

# Design of a Neural – PLC controller FOR Industrial Plant

Eng.Wael A.Monsef  
Majma'ah College of Technology ,General  
Organization for Technical Education and  
Vocational Training (GOTEVT) ,KSA

Prof. Dr. Fayez F.G. Areed  
Computer Science Department  
Faculty of Engineering, Mansoura University  
Egypt

**Abstract:** *The present paper introduces a new algorithm for industrial processes by using neural network (N.N). The suggested method aims to make a combination between PLC and N.N. PLC (programmable logic control) is a good method for controlling any industrial processing but it did not have full picture about this process in any time. Neural network can made a survey on the process at all time. Learning N.N can expect the next instruction of the industrial operation before applying the input signal controlling. This paper introduces the new suggested technology presenting the architecture of the control system, and its components. A practical system will be constructed to satisfy the suggested technology.*

**Key words:** Neural network, PLC, Learning , Training.

## 1.0 Introduction

**Using** neural network in control can be classified into two phases :Neural network only as aid i.e help in control and Neural network as controller.The first phase [16][9] (neural network as aid) is possible if three roles are satisfied : The network assists or replaces a conventional model that is based either on a derived goal, such as minimum prediction error of the controlled variable, or directly on the control goal, such as minimization of a performance index,The network simplifies implementation of a control law by sorting certain controller parameters or by solving an implicit control law and The network performs diagnostics and recommends supervisory actions for a lower – level control system.The second phase is using neural network as controller[5], it is divided into two parts: The first term is "Train based on  $u$ " where control input signals  $u$  are available for training the neural controller. The second term is " Train based on goal" where the network devises the needed control strategy on its own , based on the ultimate control objective[7][8].

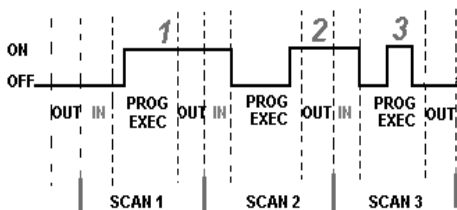
In the beginning of neural network, there were three controllers which have played important roles in research on neural control. The first one is Albus's cerebellar model articulation controller (CMAC) where in the control system , both the command and feedback signals are used as inputs to the CMAC controller . The output of the controller is fed directly to the plant . The desired output of the neural network controller has to be supplied. The training of the controllers is based on the error between the desired and actual controller output. [1][2]. The disadvantage of this scheme is that it requires a fixed controller to be designed for the plant. Miller update this scheme [13][14] , where the reference output block produce a desired output at each control cycle. The desired output is sent to the CMAC module which provides a signal to supplement the control from a fixed gain conventional error feedback control. At the end of each cycle, a training step is executed. The observed plant output during the previous control cycle is used as input to the CMAC module. The second is katwato et al's hierarchical neural network controller[12].Where the system has two identifiers, one for the identification of plant dynamics and the other for inverse plant dynamics.There is a main feedback loop, which the important in the training of the neural network.As training proceeds, the inverse dynamics part becomes the main controller.The final effects of Hierarchical neural network model based control are similar to feed forward control.The third is pasltis et al's multilayered neural network controller [18]. there are two kinds of control action: feed forward and conventional feedback control. A neural network implements the feed forward control part. The aim of training the feed forward part is minimize the error between the desired and actual plant output. This error is the input to the feedback controller. It was often assumed in the early years of neural network research that implementation in special hardware would be required to take advantage of their capabilities. Such hardware, in particular, would probably be *analog* and involve multiple parallel processing elements and connections between them. [4]Meanwhile, the development of hardware has been slow and with only modest commercial success. For a hardware implementation[6], specifications include the technology used (analog, digital, or hybrid), the precision (in numbers of bits) of the input/outputs, of the weights, and of the accumulators, etc. The most common performance rating is the Connection-Per-Sec (CPS), which is defined as the rate of multiplication and accumulate operations during recall processing. The Connection-Update-Per-Second (CUPS) value indicates the rate of weight changes during learning. Normalizing the CPS value by the number of weights on the chip (CPSPW, or CPS per weight) was suggested as a better way to indicate the processing power of the chip[ 11][22].Also, the Connection-Primitives-Per-Sec value, , includes the precision in the

processing performance[ 21]. The neural network can divide VLSI world into three broad categories:[15] digital, analog, and hybrids. Within these categories are included the various network and VLSI architectures, on-chip learning, etc

## 2.0 Statement of the problem

A PLC does not control a plant continuously but works by taking repeated “snapshots” of the plant input state. Working on this picture then updating the outputs as required. PLC had to respond to 3 things. *Input*-it took a certain amount of time for the PLC to notice the input signal from the plant. *Execution*- it took a certain amount of time to process the information received from the plant. *Output*-the plant receives a signal from the PLC unit and reaction.[20]. {TOTAL response time = *Input* response time + *Execution* response time+ *Output* response time }.

