A Field Bus Network With CAN Protocol And a Fuzzy Neural Petri Net Controller

Ali A. Abed[1], AbdulAdhem A. Ali[2], Ali F. Marhoon[3], Nauman Aslam[4]

[1,2,3] College of Engineering-University Of Basra, IRAQ.

[4] College of Engineering-Dalhousie University, CANADA.

[1] aaad_bah@yahoo.com, [2] abduladem1@yahoo.com, [3] ali_marhoon2003@yahoo.com,

[4] naslam@dal.ca

Abstract

In field bus systems different smart sensors are functionally interconnected, requiring exchange of information accurately with a specified communication latency and sample time. This paper describes implementation methodologies of efficient distributed real time controllers for monitoring and actuation. Software for these controllers with CAN communication protocol are written. CAN-Based embedded microcontrollers and transceivers are used for the practical implementation. Temperature control by a robust neural fuzzy Petri net (RNFPN) controller based on an indirect forward control structure is also proposed. After offline learning to get the initial weights, the RNFPN is online constructed by concurrent structure/parameter learning. The RNFPN has many advantages when applied to temperature control plants such as: high learning ability which reduces the controller training time, no a priori knowledge of the plant is required which simplifies the design task, and lastly the high control performance. As a case study, the field bus system is tested with a water bath temperature control system. The RNFPN control of the plant is implemented in the host computer connected to the master controller via the RS232 port. Implementation results provide a good indication that the CAN protocol is a high reliable and can be potentially adopted in numerous industrial applications. Also, the RNFPN intelligent controller has a reasonable robustness against disturbance, rapidity and good dynamic performance.

I. INTRODUCTION

The origin of field bus was to replace the point-to-point 4-20mA analog links between the field devices (sensors and actuators) and their controllers (e.g. PLC's, CNC's,…) with a digital single link on which all data is transmitted serially and multiplexed in time. The serial transmission reduces the number of data lines

and increases distance. There are many benefits of using field bus instead of the point-to-point communications such as:

- Flexibility: network extension is easy.
- Larger distance can be satisfied.
- Physical wiring reduction.
- Reduced cost.
- Simple installation, operation and maintenance.
- Interoperability: connecting of units from different manufacturers.
- High reliability.

The field bus played a major role in the modern industry automation development. It is used for interconnecting real-time distributed controllers of the DCS (Distributed Control Systems). It gives the user the freedom in choosing the suitable protocol that best matches his needs without changing the whole automation system architecture. Since the field bus is a network, there is a relation with the famous 7-layers OSI model. However, for real time applications layers 3, 4, 5, and 6 are not considered since they deal with the data transfer among networks. So, the field bus is a 3-layers architecture with physical, data link, and application layers.

The communication protocol is responsible of two main rules in the field bus: medium access control (MAC) and synchronization among the different slaves. Choosing a MAC protocol is a crucial issue in the design of a DCS system. There is a need to know how the field bus works in order to choose the suitable communication protocol for the application. There are many types of field bus such as: BACnet, ISP, CAN, CEBus, WorldFIP, etc [1]. Profibus and WorldFIP are used widely in industrial control applications e.g. [2] [3]. So, in this paper we adopt the CAN (Controller Area Network) protocol for this reason and due to the high specifications of the CAN. The field bus network may then be used in the design of a DCS. There are three main issues that must be considered in the design of a DCS systems:

- The adopted communication protocol.
- The adopted network topology.
- The Interoperability: Using components from different manufacturers.

The design of a DCS should satisfy many requirements such as: QoS (Quality of Service) of the communication protocol, system throughput, system performance, fast response time, and some real time requirements such as sample time.

Soft computing techniques like fuzzy logic, neural networks, and fuzzy neural networks have been used widely in the identification and control of dynamic systems [4]. The merging of fuzzy and Back propagation neural network (BPNN) has inspired new resources for the design of efficient controllers [5]. These types of computing techniques have many advantages, such as: it is used for nonlinear systems, model-free systems, and good self learning abilities. In [6] the BPNN has been applied to temperature control system but the convergence speed is slow, which makes it unsuitable for real time applications. In [7] a neural fuzzy inference network is proposed for

temperature control and proved good learning ability but the inclusion of too many input and output variables increases the network size and decreases the learning speed. In [8], a recurrent neural fuzzy network controller is presented. In [9], a dynamic fuzzy neural network is introduced. In these recurrent neural fuzzy networks i.e. in [8,9], one common characteristic is that the recurrence is achieved by including external feedback. To apply these networks to temperature control problems, they still need to know the order of both control input and network output to participate in the recurrent model. In [5], a TSK-type (Tagagi-Suggeno) recurrent fuzzy neural network is proposed. It proved a good performance but with inverse control structure and it is still need at least 8 input nodes and 10^7 iterations of training to get a sum of square error (SSE) of 3.1. Therefore, in spite of that the fuzzy neural (FNN) has been proved to be a powerful technique in system control but its real time applications may be difficult due to heavy computations especially if huge parameters are to be tuned. In order to overcome this problem, the concept of a Petri net (PN) is included into the FNN to reduce the redundant or inefficient computations and hence improves the control reliability. The PN consists of "places" and "transitions" with the input and output functions [10]. The basic concept of PN is incorporated into the conventional FNN to get what is called NFPN controller.

The rest of the paper is organized as follows: Section II is concerned to the CAN communication protocol with its main features and benefits. Section III deals with main principles and requirements for the DCS system design. Section IV describes the hardware components needed for our field bus network. In section V, the complete description of the RNFPN controller is presented. Section VI concerns to the identification model. In section VII, the detailed design procedures of our proposed online controller are presented. Section VIII presents the simulation results of the RNFPN controller that proves its performance. Section IX presents the overall field bus system setup and test. The last section is specialized with some conclusions, comments and future work.

II. CAN COMMUNICATION PROTOCOL

The CAN bus is a 2 wires serial bus with multi-master capability i.e. multiple nodes (slaves) connecting to a single 2-wires bus can communicate with each other. CAN implements a priority-based bus with a carrier sense multiple access with collision avoidance (CSMA/CA) as a medium access algorithm which is appropriate for sending and receiving real-time messages at baud rates up to 1Mb/s [11]. The arbitration mechanism ensures that one node will have access to the medium at any given time hence reducing collisions. In Ethernet, a device trying to transmit a frame will monitor the line and wait until the line is available then start transmitting.

