ASEAN Engineering Journal

Full Paper

APPLICATION OF THE STRUCTURED CONTROL LANGUAGE IN BUILDING THE NEURAL NETWORK BASED PID INTEGRATED IN THE PROGRAMMABLE LOGIC CONTROLLER

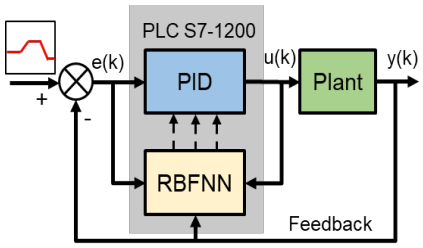
Van-Khanh Nguyen[a], Vy-Khang Tran[a], Van-Den Ngo[b], Hoang-Dung Nguyen[a], Chi-Ngon Nguyen[b*]

[a]Faculty of Automation Engineering, Can Tho University; Campus II, 3/2 street, Ninh Kieu ward, Can Tho City 94000, Vietnam
[b]Nam Can Tho University; 168 Nguyen Van Cu Ext. street, An Binh ward, Can Tho City 94000, Vietnam

Article history
Received
08 December 2024
Received in revised form
07 July 2025
Accepted
24 June 2026
Published online
31 August 2026

*Corresponding author
ncngon@nctu.edu.vn

Graphical abstract



Abstract

Programmable Logic Controllers (PLC) have been developed and deployed mainly in industry because of their reliability and stability in a harsh environment. The PLCs have been integrated with stronger processors, input/output modules, and modern communication protocol. Therefore, they are not only ideal for working in industrial zones but also in testing automotive algorithms. This paper aims to apply the structured control language (SCL) to directly implement the radial basis function neural network (RBFNN) into a Siemens PLC, two RBFNN-based adaptive proportional–integral–derivative (PID) controllers were implemented to control the working temperature of a metal–oxide–semiconductor field-effect (MOSFET) transistor named IRFZ44N via changing the gate voltage. The experimental results show that, the RBFNN is working correctly in PLC and could be adapted by the PID controller to improve the temperature response in comparing with the PID controller integrated in the PLC (model: S7-1200, CPU 1215C DC/DC/DC) 48.1% and 30% in terms of the overshoot and steady state error, respectively. Moreover, the control rules of the proposed structure have eliminated the output chattering which reduced the negative effect on the actuators. These results also proved the ability of deploying some modern control algorithms on the PLCs and applying the PLC in testing the control theory.

Keywords: Structured control language, Radial basis function, PID controller, Programmable logic controller, Neural network

1.0 INTRODUCTION

Programmable Logic Controllers (PLCs) are designed to operate reliably in harsh industrial environments, such as those involving high temperatures, electrical noise, and exposure to chemicals. With advancements in technology, modern PLCs have been equipped with powerful processors, enhanced signal-processing capabilities, and efficient communication interfaces. Additionally, PLCs feature a wide range of integrated digital and analog input/output (I/O) modules, high-speed pulse inputs, and pulse train outputs. They are also compatible with various industrial communication protocols, making them highly versatile tools for solving industrial automation problems as well as for use in research and development projects [1–5]. Given their robust hardware and flexible architecture, PLCs are not only well-suited for traditional industrial applications but also hold significant potential for implementing and testing both classical and modern control algorithms.

Several researchers have proposed various approaches to directly implement automotive control algorithms on PLC platforms [6–10]. A commonly used method involves simulating the controller on a computer and establishing a connection with the PLC via an OPC (Object Linking and Embedding for Process Control) server. In this configuration, the PLC acts as an interface

that reads sensor feedback and sends control signals to the plant through actuators [6]. Another approach is to deploy the controller directly onto the PLC using code automatically generated by MATLAB/Simulink, via tools such as the PLC Coder [7]. In addition, modern control algorithms or advanced features have also been flexibly integrated into PLCs through manual programming using Structured Control Language (SCL) [8–10]. While the use of automatic code generation (e.g., PLC Coder) is efficient, it is limited to supported algorithms and compatible models. On the other hand, directly implementing controllers using ladder logic (LAD) or SCL requires significant programming expertise. This remains a challenging task due to the limited resources and support available from the developer community.

The indirect integration of Artificial Neural Networks (ANNs) or Deep Learning Networks (DLNs) into PLC-based industrial applications has also garnered increasing attention from researchers. For example, Rybczak et al. utilized a Neural Processing Unit (NPU) module to implement an ANN for color-based object detection using an IRB1200 robotic arm, which was controlled by the high-performance Siemens S7-1512C PLC [11]. Similarly, the well-known YOLOv3 deep neural network was deployed on a personal computer (PC) to perform online defect detection in a simulated pill manufacturing system, where the process was controlled by a Siemens S7-1200 PLC [12]. In other works, ANNs have also been employed for anomaly detection in industrial systems [13–15]. Despite these advances, directly deploying ANNs or DLNs on PLCs remains a significant challenge due to the limited processing power and memory capacity of PLCs compared to PC-based systems. However, integrating simpler neural network architectures—such as the Radial Basis Function Neural Network (RBFNN)—directly into PLCs may offer a feasible and promising step toward the development of intelligent controllers for industrial automation..

This paper aims to utilize Structured Control Language (SCL) to directly implement and evaluate a Radial Basis Function Neural Network (RBFNN) algorithm on a Siemens S7-1200 PLC. The mathematical formulation of the RBFNN is first analyzed and translated into a Function Block (FB) written in SCL for use within the PLC program. This custom RBFNN FB is then employed to develop an adaptive control scheme that enhances the performance of the built-in PID controller—referred to as the PID Compact FB—in the S7-1200 PLC. Two adaptive control structures are proposed and integrated with the PID Compact FB, and their performance is compared against the standalone PID Compact to assess control quality. In the first structure, named RBFNN-PID1, the RBFNN is used to dynamically adjust the three PID parameters: KPK_PKP, KIK_IKI, and KDK_DKD. In the second structure, RBFNN-PID2, the RBFNN directly modifies the output of the PID Compact controller. The control target in both cases is the operating temperature of an N-channel metal–oxide–semiconductor field-effect transistor (MOSFET), specifically a modified IRFZ44N module based on the TCLab kit [16].

2.0 EXPERIMENTAL SETUP

The proposed controllers were implemented and tested using an experimental setup based on the Siemens S7-1200 PLC, located in the PLC and Industrial Internet of Things (PLC&IIoT) Laboratory, College of Engineering, Can Tho University, Vietnam. The overall hardware layout and the connection between the PLC and the control plant are illustrated in Figure 1. A 7-inch Human-Machine Interface (HMI) screen was employed to monitor system behavior and adjust controller parameters in real time. The I/O expansion module served as the interface between the S7-1200 PLC and the external components of the system. Key technical specifications of the PLC used in this study are summarized in Table 1.

