model-based prior has the potential to reduce the number of RL cycles needed before the optimal policy is achieved. Finally, the generalizability of findings needs to be strengthened through more systematic testing. Controlled statistical repetitions and a wider variety of hardware — including other microcontrollers with different memory capacities — will provide a stronger empirical foundation for the conclusions of this research.

## CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

**Author1:** Conceptualization, Methodology, Software, Project Administration, Writing – Original Draft Preparation. **Author2**: Software, Validation, writing – Review and Editing, Supervision.

## DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## REFERENCES


[1] M. Islamiyah and S. Arifin, "Control Sistem Inkubator Telur Ayam Menggunakan Metode Fuzzy Logic," *Positif J. Sist. dan Teknol. Inf.*, Dec. 2023.

[2] S. P. Ashary, M. Irwan, and Armayani M, "Pengaruh Suhu Penetas Yang Berbeda Terhadap Fertilitas, Bobot Tetas, dan DeadIn Shell Telur Itik," *Innov. J. Soc. Sci. Res.*, 2025.

[3] L. O. D. Saputra, R. Badruddin, and T. Saili, "Pengaruh Suhu Penetasan yang Berbeda Terhadap Fertilitas , Daya Hidup Embrio , Daya Tetas dan Bobot Tetas Telur Ayam Arab ( G allus Turcicus )," *JIPHO (Jurnal Ilm. Perternakan Halu Oleo)*, vol. 8, no. 1, pp. 13–18, 2026.

[4] M. M. Adame, Y. Yusuf, and N. T. Kuda, "Influences of Types of Incubators on Hatchability of Eggs," *Adv. Appl. Sci.*, vol. 8, no. 3, pp. 80–85, 2023.

[5] F. A. Septian, A. M. T. Almy, M. N. Nashir, and T. Andrasto, "Simulasi Pengendalian Suhu pada Inkubator Penetasan Telur Ayam Menggunakan Arduino Berbasis PID," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 13, no. 3, p. 238, 2025.

[6] N. Hu, "The Limitations of Traditional PID Controllers and Modern Optimization Methods," in *Proceedings of the 3rd International Conference on Mechatronics and Smart Systems*, 2025, vol. 147, no. 1, pp. 238–244.

[7] G. Bujgoi and D. Sendrescu, "Tuning of PID Controllers Using Reinforcement Learning for Nonlinear System Control," *Processes*, vol. 13, no. 3, p. 735, 2025.

[8] D. Irawan, S. A. A. Syah, and D. O. Cahyani, "Tuning PID Controller Berbasis Algoritma Kecerdasan Buatan," *Multitek Indones. J. Ilm.*, vol. 18, no. 1, pp. 72–82, 2024.

[9] B. H. Sanjaya, A. Pujiyanta, and R. D. Puriyanto, "Fuzzy Logic and Neural Network-Based Self-Tuning PID for Vacuum Pressure Stabilization," *J. Appl. Informatics Comput.*, vol. 9, no. 5, pp. 2247–2256, 2025.

[10] Y. S. Lee and D. W. Jang, "Optimization of neural network-based self-tuning pid controllers for second order mechanical systems," *Appl. Sci.*, vol. 11, no. 17, pp. 1–12, 2021.

[11] J. C. Almachi, R. Vicente, E. Bone, J. Montenegro, E. Cando, and S. Reina, "Implementation of a Neural Network for Adaptive PID Tuning in a High-Temperature Thermal System," *Energies*, vol. 18, no. 12, p. 3113, 2025.

[12] Y. Z. Maulana, F. Fathurrohman, and G. Wibisono, "Egg Incubator Temperature and Humidity Control Using Fuzzy Logic Controller," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 7, no. 2, pp. 318–325, 2023.

[13] R. Wahyuni, Y. Irawan, A. Febriani, Nurhadi, H. T. Saputra, and R. Andrianto, "Smart Egg Incubator Based on IoT and AI Technology for Modern Poultry Farming," *Ilk. J. Ilm.*, vol. 16, no. 2, pp. 134–144, 2024.