If 2 or more transmitters start, a collision will occur so all transmitters will stop. This explains why in most Ethernet LAN applications, nodes are connected point-to-point to a bridge or router. Errors in CAN protocol are detected using a CRC check. Therefore, the CAN bus provides a 100% data integrity. All nodes are synchronized on the same bit time period. The group delay cannot exceed a fraction of the bit time period which lead to the maximum throughput to be a function of the bus length e.g. 1 Mb/s is satisfied with 40 m bus length while 50Kb/s with up to 1km length [12].

Each message starts with an address field which is unique. So the arbitration will be completed at the end of address transmission and only one device will carry on with the message transmission. CAN protocol has two addressing schemes: CAN2.0A with 11 bit address and CAN2.0B with 29 bit address. CAN2.0A is used for small networks with 2048 different IDs but actually the number of nodes is limited by the type of the CAN transceiver used. The data field (DLC) in CAN2.0A is 8 bytes. The receiver drives the ACK bit. Bit stuffing provides a minimum density of edges to help devices remain synchronized. The CAN physical medium is a 2-wires bus (one can use UTP cable) terminated with a resistor. Differential signal is used for good immunity against noise.

A typical CAN field bus systems with master-slave mode suitable for industrial applications is shown in Figure 1[3].

III. DISTRIBUTED CONTROL AND NETWORKS

A DCS is a control system in which several controllers are operating simultaneously and cooperatively to perform the desired task. They are divided into two types: tightly-coupled multiprocessor systems sharing a common bus; and loosely-coupled systems where processor-based controllers are located at the points of control and exchange information via a network. There are many physical arrangements for field bus network some of them are: Daisy-chain directly connects the field device to the communication cable as shown in Figure 2; and Multi-drop which use a T-tab (stub) for device connection to the cable as shown in Figure 3. In our work, we used the daisy-chain configuration because it is the simplest.

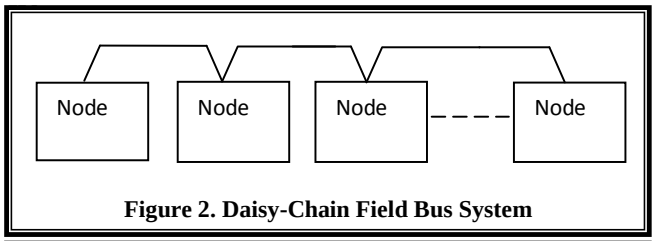

Figure 2. Daisy-Chain Field Bus System

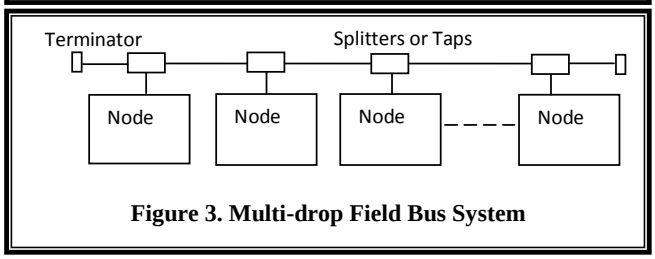

Figure 3. Multi-drop Field Bus System

V. HARDWARE COMPONENTS

Before describing the complete implementation of our system, the hardware components used in

the design should be explained in a little bit detail.

A. The CAN microcontroller (T89C51CC01)

The T89C51CC01 is the first member of the CANary family of 8-bit microcontrollers dedicated to CAN networking. It can reach to 20 MHz external clock rate. It is available in many different pin configurations: PLCC44, PLCC52, VQFP, etc. The detailed specification of this hardware is available in its data sheet [12]. It contains about 99 special function registers (SFR) including 34 CAN SFRs. The bit timing and baud rate for the CAN controller should be calculated and the corresponding registers should be downloaded with the calculated values. This is done through the software that the microcontroller should be programmed with. This software should also contain all the required functions to be performed by the slave controller.

B. The High Speed CAN Transceiver (MCP2551) [13]

The MCP2551 is a high speed CAN, fault-tolerant device that works as an interface between a CAN controller and the physical bus. It provides a differential transmit and receive capability for CAN controllers. Each node in a CAN system must have a device to convert the digital signals (Tx and Rx) generated by the CAN controller to differential output (CANH and CANL). It also works as a buffer between the CAN controller and the high voltage spikes that can be generated on the CAN bus by outside sources. The CAN bus has two states: "Dominant", which occurs when the differential voltage is greater than a defined voltage; "Recessive",- occurs when the differential voltage is less than a defined voltage (typically 0). The two states correspond to the low and high state of the TXD input respectively. The low and high states of the RXD output correspond to the dominant and recessive states of the CAN bus respectively.

The MCP2551 output drives a minimum load of 45 ohms allowing a maximum number of nodes equal to 112 nodes. The termination resistor used here is 120 ohms. The high speed mode is selected by connecting the RS pin of the transceiver chip to Vss.

C. The Water Bath Temperature Control System

The experiment is done on a real water bath to control its temperature. The water bath is an example of an important component in a batch-reactor process. It can be described by the following model [14]:

$$G(s) = \frac{ke^{-t_d s}}{\tau s + 1} \qquad \ldots 1$$

Where k is magnification, τ is the time constant, and t_d is the delay time. In this paper, the water bath is chosen as the controlled object to study the temperature control system . In order to get the parameters of the plant, a step signal is applied to the input of the plant and the S-shape (process reaction) curve is obtained. From this curve we

got the values of k to be 1, t_d is 60 seconds, and τ is 180 seconds. The difference equation derived for this plant is found to be:

$$y(k) = \frac{\tau}{T_S + \tau} y(k-1) + \frac{kT_S}{T_S + \tau} u\left(k - \frac{t_d}{T_S}\right) \quad \dots 2$$

Where T_S is the sampling time that should be chosen accurately. In this paper, T_S is equal to 15 Sec.