Figure 2 illustrates the schematic diagram of the temperature plant. This circuit is modified from the TCLab kit. The heater *Q1* was replaced by a N-channel MOSFET based on the voltage divider circuit, var-resistor *VR1* is used to adjust the bias voltage at the *G* gate. The heating rate of *Q1* is proportional to the current I_d from the *D* gate to the *S* gate or the average voltage V_G at the gate. In addition, the V_G is the input of the plant that is modified by the pulse-width modulation (PWM) signal generated by S7-1200 to change the operating temperature of *Q1*. *Q1*'s temperature is fed back to the controller by a temperature sensor named LM35 [18] acquired by the analog input of the S7-1200. The acquired temperature is filtered by a moving-average filter (MAF) before feeding back to the controller to calculate the error. The mathematics of the MAF is represented as follows.

$$t_f = \frac{1}{N}\sum_{i=0}^{N} t_i \tag{1}$$

where t_f is the filtered temperature, t_i is the raw temperature, N is the filter window size, N = 5 utilized in this paper. Biased-voltage V_G of *Q1* is calculated by the following equation.

$$V_G = \frac{R_1}{R_1 + VR_1} V_{PWM} \tag{2}$$

where R_1 and *VR1* are the resistance of voltage divider circuit; V_{PWM} is the average voltage of the PWM signal from the S7-1200. V_{PWM}'s maximum is 24 VDC when PWM signal reaches 100% duty. The load resistor is not used to guarantee the heat emission from *Q1*. V_G should be controlled below 2.0 VDC to avoid overcurrent. Figure 3 presents the temperature curve of *Q1* when continuously applying a 1 kHz, 80% duty PWM signal to the PWM input corresponding to some values of V_G. These curves show that *Q1*'s temperature behavior is dramatically changed due to a minor altering of V_G. This is the main difference of this temperature plant in comparing with the original one of the TCLab kit.

Table 1 S7-1200 (CPU 1215C) basic parameters [17]

Features		Value
User memory	Work	100 Kbytes
	Load	4 Mbytes
	Retentive	10 Kbytes
Local on-board IO	Digital	14 inputs/10 outputs
	Analog	2 inputs
Pulse outputs		4

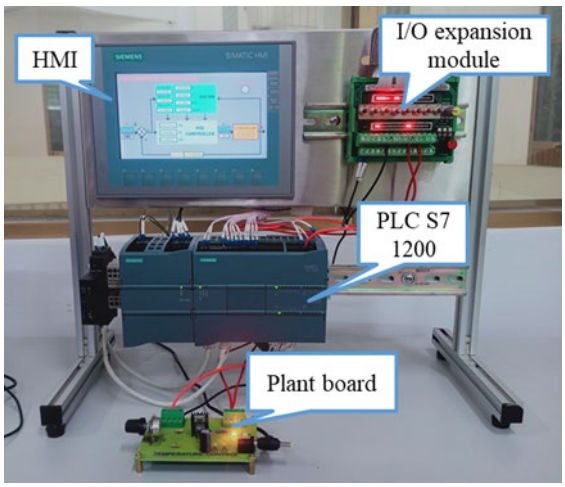


Figure 1 Experimental layout

Consistent reference in testing cases is essential in this paper because some controller structures are tested at different time points. To cope with this problem, the reference temperature is generated by Function Block (FB) written in the SCL language to make sure all controllers have the same reference. Using the same reference, the control rules and responses could be compared easily. The data are automatically acquired by SCADA features of the WINCC tool and stored in Excel format to analyze the control quality and to make graphs.

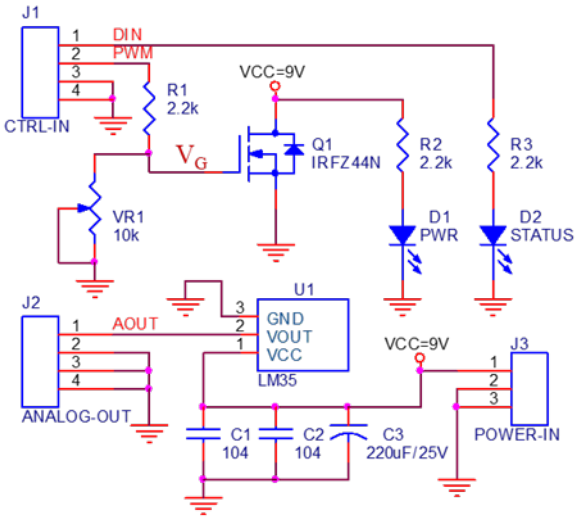


Figure 2 Schematic diagram of the control plant

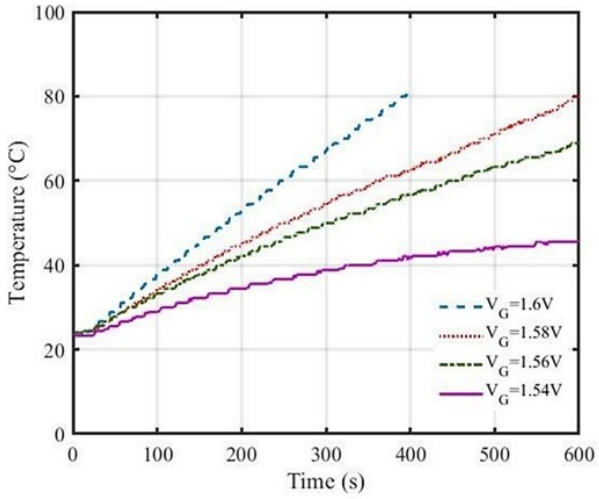


Figure 3 Temperature of MOSFET at various biased levels of V_G

3.0 RESULTS AND DISCUSSION

3.1 Implementation of RBFNN-PID on the PLC

The RBFNN described above is directly deployed on the PLC to implement a self-tuning PID controller. The key principle behind the proposed control structures is that the primary control effort is generated by the PID controller, while the RBFNN serves to fine-tune the system response. This design allows the RBFNN to enhance performance without replacing the conventional controller, making the proposed structures practical and suitable for real-world applications.