[14] F. P. Budiana, "Model Atensi Citra Wajah Audiens Berbasis Edge Machine Learning," Dec. 2024.

[15] M. Merenda, C. Porcaro, and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors (Switzerland)*, vol. 20, no. 9, p. 2533, 2020.

[16] Y. Yao, N. Ma, C. Wang, Z. Wu, C. Xu, and J. Zhang, "Research and Implementation of Variable-Domain Fuzzy PID Intelligent Control Method Based on Q-Learning for Self-Driving in Complex Scenarios," *Math. Biosci. Eng.*, vol. 20, no. 3, pp. 6016–6029, 2023.

[17] O. Dogru *et al.*, "Reinforcement Learning Approach to Autonomous PID Tuning," *Comput. Chem. Eng.*, 2022.

[18] I. Kamenko, V. Čongradac, and F. Kulić, "A Novel Fuzzy Logic Scheme for PID Controller Auto-Tuning," *Automatika*, vol. 63, no. 2, pp. 365–377, 2022.

[19] J. Huang, F. Huang, J. Liang, J. Chen, and P. Li, "Fuzzy Adaptive PID Algorithm in Temperature Control System of Lead Patenting Furnace," *Adv. Civ. Eng.*, 2024.

[20] W. Zhang *et al.*, "Integrated Irrigation of Water and Fertilizer with Superior Self-Correcting Fuzzy PID Control System," *PLoS One*, vol. 20, no. 5 May, pp. 1–36, 2025.

[21] K. Cusihuallpa-Huamanttupa *et al.*, "Deep Reinforcement Learning-Based Intelligent Water Level Control: From Simulation to Embedded Implementation," *Sensors*, vol. 26, no. 1, p. 245, 2025.

[22] A. Sensirion, "Datasheet SHT3x-DIS Humidity and Temperature Sensor," 2022.

[23] A. Mauhib and R. Hermawan, "Komparasi Keandalan Sensor DHT22 dan SHT31 pada Greenhouse Berbasis IoT," *Competitive*, vol. 20, pp. 49–60, 2025.

[24] R. F. Pratama, R. S. R. Wicaksono, and A. N. Pramudhita, "Perancangan dan Implementasi Protokol MQTT pada Sistem Parkir Cerdas Berbasis IoT," *J. Inform. dan Tek. Elektro Terap.*, vol. 11, no. 3, Aug. 2023.

[25] I. K. A. A. Aryanto, D. Maneetham, and E. Triandini, "Developing a Smart System for Infant Incubators Using the Internet of Things and Artificial Intelligence," *Int. J. Electr. Comput. Eng.*, vol. 14, no. 2, pp. 2293–2312, 2024.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Second Edition)*. 2018.

[27] C. Szepesvári, *Algorithms for Reinforcement Learning*, vol. 4. Morgan & Claypool Publishers (Synthesis Lectures on AI and Machine Learning), 2009.

[28] K. Ogata, "Modern Control Engineering (5th Edition)," *Book*. Prentice Hall, 2010.

[29] P. Bistak, M. Huba, D. Vrancic, and S. Chamraz, "IPDT Model-Based Ziegler–Nichols Tuning Generalized to Controllers with Higher-Order Derivatives," *Sensors*, vol. 23, no. 8, p. 3787, 2023.

[30] U. Alejandro-Sanjines, A. Maisincho-Jivaja, V. Asanza, L. L. Lorente-Leyva, and D. H. Peluffo-Ordóñez, "Adaptive PI Controller Based on a Reinforcement Learning Algorithm for Speed Control of a DC Motor," *Biomimetics*, vol. 8, no. 5, p. 434, 2023.

[31] F. Peprah, S. Gyamfi, M. Amo-Boateng, E. Buadi, and M. Obeng, "Design and construction of smart solar powered egg incubator based on GSM/IoT," *Sci. African*, vol. 17, p. e01326, 2022.