VI. STRUCTURE AND LEARNING OF RNFPN

A. *Structure Of RNFPN*

The newly presented RNFPN controller for temperature control system is shown in Figure 4. The difference between this framework and that of the FNN is the transition layer. The operation of each layer of this RNFPN framework is introduced as follows:

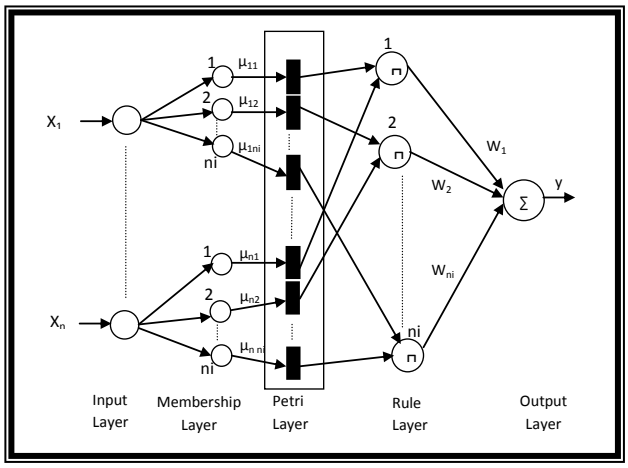

Figure 4: Framework of the RNFPN

1. Input Layer

   Each node in this layer transmits the input crisp variable x_i *(i=1,2,3,…n)* to the next layer.

2. Membership Layer

   This layer represents the fuzzification stage for the RNFPN in which the crisp inputs x_j are transformed to fuzzy inputs via the adopted Gaussian membership function which is given by:

$$\mu_{ij} = exp\left(-\frac{1}{2}\frac{(x_j - c_{ij})^2}{s_{ij}^2}\right) \quad \dots 3$$

   Where c_{ij} and s_{ij} are the center and width of the membership function.

3. Petri Net Layer

   This layer is used to produce tokens that make use of competition laws for node firing as follows:

$$t_{ij} = \begin{cases} 1 \; if \; \mu_{ij} \geq d_{th} \\ 0 \; if \; \mu_{ij} \leq d_{th} \end{cases} \quad \dots 4$$

   Where t_{ij} is the transition and d_{th} is the dynamic threshold that varies with error and can be tuned by the following equation [15]:

$$d_{th} = \frac{\alpha \exp(-\beta E)}{1 + \exp(-\beta E)} \quad \dots 5$$

   Where α and β are positive constants that can be chosen randomly. It is clear that the larger the error is, the smaller the threshold is. If the error becomes large, the threshold values will be decreased to fire more rules for the current situation. Of course one can use a constant value for the threshold. It is important to mention that if the threshold value is chosen to be 0, then the RNFPN system will transformed to FNN system.

4. Rule layer

The output of each node is the product of its inputs and it is given by:

$$\emptyset_j = \begin{cases} \prod_i^n \mu_{ij}\,; & if\ t_{ij} = 1 \\ 0; & if\ t_{ij} = 0 \end{cases} \quad \dots 6$$

Where $\emptyset_j$ is the output of the j^{th} node of the rule layer;

n is the number of crisp inputs.

5. Output Layer

The output node calculates the total output y as a summation of the input signals as follows:

$$y = \sum_j^{ni} w_j \emptyset_j \quad \dots 7$$

Where the connection weights w_j is the output action strength associated with the j^{th} rule; ni is the number of rules.

B. Learning Algorithm Of RNFPN

The task of constructing the RNFPN is divided into two subtasks: *structure learning* and *parameter learning*. In the structure learning, the number of fuzzy rules, initial location of membership functions, and initial consequent parameters are chosen. The parameter learning is used to tune the free parameters of the constructed network to its optimal values. To explain the learning algorithm of RNFPN using the supervised back propagation method, the error function is defined as:

$$E = \frac{1}{2}(y - y_p)^2 \quad \dots 8$$

Where y is the output of RNFPN network and y_p is the output of the plant.

The update laws for w_j, c_{ij}, and s_{ij} are[15]:

$$w_j(k+1) = w_j(k) - \eta_w \frac{\partial E}{\partial w_j}$$

$$c_{ij}(k+1) = c_{ij}(k) - \eta_c \frac{\partial E}{\partial c_{ij}} \quad \dots 9$$

$$s_{ij}(k+1) = s_{ij}(k) - \eta_s \frac{\partial E}{\partial S_{ij}}$$

Where η_w is the learning rate for the weights of rule layer, η_c , η_s are the learning rates for the center and width of the Gaussian membership function respectively. Choosing suitable values for these learning rates is very important during training process. The three partial derivatives in the above three equations are derived and given by:

$$\frac{\partial E}{\partial w_j} = (y - y_p)\emptyset_j$$

$$\frac{\partial E}{\partial c_{ij}} = \begin{cases} (y - y_p) w_j \emptyset_j \dfrac{(x_i - c_{ij})}{s_{ij}^2}\,, & if\ t_{ij} = 1 \\ 0\,, & if\ t_{ij} = 0 \end{cases} \quad \dots 10$$

$$\frac{\partial E}{\partial s_{ij}} = \begin{cases} (y - y_p) w_j \emptyset_j \dfrac{(x_i - c_{ij})^2}{s_{ij}^3}\,, & if\ t_{ij} = 1 \\ 0\,, & if\ t_{ij} = 0 \end{cases}$$

VII. IDENTIFICATION MODEL

For single input single output (SISO), linear, and time invariant plants with unknown parameters,

the identification model may be similar to the following [16]:

$$y_p(k+1) = \sum_{i=0}^{n-1} \propto_i y_p(k-i) + \sum_{j=0}^{m-1} \beta_j u(k-j) \quad \ldots 11$$

Where α_i and β_j are unknown parameters, *u* is the input to the plant, and y_p is the output. Two representations for the identification models are available: Parallel model and Series-Parallel model. In this paper, the second model is adopted.

Series-Parallel Identification Model

This model is obtained by feeding back the past values of the plant output as shown in Figure 5. The identification method depends on the control strategy. The forward control (indirect) strategy requires forward identification and the inverse control (direct) requires inverse identification.

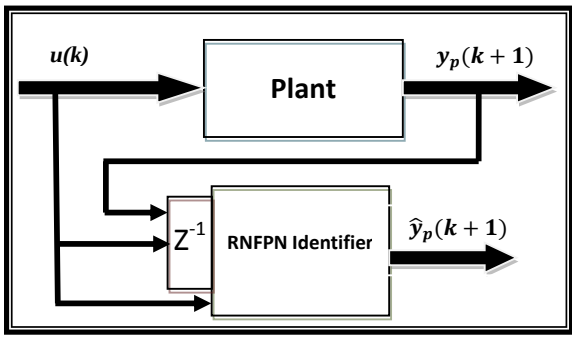

Figure 5: Series-Parallel Identification

In this paper, the forward control with forward identification is used because it is simpler and more accurate than the inverse control. The forward (offline) identification system for our plant is shown in Figure 6.