3.1.1 RBFNN-PID1 structure

RBFNN is a 3-layer feed-forward neural network including only one hidden layer activated by the RBF function [19]. This layer comprises an array of computing units called hidden nodes [20]. The projection from inputs to output is nonlinear, except from the hidden layer to output is linear. RBFNNs have been proved that it can approximate any continuous functions with any arbitrary accuracy [21]. Figure 4 depicts a general structure of a RBFNN. The output of the RBFNN y_m can be expressed by the following equation.

$$y_m(k) = \mathrm{W}^T h = w_1 h_1 + w_2 h_2 + \ldots + w_m h_m \quad (3)$$

where, W is the weight vector $W = [w_1, w_2, \ldots, w_m]^T$; h is the output vector of the Gaussian function $h = [h_1, h_2, \ldots, h_m]$; and the equation of h_j is calculated as follows.

$$h_j = \exp\left[-\frac{\|x - C_j\|^2}{2b_j^2}\right], j = 1, 2, \ldots, m \quad (4)$$

where, x is the input vector $x = [x_1, x_2, \ldots, x_n]^T$; C_j is the center vector of the j^{th} node $C_j = [c_{j1}, c_{j2}, \ldots, c_{ji}, \ldots, c_{jn}]^T$; and b_j is the width of the j^{th} node of the Gaussian function.

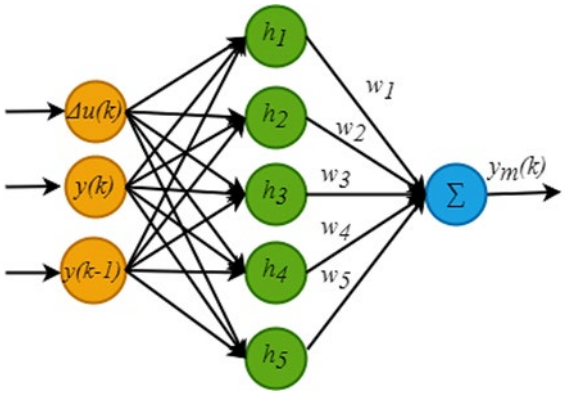


Figure 4 RBFNN structure of the RBFNN-PID1

Figure 5 illustrates the structure of the RBFNN-PID1. In this structure, the RBFNN plays a role of fine tuning the parameters K_P, K_I, and K_D of the PID compact by the corresponding values ΔK_P, ΔK_I and ΔK_D, respectively. The RBFNN with a [3-5-1] structure comprises 3 inputs, 5 hidden nodes, and 1 output. The input vector is $x = [\Delta u(k), y(k), y(k-1)]^T$ while $\Delta u(k)$ is the output of PID; $y(k)$ is the output of the plant at k time; and $y(k-1)$ is the previous output of the plant at the $k-1$ time. The output vector of the Gaussian function of hidden nodes is $h = [h_1, h_2, \ldots, h_j]^T$, j = 5, on which h_j is calculated by the equation (4), the center values of the RBFs is a 3x5 matrix as follows.

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1j} \\ \vdots & \ddots & \vdots \\ c_{i1} & \cdots & c_{ij} \end{bmatrix}, i = 3, and\ j = 5$$

The width vector of the j^{th} node of the Gaussian function is $b = [b_1, b_2, \ldots, b_5]^T$. The weight vector of the network is $W = [w_1, w_2, \ldots, w_5]^T$. The output $y_m(k)$ of this RBFNN is estimated by the equation (3). This output is used to estimate the update rules for the PID's parameters.

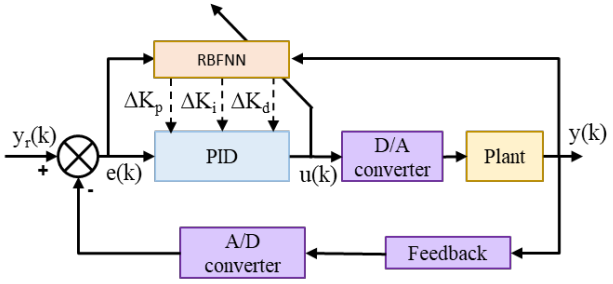


Figure 5 RBFNN-PID1 structure

In this paper, b and c are constant and their values are chosen by try-and-error method. Only the weight vector W is continuously adjusted by Applying the gradient descent method during the operation of the controller, the update rule of the weight vector is computed as follows.

$$\Delta w_j(k) = \eta(y(k) - y_m(k)h_j \tag{5}$$

$$w_j(k) = w_j(k-1) - \Delta w_j(k) + \alpha(w_j(k-1) - w_j(k-2)) \tag{6}$$

where, $\in(0,1)$ is the learning rate and $\in(0,1)$ is the momentum factor. The learning rate η and the momentum factor α are also modified by experiments to find the optimal values. The Jacobian is computed by the following equation.

$$\frac{\partial y(k)}{\partial \Delta u(k)} = \sum_{j=1}^{m} w_j h_j \frac{c_{ji} - x_1}{b_j^2} \tag{7}$$

where, $x1 = \Delta u(k)$.

The error of the controller is defined by equation (8) as shown below.

$$e(k) = y_d(k) - y(k) \tag{8}$$

Three inputs of the controller are estimated by following equations:

$$x_c(1) = e(k) - e(k-1) \tag{9}$$

$$x_c(2) = e(k) \tag{10}$$

$$x_c(3) = e(k) - 2e(k-1) + e(k-2) \tag{11}$$

Then the incremental PID control algorithm is written as follows:

$$\Delta u(k) = K_p[e(k) - e(k-1)] + K_i e(k) + K_d[e(k) - 2e(k-1) + e(k-2)] \tag{12}$$

Neural network-tuning of indicators is as follows:

$$E(k) = \frac{1}{2}e(k)^2 \tag{13}$$

The control parameters of the PID controller are adjusted based on the gradient descent method are written as follows [22]:

$$\Delta k_p = \eta e(k)\frac{\partial y(k)}{\partial \Delta u(k)}x_c(1) = \eta e(k)[\sum_{j=1}^{m} w_j h_j \frac{c_{ji} - x_1}{b_j^2}]x_c(1) \tag{14}$$

$$\Delta k_i = \eta e(k)\frac{\partial y(k)}{\partial \Delta u(k)}x_c(2) = \eta e(k)[\sum_{j=1}^{m} w_j h_j \frac{c_{ji} - x_1}{b_j^2}]x_c(2) \tag{15}$$

$$\Delta k_d = \eta e(k)\frac{\partial y(k)}{\partial \Delta u(k)}x_c(3) = \eta e(k)[\sum_{j=1}^{m} w_j h_j \frac{c_{ji} - x_1}{b_j^2}]x_c(3) \tag{16}$$

3.1.2. RBFNN-PID2 structure

Instead of adjusting the parameters of the PID controller, the RBFNN can also be applied to directly tuning the output of the PID by using the second structure named RBFNN-PID2 modified

from the structure of [20] and [23]. In this controller, the RBFNN has the same [3-5-1] structure as the RBFNN-PID1 which is different from the RBFNN input proposed by [20] and [23]. Figure 6 illustrates the detailed structure of the RBFNN-PID2 controller.

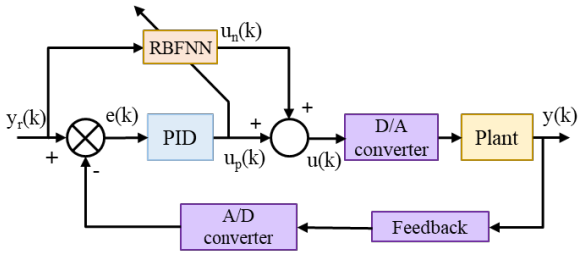


Figure 6 RBFNN-PID2 structure

3.2 Implementation of RBFNN on the PLC

To date, RBFNNs have not been natively integrated into PLC platforms. In previous studies, RBFNNs were typically executed on a PC, with control signals transmitted to the PLC via an OPC (Open Platform Communications) server [24]. In contrast, this paper proposes a direct implementation of RBFNNs on a PLC using SCL. SCL is a high-level, Pascal-based programming language widely supported in Siemens PLCs, and it allows for the development of customized FB. SCL provides built-in support for 1-D and 2-D arrays, as well as a wide range of mathematical functions. Vectors and matrices in RBFNNs can be managed using arrays. For example, the weight vector *W* is declared by 1-D array as follows:

W: array[0..4] of real := 5(5.0);

The center matrix *C* is declared by the 2-D array as the following code:

C: array[0..2, 0..4] of real;

According to the equations (3), (4), (5), (6), and (8), the RBFNN in Figure 5 is implemented as Algorithm 1. All vector and matrix operators are estimated indirectly by manipulating arrays and FOR loops. All *b*, *W*, and *C* are initialized manually and adjusted by try-and-error method to get the optimal values.

Algorithm 1: The algorithm of the RBFNN in SCL language

Input : u(k), y(k), y(k-1)
Output : ym(k)

```
IF #ENABLE = TRUE AND #temp = FALSE THEN
    // Variables initial
    #temp := #ENABLE;
    #x_i[0] := #D_U;
    #x_i[1] := #Y;
    #x_i[2] := #Y1;
    // h vector calculation [Equ. 2 in the paper]
    FOR #j := 0 TO 4 DO
        #h[#j] := EXP_REAL(- ((#c_temp_1[#j] *
        #c_temp_1[#j] + #c_temp_2[#j] * #c_temp_2[#j]  +
        #c_temp_3[#j] * #c_temp_3[#j]) / (2 * #b_0[#j] *
        #b_0[#j])));
        #ym := #ym + #w[#j] * #h[#j];
    END_FOR;
    //Error calculation
     #em := #Y - #ym;
    FOR #j := 0 TO 4 DO
        #w_1[#j] := #w[#j];
    END_FOR;
    // delta w(j) and w(j) calculation [Equ. 4 and 5 in
    //the paper]
    FOR #j := 0 TO 4 DO
        #d_w[#j] := #xite * #em * #h[#j];
        #w[#j] := #w_1[#j] + #d_w[#j] + #alpha * (#w_1[#j] -
        #w_2[#j]);
    END_FOR;
    // END of RBFNN
ELSE
    #temp := #ENABLE;
END_IF;
```

3.3 Implementation of RBFNN-PIDs in PLC S7-1200

The PID controller can be applied in the PLC program by three main approaches. The first method is using the PID compact, a built-in FB of the S7-1200 PLC, with the auto tuning tool to support users to find suitable parameters for the new plant [25]. This is the simplest method, but the output of PID compact may fluctuate at high frequency because of the nonoptimal parameters, influencing actuators for long time. The second method is using the PLC coder toolbox of MATLAB/Simulink to automatically generate the SCL code for the PID controller designed on MATLAB/Simulink [26]. The third method is writing the PID algorithm from scratch by using the SCL language. This is the most difficult one because users must have not only knowledge about the mathematics of PID but also have strong skills in writing SCL programs. In this paper, the first method is applied to build the PID controller for the S7-1200 PLC. However, the initial parameters were manually adjusted based on the Ziegler–Nichols method [27] and optimized by the RBFNN instead of using the auto-tuning tool.

Both structures RBFNN-PID1 and RBFNN-PID2 are implemented and tested with the modified temperature plant in Figure 2. The updating rules of the RBFNN-PID1 are based on (7), (8), (9), (10), (11), (12), (14), (15), and equations (16) that are written in the SCL language as depicted in Algorithm 2. The RBFNN-PID2 is a supervisory controller and the output ym of RBFNN directly adjusts the output of the PID compact (i.e., refer Figure 6), the control rule is as follows.

$$u(k) = u_p(k) + y_m(k) \tag{17}$$

Algorithm 2: The algorithm of the RBFNN-PID1 in SCL language

Input : u(k), y(k), y(k-1)
Output : delta_Kp, delta_Ki, delta_Kd

```
IF #ENABLE = TRUE AND #temp = FALSE THEN
    // ...
    // RBFNN SCL Code of Algorithm 1
    // ...
    // PID controller's parameters updating rules
    //Jacobian calculation [Equ. 20 in the paper]
     #jacobian := 0;
    FOR #j := 0 TO 4 DO
        #jacobian := #jacobian + (#w[#j] * #h[#j] * (#c_1[0, #j]
        - #D_U)) / (#b_0[#j] * #b_0[#j]);
          #w_2[#j] := #w_1[#j];
```

```
    END_FOR;
    // xc calculation [Equ 12, 13, 14 in the paper]
    #ek2 := #ek1; #ek1 := #ek; #ek := #e;
    #xc_1 := #ek - #ek1; #xc_2 := #ek;
    #xc_3 := #ek - 2 * #ek1 + #ek2;
    // Delta Kp, Delta Ki, and Delta Kd calculation
    // [Equ 17, 18, 19 in the paper]
    #delta_Kp := (#eta * #ek * #jacobian * #xc_1);
    #delta_Ki := (#eta * #ek * #jacobian * #xc_2);
    #delta_Kd := (#eta * #ek * #jacobian * #xc_3);
    #YM_1 := #ym;
    #Y1 := #x_i[1];
ELSE
    #temp := #ENABLE;
END_IF;
```

This equation can be carried out easily by LAD's block or SCL commands. The output and response of the PID compact are tuned by the auto tuning tool and are also acquired to analyze and compare with RBFNN-PID structures.

The fluctuation of the LM35 sensor's output should be reduced. In fact, the temperature of the plant is acquired by a low precision LM35 sensor, the accuracy is +/-0.5 °C. At a specific temperature, LM35's output fluctuates randomly in a bound of 1 °C, it may cause a high frequency switching at the output of the PID controller if its parameters are not suitable and may reduce the lifetime of the actuators. The MAF should be applied to smooth the feedback to avoid the fluctuation of the PID's control output. The MAF filter is also involved to the PLC program by using the SCL language based on the equation (18), and *N* equals 5.

$$t_{out} = \frac{1}{N}\sum_{i=0}^{N} x_i \tag{18}$$

with t_{out} is the post-filter temperature feeding back to the controller; x_i is the temperature samples within the sliding window; *N* is the size of the sliding window. The SCL code of MAF is presented in Algorithm 3.