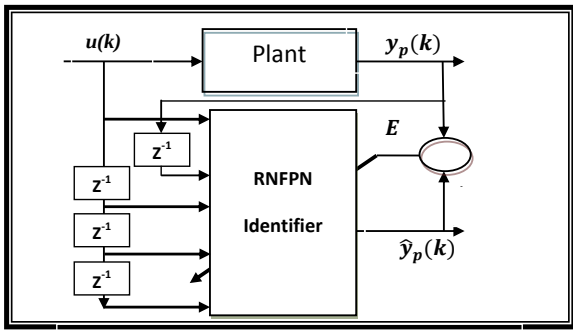

Figure 6: Forward Offline Identification (t_d/T_S=3)

In Figure 6, the value of (t_d/T_S) of the difference equation is 3 i.e. there is only *u(k), u(k-1), u(k-2), and u(k-3)* inputted to the identifier. Of course the number of delays (z^{-1}) depends on the transport lag of the plant.

VIII. CONTROL SYSTEM DESIGN

The PID controllers were used to control industrial temperature systems and some of these systems are still usable today. These systems were used as PID or PI controllers. Today PID controllers are modified to be fuzzy PID controllers. In general, PID controller is a direct controller placed in the feed forward path of the system, it receives the error signal and produces control command according to the value of the error signal, its derivative and its integral. Adaptive control is introduced to deal with complex systems that could not be controlled by conventional methods such as ill-defined plants or time varying parameters plants. In this type of control systems, plant response should follow the response of a defined system. Adaptive controllers could take many forms like Variable Structural Systems (VSS), Self Tuning Systems

(STS) and Model Reference Adaptive Control systems (MRAC). Any of these types of adaptive control systems may be used in the field of temperature control. In this work, the basic concept of PN is used with the FNN to create the presented RNFPN controller for any type of temperature control system. The proposed RNFPN forward (online) control system is shown in Figure 7. This control system, which is based on MRAC, performs two tasks: system identification and control. The initial values for w_j, c_{ij}, and s_{ij} of the identifier should be taken from offline learning that is achieved according to Figure 6. The purpose of the identification process is to overcome the changes that may occur in the system parameters due to disturbance. Also, the control process requires the sensitivity parameter (S) of the plant to be introduced by the identifier. The reference model specifies the desired output performance of the control system i.e. the system will track the desired output of the reference model. Therefore, it is designed as a PID closed loop system with the same plant to be controlled but with different parameters. The RNFPN for the controller and the identifier have the same structure with a small difference in the number and type of the inputs and outputs. The update training laws for the RNFPN is the same as the previously mentioned equations but the error function is given by:

$$\boldsymbol{Ec = \frac{1}{2} ec^2 = \frac{1}{2}(y_r - y_p)^2} \qquad \dots \mathbf{12}$$

It is clear that y_r is the output of the reference model. Now, the partial derivatives $\frac{\partial E_c}{\partial W_{cj}}$, $\frac{\partial E_c}{\partial C_{cij}}$ and $\frac{\partial E_c}{\partial S_{cij}}$ of the updating laws should be derived according to the chain rule as follows:

$$\boldsymbol{\frac{\partial E_c}{\partial W_{cj}} = \frac{\partial E_c}{\partial e_c}\frac{\partial e_c}{\partial y_p}\frac{\partial y_p}{\partial m}\frac{\partial m}{\partial W_{cj}}} \qquad \dots 13$$

Where u is the plant input or the controller output. The term $\boldsymbol{S = \frac{\partial y_p}{\partial m}}$ represents the system sensitivity that should be computed because the convergence of the RNFPN controller cannot be insured if S is unknown. Since the RNFPN identifier of Figure 7 is used to provide the S function and since it has the same plant input and output i.e.:

$$\boldsymbol{\frac{\partial y_p}{\partial m(k)} = \frac{\partial \hat{y}_p}{\partial m(k)}} \text{ then:}$$

$$\boldsymbol{\frac{\partial \hat{y}_p}{\partial m(k)} = S = \frac{\partial \hat{y}_p}{\partial \emptyset_j}\frac{\partial \emptyset_j}{\partial \mu_{ij}}\frac{\partial \mu_{ij}}{\partial m(k)}} \qquad \dots \mathbf{14}$$

Hence, the S function is derived to be:

$$\boldsymbol{S} = \begin{cases} -\sum_{i=1}^{n_i} w_j \emptyset_{cj} \frac{u(k) - C_{ik}}{S_{ik}^2}, & \boldsymbol{if\ \ t_{ij} = 1} \\ \mathbf{0}, & \boldsymbol{if\ \ t_{ij} = 0} \end{cases} \qquad \dots \mathbf{15}$$

Therefore, we can summarize the updating laws for RNFPN controller by:

$$\boldsymbol{\frac{\partial E_c}{\partial W_{cj}} = ec\, S\, \emptyset_{cj}}$$

$$\boldsymbol{\frac{\partial E_c}{\partial C_{cij}}} = \begin{cases} \emptyset_{cj}(X_{ci} - C_{cij}) \frac{W_{cj}\, S\, ec}{S_{cij}^2} & \boldsymbol{if\ t_{ij} = 1} \\ \mathbf{0} & \boldsymbol{if\ t_{ij} = 0} \end{cases} \qquad \dots 17$$

$$\frac{\partial E_c}{\partial S_{cij}} = \begin{cases} \emptyset_{cj}(X_{ci} - C^2_{cij}) \dfrac{W_{cj}\, S\, ec}{S^3_{cij}} & if\ t_{ij} = 1 \\ 0 & if\ t_{ij} = 0 \end{cases}$$

In practice, the simple form of the PID algorithm, which depends on error, its derivative, and its integral, is not satisfactory and has to be modified to overcome its limitations [17]. The required modifications are: *Bumpless transfer*, and *saturation (integral windup).*

A. Bumpless Transfer

In the steady state with zero error, the controlled variable is zero. In many practical applications this is not suitable e.g. the case of temperature control system which requires a non zero voltage to be applied to the heater input to provide heat output. The availability of the integral term may lead to avoidance of the zero value for the controller output, but there will be difficulties in changing smoothly, without disturbance to the plant, from manual to automatic control. There will also be the danger that a large change will occur on changeover. Plant operating requirements demanded that manual/automatic changeover is done in the so-called "bumpless" manner [17]. Bumpless transfer can be satisfied by several methods. One of the most widely used methods is the velocity algorithm, which gives the change in the value of the manipulated variable Δm for each sample rather than the absolute value of the variable m. The difference equation for Δm is derived to be:

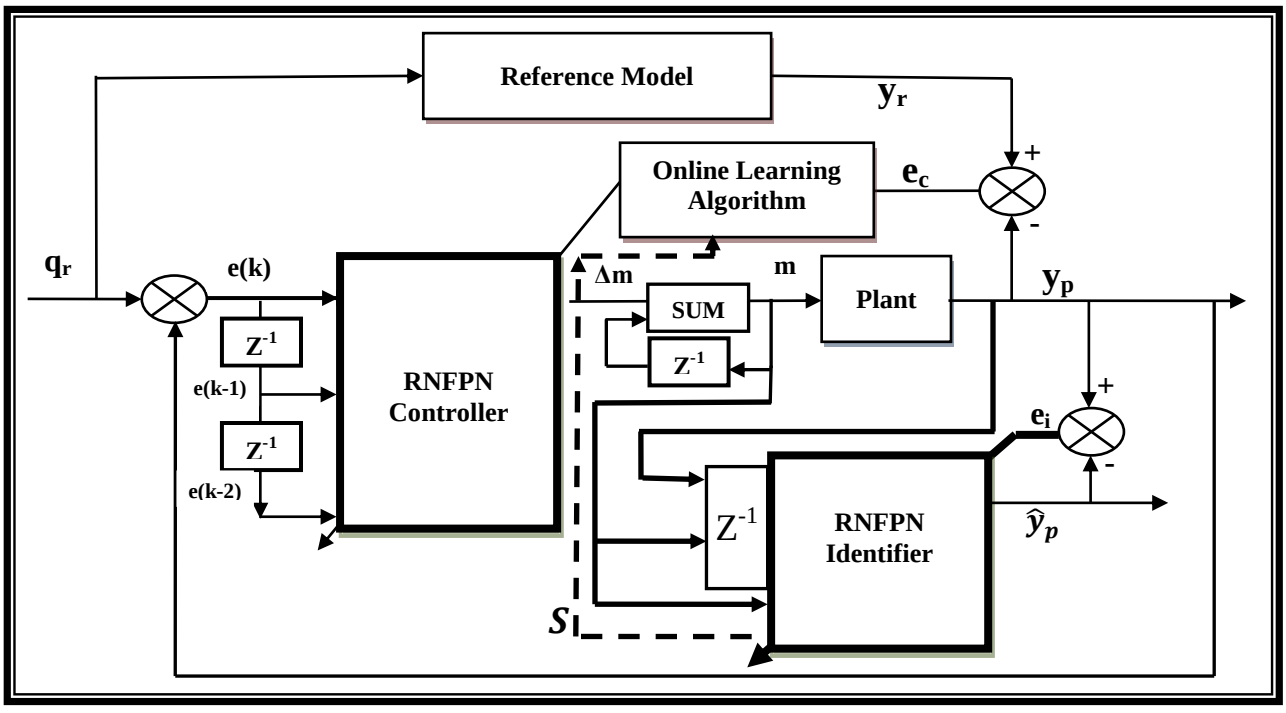

Figure 7: The proposed RNFPN forward (online) control

$$\Delta \boldsymbol{m}(\boldsymbol{k}) = \boldsymbol{k1} * \boldsymbol{e}(\boldsymbol{k}) + \boldsymbol{k2} * \boldsymbol{e}(\boldsymbol{k-1}) + \boldsymbol{k3} * \boldsymbol{e}(\boldsymbol{k-2})$$

Where $\boldsymbol{e}(\boldsymbol{k}) = \boldsymbol{q_r}(\boldsymbol{k}) - \boldsymbol{y_p}(\boldsymbol{k})$ is the error signal, $\boldsymbol{q_r}(\boldsymbol{k})$ is the set commands, and $\boldsymbol{k1}, \boldsymbol{k2}, \boldsymbol{k3}$ are constants to be tuned by the RNFPN controller. Hence, the inputs to the RNFPN controller are $\boldsymbol{e}(\boldsymbol{k}), \boldsymbol{e}(\boldsymbol{k-1}), \boldsymbol{e}(\boldsymbol{k-2})$ as shown in Figure 7. Since it outputs only the change in the controller position, this

algorithm automatically satisfies "bumpless transfer". Also, it is easy to program and safer in that large changes will not occur.

B. Saturation (Integral Windup)

The second major reason for using the velocity algorithm in our control system design is to overcome the problem of integral windup. In any practical application the value of the manipulated variable *m* is limited by physical constraints. An electric heater can supply only a given maximum heat and cannot supply a negative heat. If the value of the manipulated variable exceeds the maximum output of the actuator, effective feedback control is lost. Under some conditions, a large standing error in temperature will exist. If a PID controller is used, the integral term will continue to grow because there is a standing error. This effect is called integral windup which may lead to a poor response. The integral windup can be avoided by using the velocity algorithm since the integral action is obtained by a summation of the increments in the controller output (plant input). Therefore, there is an inherent integral limit which prevents a buildup of error leading to avoiding the saturation problem.

IX. SIMULATION RESULTS FOR THE RNFPN

A. Offline Identification

The software written for the simulation results of the offline identification is an m-file. As mentioned before the results of the offline is taken to be the initial values for the online identification and control. The initial values will speed up the training process of the online control very much. The offline itself needs an initial values for the weights which are taken to be 0.002 and the center and width of the Gaussian membership functions which can be calculated from the following two equations respectively [18]:

$$\boldsymbol{c_{ij}} = \boldsymbol{X_{n\,max}} - (\boldsymbol{i-1})\frac{\boldsymbol{X_{n\,max}} - \boldsymbol{X_{n\,min}}}{\boldsymbol{ni-1}} \quad \ldots \mathbf{19}$$

$$\boldsymbol{s_{ij}} = \mathbf{2}\frac{\boldsymbol{X_{n\,max}} - \boldsymbol{X_{n\,min}}}{\boldsymbol{ni-1}}$$

Where $X_{n\ max}$, $X_{n\ min}$ are the pre-specified maximum and minimum values for the universe of discourse for each input to the controller network; *ni is* the number of membership functions for each input variable. The model to be identified is a type 0 system which represents a small water bath temperature control system. A velocity PID controller is designed to generate the appropriate voltage inputs *u(k)* to the plant and the identifier as shown in Figure 6.