Algorithm 3: The algorithm of the MAF in SCL language

```
Input  : I_Raw_Value
Output : Q_MAV
IF #I_Reset THEN
    FOR #i := 0 TO 4 DO
        #Buffer[#i] := 0;
    END_FOR;
    #i := 0;
    #Full := False; // Reset Full Pulse
END_IF;
IF #i > 4 THEN// Reset ring buffer index
    #i := 0;
END_IF;
// Update Sum
#Sum := #Sum + # I_Raw_Value – #Buffer[#i];
// Store new raw value
#Buffer[#i] := # I_Raw_Value;
// Calculate Moving Average
#Q_MAV := #Sum / 5.0;
IF #i =4 THEN   // Buffer full test
    #Full := True;
END_IF;
#i := #i + 1; // Update pointer
```

The RBFNN-PID FB is invoked by a periodic Organization Block (OB) in the S7-1200 PLC at a rate of 2 Hz to dynamically adjust the parameters of the PID Compact FB. In parallel, the PID Compact FB is executed by another periodic OB at a rate of 100 Hz to continuously regulate the plant temperature. All OBs are implemented using the LAD language. Figure 7 presents the LAD program to test the RBFNN-PID1 structure. The RBF FB runs the RBFNN to estimate *Delta_Kp*, *Delta_Ki*, and *Delta_Kd*. These values are used to respectively adjust K_P, K_I, and K_D of the PID compact, *Pid coefficient process* FB. Ranges of PID compact's parameters are limited to certain ranges to guarantee that the PID compact is not unstable by this adjustment. These ranges are depicted in Table 2. The *ANALOG* FB reads the voltage from LM35's output, converts to the plant's temperature, and feeds to the MAF before applying to estimate the error for the RBFNN-PID. The main point in this paper is that the RBFNN only adjusts the PID's parameters when error drops in the range from -3 °C to 3 °C, while the main controlling energy is for out of this range from the PID compact. The purpose of this control strategy is to avoid the RBFNN causing the unstable status for the PID controller. The output of PID controller *OUT_PID* is sent to the pulse training module of the S7-1200 PLC to generate the PWM signal in order to modify the plant's temperature.

It is also necessary to supervise and modify all parameters of the program. The users could supervise the behavior of the system and easily modify if necessary. Figure 8 shows the SCADA application on the HMI screen to monitor and adjust the RBFNN-PID1 program. From this screen, all constants including *b*, *c*, *eta* (η), *alpha* (α), and y_r could be modified easily by the built-in keyboard. Besides, the outputs as *Delta_Kp*, *Delta_Ki*, *Delta_Kd*, *u*, and the feedback temperature are also visualized for supervisory purposes.

Vital historical data are also recorded and stored to access the controllers. The acquisition data feature of the WINCC program in the TIA Portal is used to acquire data from PLC's tags and store them in the csv extension files. The plant's output *y*, *Delta_Kp*, *Delta_Ki*, *Delta_Kd*, u_p, u_n, and the controller's outputs are sampled every second. This data is mainly used for accessing and comparing the control quality.

During the experimental process, the coefficients *b*, *W*, and *C* of the RBFNN were initialized manually as follows:

$$b = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 \end{bmatrix}^T$$

$$W = \begin{bmatrix} 5 & 5 & 5 & 5 & 5 \end{bmatrix}^T$$

$$C = \begin{bmatrix} -10 & -5 & 0 & 10 & 5 \\ -10 & -5 & 0 & 10 & 5 \\ -10 & -5 & 0 & 10 & 5 \end{bmatrix}$$

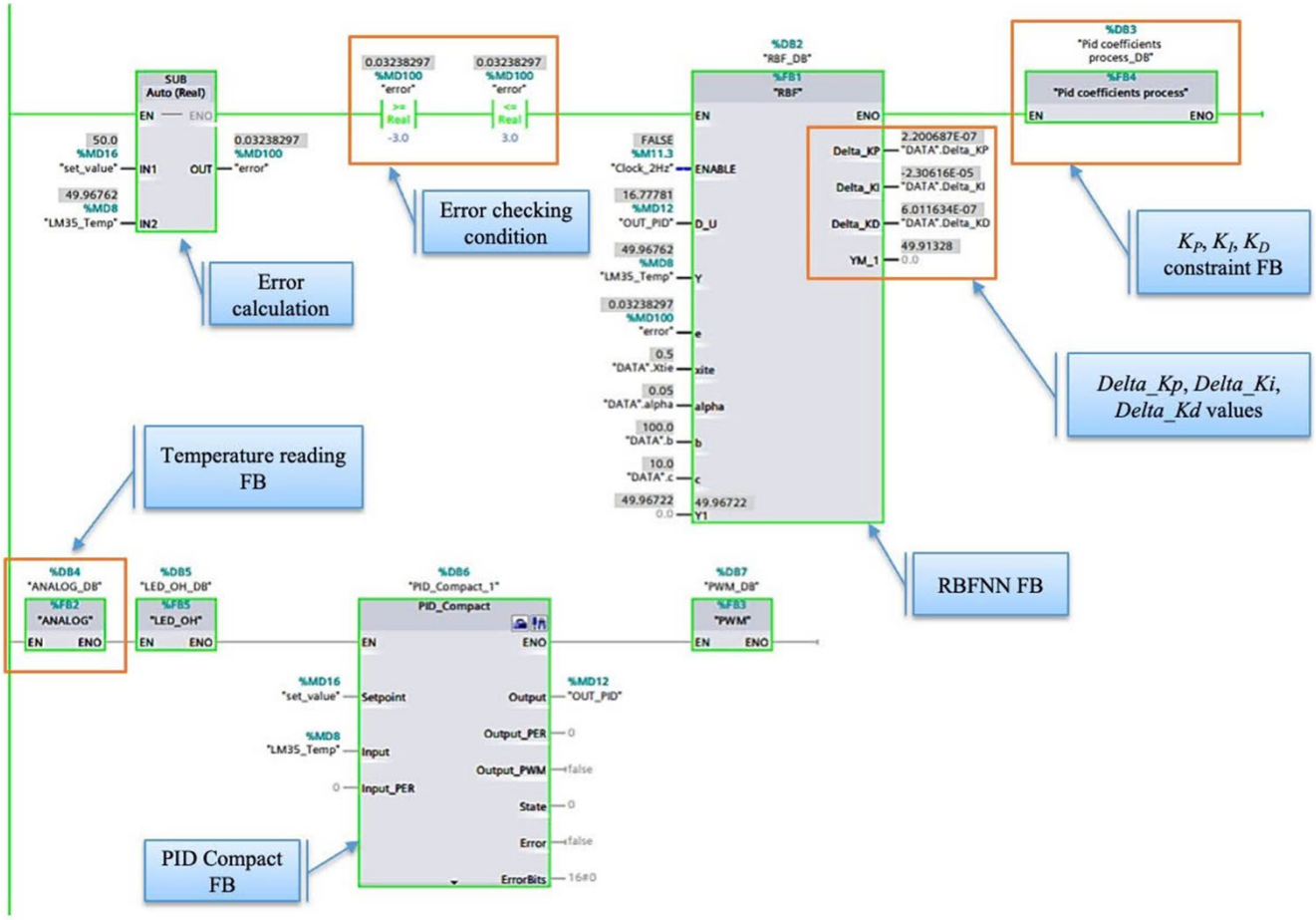


Figure 7 Ladder program network

Table 2 PID controller's parameters tuning constraint

PID parameter	Lower limit	Upper limit
K_P	0.5	0.7
K_I	0.01335	0.01875
K_D	0.0033375	0.0046875

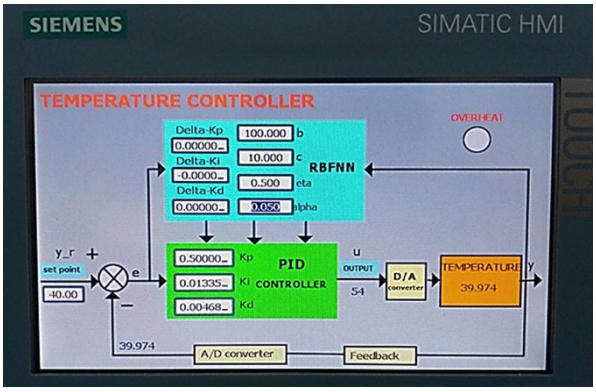


Figure 8 SCADA application on the HMI screen

4.0 RESULTS AND DISCUSSION

4.1 S7-1200's built-in PID compact module response

The PID Compact FB has been applied to control the plant's temperature to compare with the designed controllers. PID compact's parameters are automatically recognized by the built-in auto-tunning tool. The system's tracking performance is illustrated in Figure 9a, which shows that the plant's temperature follows the reference signal accurately, with negligible steady-state error and no overshoot. However, the control signal exhibits high-frequency switching behavior due to the fluctuation of the output of LM35 sensor as shown in Figure 9b. As mentioned above, the MAF filter has been used to smooth the feedback temperature. In fact, after using the MAF both the temperature response and the control signal became significantly more stable. This improvement is clearly illustrated in the control signal profile shown in Figure 9.

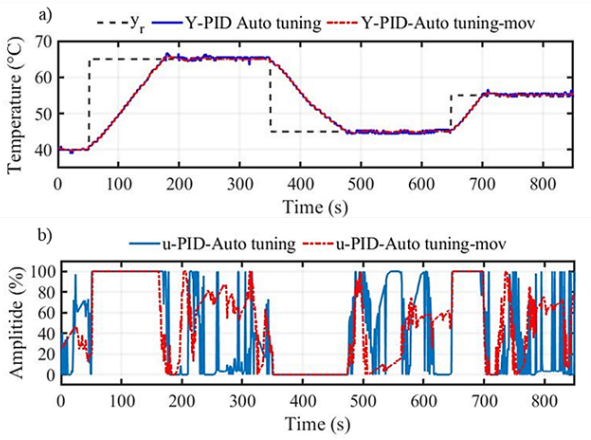


Figure 9 PID compact controller's response: a) plant's temperature, b) controller's output

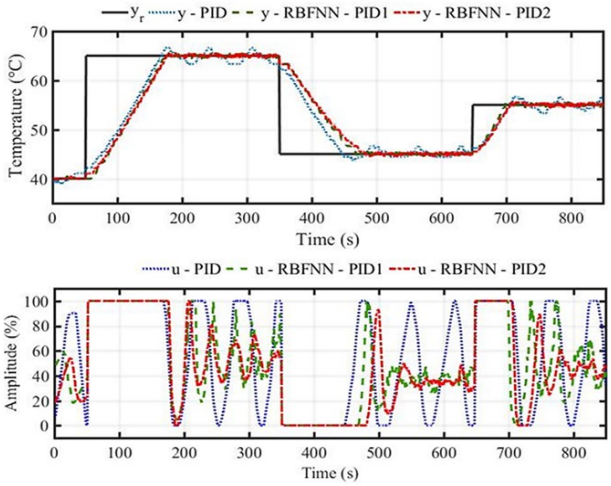


Figure 10 Controllers' comparison

4.2 RBFNN-PID Control Strategies

Although after applying the MAF the response and control rule were improved significantly, PID Compact's output still exhibits occasional fluctuations. This phenomenon may have adverse effects on physical actuators. The switching of control output may occur by the optimal objective of the tuning tool that only focus on optimizing the response and ignore the control rule. To address this limitation, this study proposes the use of a customized autotuning mechanism implemented via a developed RBFNN FB. The central aim is to demonstrate that an intelligent control algorithm, such as RBFNN, can be effectively deployed in a PLC environment to enhance PID performance, particularly when dealing with nonlinear systems. The Ziegler-Nichols method is applied to find the initial parameters for the PID compact. The RBFNN be responsible for optimizing these initial parameters of PID by the RBFNN-PID1 or directly supervising and adjusting the output of the PID compact using the RBFNN-PID2 structure.

Each RBFNN-PID structure influences the behavior of the PID controller in a distinct manner. Figure 11 presents the variation of the PID Compact's parameters under the RBFNN-PID1 strategy. The results indicate that the RBFNN primarily adjusts K_P and K_D, while K_I is relatively minor. This suggests that RBFNN-PID1 focuses on refining the transient and dynamic performance without heavily affecting the accumulation of steady-state error. In contrast, the RBFNN-PID2 structure directly adjusts the control output of the PID Compact rather than tuning its internal parameters. As shown in Figure 12, the evolution of the RBFNN output (u_n) and the original PID output (u_p) during the tracking phase is presented. The total control signal u—as shown in Figure 10—is the sum of u_p and u_n. Similar to RBFNN-PID1, the primary control effort is contributed by the PID controller, particularly when the tracking error is large. In such cases, u_n remains zero. When the error falls within a predefined threshold, un becomes active and is added to fine-tune the overall control signal. This adaptive behavior ensures that the RBFNN refines the control action only when the system approaches the desired state, thereby reducing the risk of destabilizing the controller.