B. Online Identification and Control

The online forward control of Figure 7 is implemented by another m-file. The forward

identifier, which is used to estimate the plant sensitivity, will take its initial parameters from the final values of the offline identification system. All initial values for parameters such as: learning rates, number of rules, number of membership functions, constants of the dynamic threshold, etc. are chosen by trial and error for the online and offline identification. Figure 8 shows the trained response after few epochs while Figure 9 shows the result after 8000 epochs. The sum of square error (SSE) during the 8000 training epochs is drawn in Figure 10. It shows an average value for error of nearly 0.1, which is very good and acceptable in industrial applications. Of course, the value of SSE may be improved if the number of epochs is increased beyond 8000. In fact, we reach to 0.05 after about 16000 epochs then the SSE became fixed to this value. It is clear that these values (0.05 error in 16000 iterations) are much more better than the values obtained in [5] (3.1 error in 10^7 iterations). The value of error may change if some parameters are changed such as plant time constant, plant time delay, plant gain, learning rate, etc. If the values of weights and center and width of the membership functions become fixed during training i.e. there is no large change in their values then this is a good criterion for accuracy and convergence of the training process. Hence, Figures 11,12,13 show the norm for weight, center, and width. They show a very small change in their norm values, which means that the training process is in the right direction and convergence is reached. Now, the proposed controller is ready to be used in real time water bath temperature control system.

X. PRACTICAL IMPLEMENTATION

The schematic diagram of our setup is shown in Figure 14.

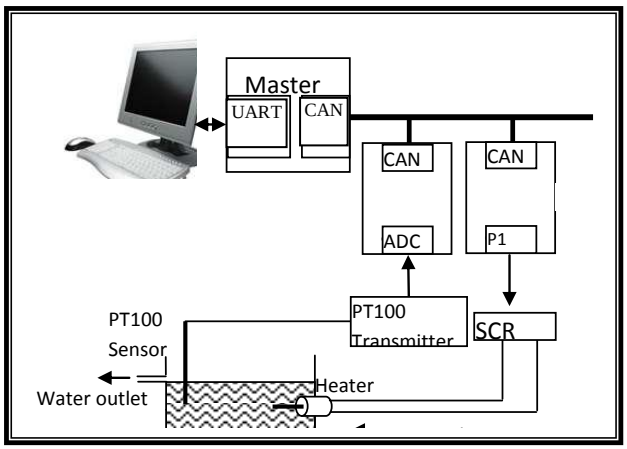

Figure 14: Water Bath Temperature Control System Based On Field Bus

One can use only one slave for inputting the sensor value and outputting the control signal to one of the ports of the microcontroller. This will reduce the cost and implementation effort. Note that the T89C51CC01 has an internal ADC components and some SFRs for doing that function.

The implementation process starts with the building of the master and slave controllers shown in Figure 15 and 16 respectively.

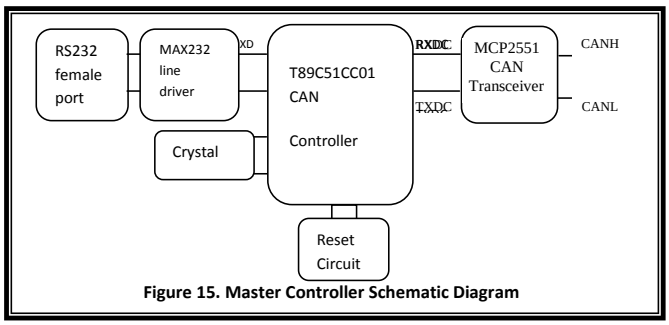

Figure 15. Master Controller Schematic Diagram

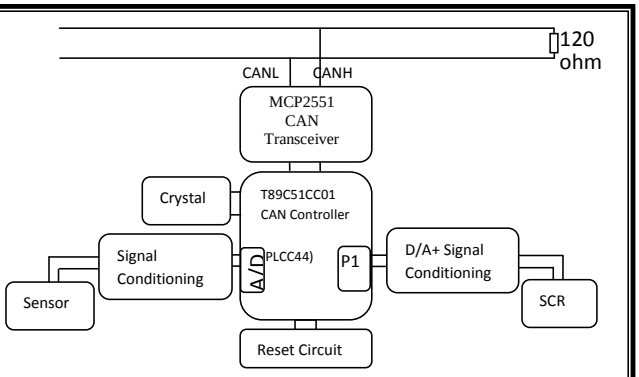

Figure 16. Slave Controller Schematic Diagram

The output signal from the sensing element (PT100) is fed to the A/D converter input of the CAN controller via a signal conditioning circuit used for converting the current signal to 0~3V signal suitable for the A/D converter. The set point generation is done in the software driver (control program). The error signal is then transferred to the RNFPN controller and an output control signal is generated in accordance and sent via the master through the CAN bus to the slave. The signal is received by the CAN controller and output it to port 1. Then the data on port 1 is converted to analog by DAC0808 converter [19] then to the signal conditioning circuit that is responsible for producing a 4-20 mA actuation signal that should be fed to the SCR. This control signal is used to control the phase angle of the SCR leading to a variable incident driving voltage to the heater and hence a temperature control is satisfied. The max power of the heater used in this setup is 2000 W. The obtained response is shown in Figure 17 which proves the performance of the implemented setup.

We clearly observe from Figure 17 that the steady state is achieved after approximately 2 minutes. This value is normal and accepted in such temperature control systems since these systems have a large inertia and time delay.

XI. CONCLUSIONS AND FUTURE WORK

In this paper, a general design methodology of a DCS system based on field Bus network protocol with CAN is presented. The design proved that the CAN protocol is a high reliable protocol and therefore can be adopted and highly recommended in industrial applications. As a case study, a temperature control system is designed and implemented. Because the temperature control system has characteristics of large time delay and large inertia, we need a high speed communication system to satisfy the real time requirements of the system. The overall system is implemented and controlled with a newly designed RNFPN controller designed with a velocity algorithm to satisfy the bumbless transfer and to avoid integral windup that may occur in real time. The response curve obtained is acceptable. There may be a development to this work which is the use of wireless sensor network (WSN) for data collection and monitoring. This may reduce the congestion on the field bus network i.e. the field bus will be used only for actuation leading to what is called wireless sensor/actuator networks (WSAN). WSAN may

be considered as a first gradual step towards building a wireless control system.