Each RBFNN-PID structure influences the behavior of the PID controller in a distinct manner. Figure 11 presents the variation of the PID Compact's parameters under the RBFNN-PID1 strategy. This graph implies that the RBFNN mainly adjusts K_P and K_D of the PID, while K_I is minorly modified to improve the control response. In contrast, the RBFNN-PID2 directly alters the PID compact's control rule instead of modifying its parameters. Figure 12 depicts the changing of the RBFNN output u_n, PID's output u_p while tracking the reference. The sum of them is the control rule u of RBFNN-PID2 in Figure 10. Similar to the RBFNN-PID1 structure, the main control energy still comes from PID's output due to large error, and by this time, the value of u_n is 0; when the error is in a certain range, u_n is added to u_p to adapt u.

The control qualities of the RBFNN-PIDs and PID compact controllers are also estimated and summarized in Tables 3 and 4 to serve for comparison. In the case of modifying the reference from 40 °C to 65 °C (i.e., increasing 25 °C), both RBFNN-PID1 and 2 give a better response than the PID compact. These tables show that all the quality parameters are improved, especially the overshoot times and steady state errors are improved 48.1% and 30%, respectively. In the case of altering the reference from 65 °C to 45 °C (i.e., decreasing 20 °C), RBFNN-PIDs only refine the rising times and steady state errors in comparing with the PID compact, while the overshoot times of RBFNN-PID1 and 2 are bigger than PID compact up to 28.6% and 21.7%, respectively, whereas the steady state error of the RBFNN-PID1 is only better than PID compact 7%.

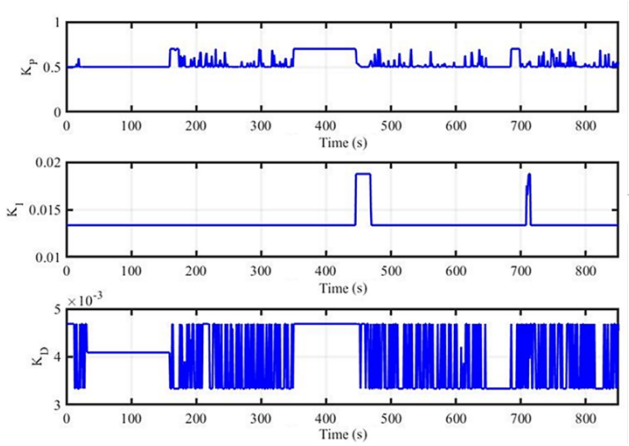


Figure 11 K_P, K_I, and K_D values

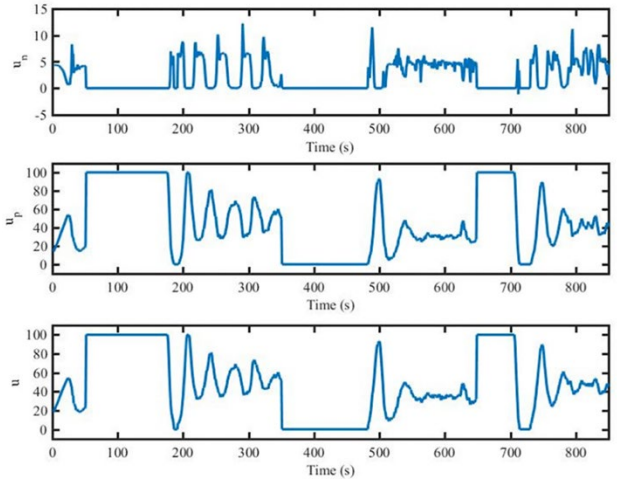


Figure 12 Adapting rule of RBFNN-PID2

Table 3 Response of the controllers after changing the yr from 40 °C to 65 °C

Controller	Rise time [s]	Overshoot [%]	Settling time [s]	Steady state error [°C]
PID Auto tuning	60	1.54	135	1.60
RBF-PID1	55	0.80	123	1.12
RBF-PID2	56	0.80	120	1.12

Table 4 Response of the controllers after changing the yr from 65 °C to 45 °C

Controller	Rise time [s]	Overshoot [%]	Settling time [s]	Steady state error [°C]
PID Auto tuning	53	0.90	133	1.00
RBF-PID1	44	1.26	121	1.07
RBF-PID2	46	1.15	116	1.00

4.3 Discussions

The RBFNN has been successfully integrated into a Siemens S7-1200 PLC through direct implementation using the SCL. The developed RBFNN FB was verified via two self-tuning PID controller structures. Additionally, a MAF was implemented using SCL to smooth the feedback signal and enhance the reliability and performance of the control system. Both RBFNN-PID control structures demonstrated superior performance in terms of eliminating overshoot, reducing steady-state error, and ensuring stable control signals when compared to the baseline PID controller. The proposed hybrid strategy—where the PID controller provides the primary control effort while the RBFNN refines the response—proves to be a practical and robust solution for nonlinear systems.

The successful deployment of RBFNN on the PLC not only confirms the suitability of PLCs for industrial automation but also highlights their potential as platforms for implementing and testing advanced control algorithms in educational and research environments. With modern PLCs equipped with high-performance processors, ample memory, and versatile communication interfaces, they are increasingly capable of supporting sophisticated control strategies. Moreover, the ability to incorporate user-defined function blocks allows for the implementation of non-native or advanced algorithms. Alongside manual programming via SCL, toolboxes such as MATLAB/Simulink's PLC Coder offer a convenient pathway to automatically generate SCL code from control models. This flexibility significantly expands the usability of PLCs for a broad range of control applications beyond conventional industrial use.

5.0 CONCLUSION

The RBFNN has been successfully deployed on a Siemens S7-1200 PLC using the SCL. To evaluate its performance, two self-tuning PID control structures were developed and tested. In these configurations, an initially non-optimal PID controller—tuned using the Ziegler–Nichols method—was adaptively improved by the RBFNN through either dynamic adjustment of PID parameters or direct modification of the control output. Experimental results show that the RBFNN-PID controllers achieved comparable, and in many cases superior, performance to the built-in PID Compact with auto-tuning. Notably, the proposed controllers demonstrated enhanced control quality, reducing overshoot by 48.1% and steady-state error by 30% compared to the conventional PID Compact. Furthermore, the output signals of the RBFNN-PID controllers exhibited minimal high-frequency fluctuations, which is beneficial for actuator longevity and overall system stability. The implemented RBFNN structure is modular, scalable, and transferable across other PLCs that support SCL, making it highly practical for broader industrial use. In addition, a simple MAF was also developed in SCL to attenuate sensor noise, contributing significantly to response smoothness and accuracy. Overall, this work demonstrates that modern PLCs are capable of hosting intelligent control algorithms such as RBFNNs. This capability not only opens doors for advanced automation solutions in industrial settings but also creates new opportunities for control education and research. The study highlights the potential of

PLCs as versatile platforms for both practical deployment and academic experimentation in adaptive and intelligent control systems. For future work, integrating the proposed system into real-time industrial networks for distributed control and diagnostics will further validate its robustness. The expansion of the control strategy to multi-input multi-output (MIMO) systems and time-varying nonlinear plants will also be considered to extend the applicability of the proposed approach.