XII. REFERENCES

[1] Web site: www.CERN.org.

[2] F. M. Behno, "Design and Implementation of an Integrated Control System Using Field Bus (Control Center)", MSc Thesis, University of Technology, 2002.

[3] M.N. Hanna, "Real-Time Analysis of FIP-based Systems", MSc Thesis, Cairo University, 2004.

[4] Abraham, A., "Optimization of Evolutionary Neural Networks Using Hybrid Learning Algorithms", IEEE, Joint International Conference on Neural Networks, IEEE Press, Vol. 3, pp. 2797-2802, 2002.

[5] Chia F. J, Jung S.C., "A Recurrent Neural Fuzzy Network Controller For A Temperature Control System", IEEE International Conference on Fuzzy Systems, pp. 408-413, 2003.

[6] Tanomaru J., Omatu S., "Process Control by online Trained Neural Controllers", IEEE Transaction on Industrial Electronics, Vol. 39, No.6, pp. 511-521, Dec. 1992.

[7] Lin C.T., Juang C.F., and Li C.P., "Temperature Control With a Neural Fuzzy Inference Network", IEEE Trans. Syst., Man, and Cyber., - Part C: Applications And Reviews, Vol.29, No.3, pp. 440-451, Aug., 1999.

[8] Zhang J. and Morris A.J., "Recurrent Neuro-Fuzzy Networks For non-linear Process Modeling", IEEE Trans. Neural Networks, Vol.10, No.2, pp. 313-326, 1999.

[9] Mastorocostas P.A., and Theocharis J.B., "A Recurrent Fuzzy-Neural Model For Dynamic System Identification", IEEE Trans. Syst., Man, and Cyber., - Part B: Cybernetics, Vol.32, No.2, pp. 176-190, 2002.

[10] J. L. Peterson, "Petri Nets", ACM Computing Surveys,Vol.9, No. 3,pp. 221-252, 1977.

[11] L.M. Pinho and F. Vasques," Reliable Real-time Communication in CAN Networks", IEEE Transaction on Computers, Vol. 52, No.12, December 2003.

[12] "Enhanced 8-bit Microcontroller with CAN Controller and Flash memory (T89C51CC01), Atmel, www.atmel.com.

[13] "High Speed CAN Transceiver MCP2551", Microchip Technology Inc., 2003.

[14] Mohammed A.B. et al., "Design and Analysis of PI-Fuzzy Controller For Temperature Control System", Fourth Asia International Conference on Mathematical. Analytical Modeling and Computer Simulation, pp. 378-383, 2010.

[15] Rong-Jong Wai, Chia-Ming Liu," Design of Dynamic Petri Recurrent Fuzzy Neural Network and Its Application to Path-Tracking Control of Nonholonomic Mobile Robot", IEEE transactions on Ind. Elec., Vol. 56, NO. 7, pp.2667-2683, July 2009.

[16] Kumpati S. Narendra and Kannan Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol. 1, No. 1, March 1990.

[17] S. Bennett, "Real-Time Computer Control: An Introduction", Prentice Hall International (UK) Ltd, First edition, 1988.

[18] Rong-Jong Wai, Chia-Chin Chu," Robust Petri Fuzzy-Neural-Network Control for Linear Induction Motor Drive", IEEE trans. on Ind. Elect. , Vol. 54, No. 1, pp. 177-189, Feb. 2007.

[19] "DAC0808 8-bit D/A Converter", National Semiconductor Corporation, 2001.

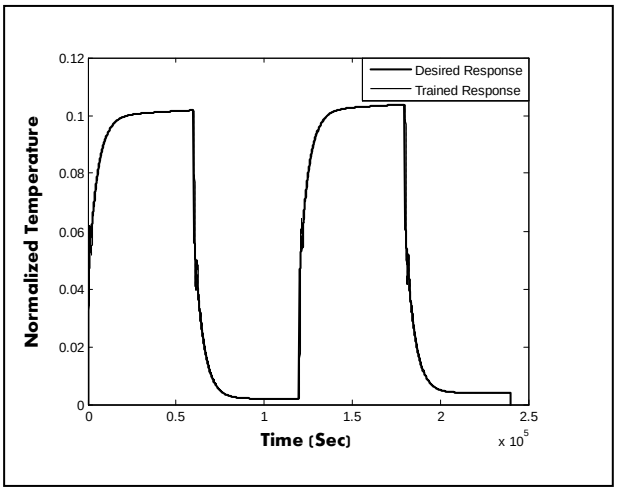

Figure 9: The Trained Response After 8000 Epochs For The Proposed Online Control System

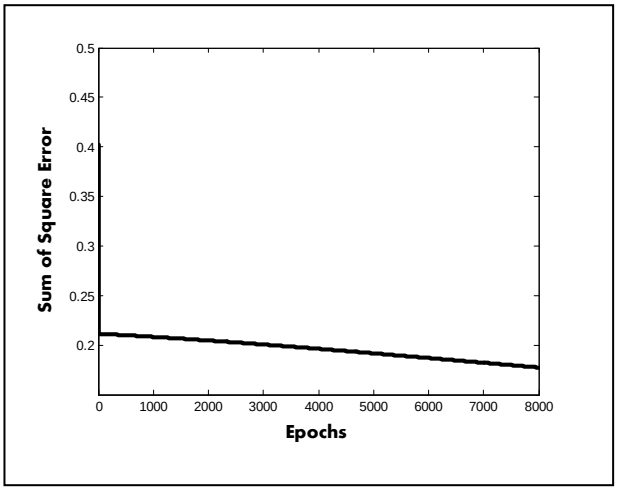

Figure 10: Sum of Square Error After 8000 Epochs For The Proposed Online Control System

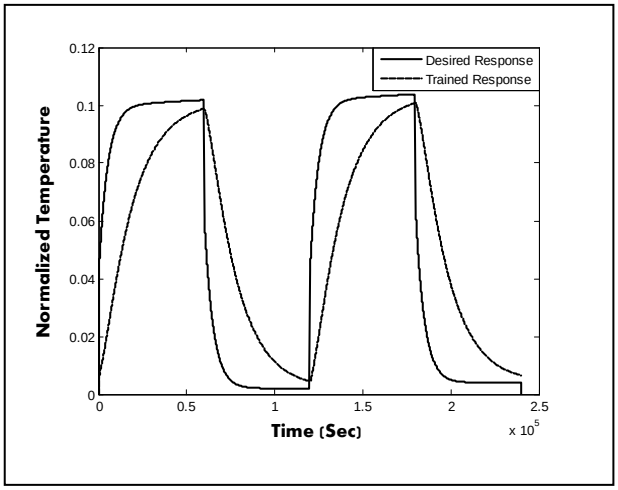

Figure 8: The Trained Response After Few Epochs For The Proposed Online Control System