Acknowledgment

The authors gratefully acknowledge the Faculty of Automation Engineering, College of Engineering, Can Tho University, for providing the equipment necessary to conduct this research.

Conflicts Of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper

References

[1] Rallabandi, S. R., Yanda, S., Rao, C. J., Ramakrishna, B., and Apparao, D. 2023. Development of a color-code sorting machine operating with a pneumatic and programmable logic control. *Materials Today: Proceedings*.In Press. DOI: 10.1016/j.matpr.2023.05.150

[2] Pandey, K., Singh, K. G., and Singh, A. 2023. Multi-sensors based smart nutrient reuse management system for closed soilless culture under protected cultivation. *Computers and Electronics in Agriculture*. 204: 107495. DOI: 10.1016/j.compag.2022.107495

[3] Maesschalck, S., Staves, A., Derbyshire, R., Green, B., and Hutchison, D. 2023. Walking under the ladder logic: PLC-VBS: a PLC control logic vulnerability scanning tool. *Computers & Security*. 127: 103116. DOI: 10.1016/j.cose.2023.103116

[4] Farahpour, H., and Hejazi, F. 2023, May. Development of integrated semi-active adaptive vibration control system for bridges subjected to traffic loads. *In Structures.* 51: 1773-1794. Elsevier. DOI: 10.1016/j.istruc.2023.03.107

[5] Osman, K., Štefić, M., Alajbeg, T., and Perić, M. 2023. Approach in development and implementation of automatic control system in a smart building. *Energy and buildings.* 293: 113200. DOI: 10.1016/j.enbuild.2023.113200

[6] Üser, Y., & Kara, C. 2019. Real-Time Communication between S7-1200 PLC and Matlab/Simulink and a Fuzzy Logic Temperature Humidity Control Application. *Scientific Journal of Mehmet Akif Ersoy University.* 2(1): 7-14.

[7] Barrera-Cuestas, A. F., Mantilla-Castañeda, M. G., Giral-Ramírez, D. A., & Montoya-Giraldo, O. D. 2021. Temperature control using thesimulink PLC Coder and the IEC 61131 standard. Scientia Et Technica. 26(4): 417-423. DOI: 10.22517/23447214.24947

[8] Ionescu, D., Filipescu, A., Simion, G., Mincă, E., Cernega, D., Șolea, R., & Filipescu, A. 2022. Communication and control of an assembly, disassembly and repair flexible manufacturing technology on a mechatronics line assisted by an autonomous robotic system. *Inventions.* 7(2): 43. DOI: 10.3390/inventions7020043

[9] Yilmaz, S., & Furat, M. 2023. A real-time cost optimization of two-section oven system with discrete gradient extremum seeking control: An experimental study in iron and steel industry. *Journal of Process Control.* 122: 84-99. DOI: 10.1016/j.jprocont.2022.12.005

[10] Bélai, I. 2019. The Vehicle Model Acceleration Control for Platooning. *IFAC-PapersOnLine.* 52(27): 157-162. DOI: 10.1016/j.ifacol.2019.12.749

[11] Rybczak, M., Popowniak, N., & Kozakiewicz, K. 2022. Applied AI with PLC and IRB1200. *Applied Sciences.* 12(24), 12918. DOI: 10.3390/app122412918

[12] Mac, T. T. 2021. Application of improved Yolov3 for pill manufacturing system. *IFAC-PapersOnLine.* 54(15): 544-549. DOI: 10.1016/j.ifacol.2021.10.313

[13] Mekala, S. H., Baig, Z., Anwar, A., & Zeadally, S. 2023. Cybersecurity for Industrial IoT (IIoT): Threats, countermeasures, challenges and future directions. *Computer Communications*. 208: 294-320. DOI: 10.1016/j.comcom.2023.06.020

[14] Du, Z., Chen, S., Anduv, B., Zhu, X., & Jin, X. 2023. IoT intelligent agent based cloud management system by integrating machine learning algorithm for HVAC systems. *International Journal of Refrigeration.* 146: 158-173. DOI: 10.1016/j.comcom.2023.06.020

[15] Ghosh, A., Wang, G. N., & Lee, J. 2020. A novel automata and neural network based fault diagnosis system for PLC controlled manufacturing systems. *Computers & Industrial Engineering*. 139: 106188. DOI: 10.1016/j.cie.2019.106188

[16] de Moura Oliveira, P. B., Hedengren, J. D., & Rossiter, J. A. 2020. Introducing digital controllers to undergraduate students using the tclab arduino kit. *IFAC-PapersOnLine.* 53(2): 17524-17529. DOI: 10.1016/j.ifacol.2020.12.2662

[17] SIMATIC, S., 2019. s7-1200 Programmable controller. System manual, 02/2019, A5E02486680-AM, 6.

[18] Texas Instruments. 1999. *LM35 Precision Centigrade Temperature Sensors*. SNIS159H Datasheet

[19] Kumpati, S. N., & Kannan, P. 1990. Identification and control of dynamical systems using neural networks. IEEE Transactions on neural networks. 1(1), 4-27.

[20] Liu, J. 2013. *Radial Basis Function (RBF) neural network control for mechanical systems: design, analysis and Matlab simulation*. Springer Science & Business Media.

[21] Xiangsong, K., Xurui, C., & Jiansheng, G. 2016. PID controller design based on radial basis function neural networks for the steam generator level control. *Cybernetics and Information Technologies.* 16(5):15-26. DOI: 10.1515/cait-2016-0048

[22] Nguyen, H. D., & Huynh, T. H. 2018, October. Controlling the Position of the Carriage in Real-Time Using the RBF Neural Network Based PID Controller. In 2018 *18th International Conference on Control, Automation and Systems (ICCAS)* 1418-1423. IEEE.

[23] Ngon, N. C., Tho, N. V., & Phuong, T. T. H. 2022. Supervisory control for the beam and ball system using the radial basis function neural network. Can Tho University Journal of Science. 58(3): 26–35. DOI: 10.22144/ctu.jvn.2022.083

[24] Mahnke, W., et al. 2009. *OPC Unified Architecture*. Springer

[25] Siemens. *PID Control with PID_Compact for SIMATIC S7-1200/S7-1500.* [Online] https://support.industry.siemens.com/cs/ww/en/view/100746401 Retrieved on 25 April, 2023

[26] MathWorks. *Simulink PLC Coder*.[Online] https://www.mathworks.com/products/simulink-plc-coder.html (accessed 25 April, 2023).

[27] Ogata, K. 2020. *Modern Control Engineering*. 5th edn. Pearson Malaysia