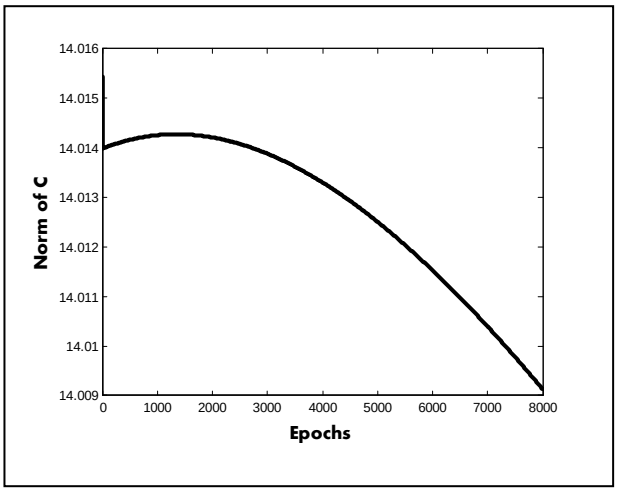

Figure 11: Norm of Center After 8000 Epochs For The Proposed Online Control System

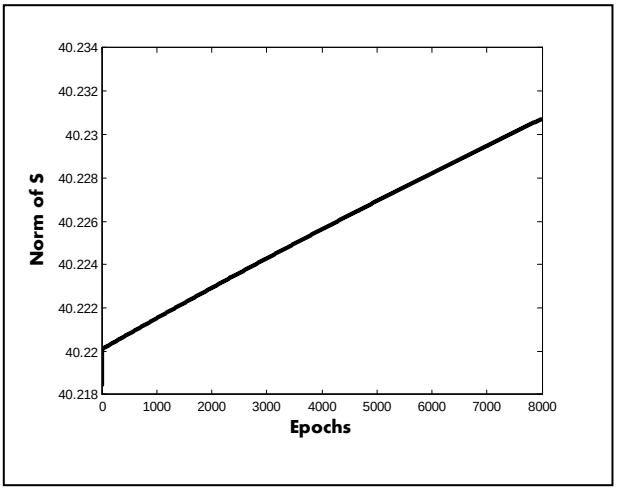

Figure 12: Norm of Width After 8000 Epochs For The Proposed Online Control System

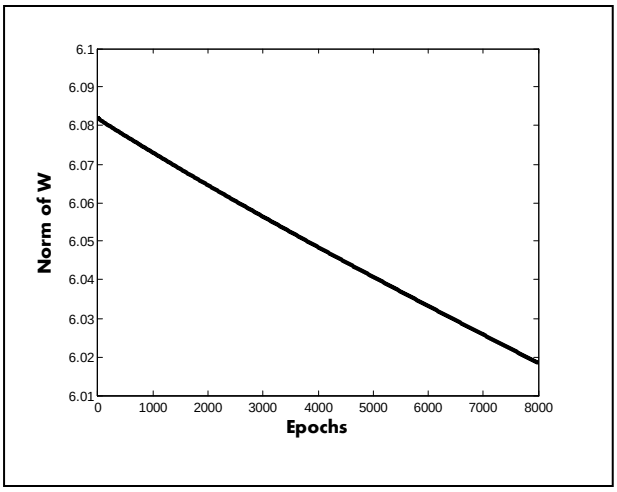

Figure 13: Norm of Weight After 8000 Epochs For The Proposed Online Control System

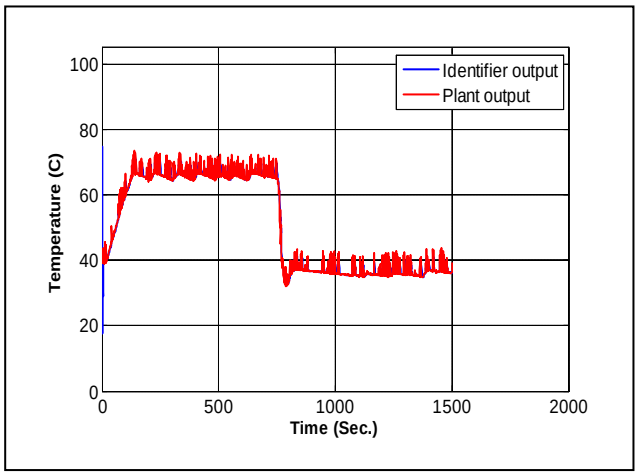

Figure 17. The measured response of the system

شبكة الناقل الحقلي المعتمدة على بروتوكول CAN مع مسيطر حرارة مبني بخوارزمية Fuzzy-Neural-Petri Net

علي احمد عبد[1]، عبد العظيم عبد الكريم علي [2]، علي فاضل مرهون[3]، نومان اسلام[4]

[1،2،3] كلية الهندسة-جامعة البصرة-العراق

[4]كلية الهندسة-جامعة دلهاوزي-كندا

الخلاصة

في شبكات الناقل الحقلي فان هناك مجموعة من الحساسات الذكية المربوطة مع بعضها وظيفياً والتي تتطلب مبادلة البيانات بشكل دقيق وضمن زمن محدد. في هذا البحث قدمت طريقة تنفيذ لنظام سيطرة موزع حقيقي لمنظومة حرارة. البرامج المضمنة للمسيطرات المكونة للشبكة تم كتابتها بلغة C. البروتوكول المعتمد للشبكة هو بروتوكول CAN ذات المواصفات العالية. المسيطر الرقمي الذي تم بناءه لمنظومة السيطرة يعتمد على مبدأ المنطق المضبب المعزز بميكانيكية الشبكات العصبية والتي تم اضافة طبقة جديدة لها هي طبقة الـ Petri. تم بناء هذا المسيطر باسلوب السيطرة الامامي. ايضاً تم تعليم المسيطر اولاً كـ offline للحصول على بيانات يتم التدريب عليها بالنسبة للـ online controller. المنظومة بنيت عملياً وطبقت على منظومة حرارة (حمام ماء) صناعية لاثبات صحة العمل والنتائج.