The PLC can only see an input turn on/off when it's looking. In other words, it only looks at its inputs during the check input status part of the scan. In the diagram(1), input 1 is not seen until scan 2. This is because when input 1 turned on, scan 1 had already finished looking at the inputs. Input 2 is not seen until scan 3. This is also because when the input turned on scan 2 had already finished looking at the inputs. Input 3 is never seen. This is because when scan 3 was looking at the inputs, signal 3 was not on yet. It turns off before scan 4 looks at the inputs. Therefore signal 3 is never seen by the PLC. To avoid this , the input should be on for at least **1 input delay time + one scan time as shown in figure (2)**. But what if it was not possible for the input to be on this long? Then the PLC doesn't see the input turn on. Therefore it becomes a paper weight! Not true... of course there must be a way to get around this. Actually there are 2 ways[17]. The first one shows in figure (3) is **Pulse stretch function**. This function extends the length of the input signal until the plc looks at the inputs during the next scan.( i.e. it stretches the duration of the pulse.). The second is shown in figure (4)is **Interrupt function**. This function interrupts the scan to process a special routine that you have written. i.e. As soon as the input turns on, regardless of where the scan currently is, the PLC immediately stops what its doing and executes an interrupt routine. (A routine can be thought of as a mini program outside of the main program.). After its done executing the interrupt routine, it goes back to the point it left off at and continues on with the normal scan process. Now let's consider the longest time for an output to actually turn on. Let's assume that when a switch turns on , a load connected is needed to turn on to the PLC output. The diagram (5) shows the longest delay (*worst case because the input is not seen until scan 2*) for the output to turn on after the input has turned on. The maximum delay is thus **2 scan cycles - 1 input delay time**. A further uncertainty in timing will be added as the remote I/O scan. As a result there is always an uncertainty , shown on figure(6) in the response time of the PLC between one program scan (minimum) and two program scans plus two remote I/O scans(maximum). Most PLCs start at the first instruction[19], and work in strict sequence to the end. During the program was run, PLC cannot thought about change it, PLC does not have any information from one task started until it finished.



Figure(1): Scan Input /Output signals

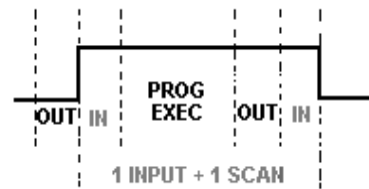


Figure (2): The input is 1 input delay time + one scan time

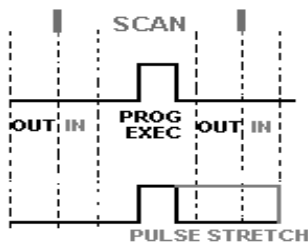


Figure (3): Pulse stretch function

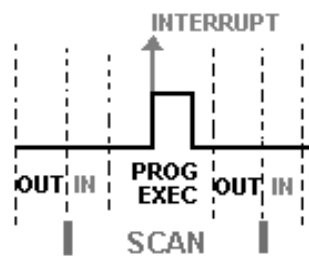


Figure (4): Interrupt function

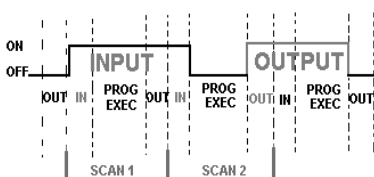


Figure (5): The maximum delay is 2 scan cycles - 1 input delay time

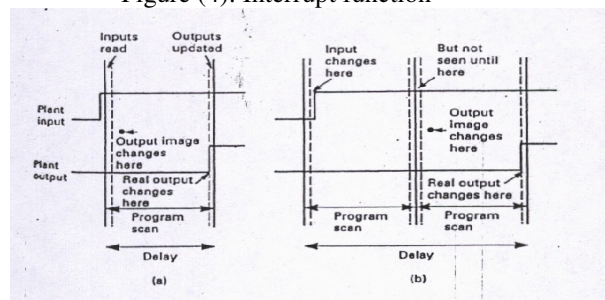


Figure (6) : The effect of program scan on response time  
(a) : Best case (b) Worst case

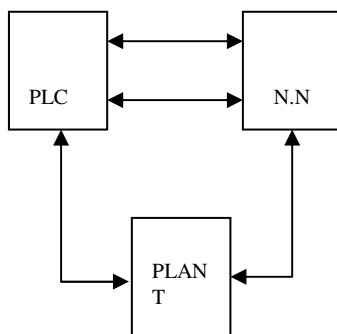
### 3.0 NN-PLC Architecture and hardware

The basic idea of mixing between neural network and PLC can be described in the block diagram shown in figure (7), where PLC measuring, operating and controlling the plant. The all data of this operation can be transferred to neural network to check and supervisor the plant behavior. Both of neural network and PLC has the same inputs (sensors, active switch, detectors,....) and the same output (actuators, air cylinders,....). Suppose a system has a group of sensors and motors to make a certain task. The block diagram of using neural network can be as in figure (8). The input signals come from sensors or another measurements elements and feed to the training neural network, then the output of the network will feed to motor or actuator. To training neural network, it connects with PLC unit where it runs the system. The same output may make a conflict, so the actual output is the output of PLC, and this output can be back propagated through the neural network. Two steps must be done, the first step is learning the neural network the behavior of the system and obtain the weights parameters of neural network then freezing it. The second step is removing backpropagation of neural network then use it in control the system or supervisor it.

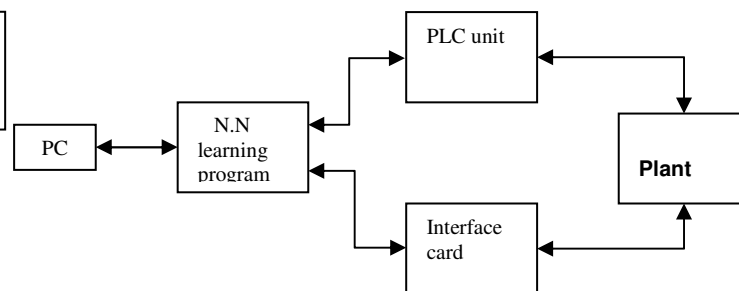
To improve this suggested method, it must be applied on an industrial process. After all theoretical basic are finished, a practical process will be carried-out to test the concept. The connection between PLC, N.N and plant can be described in figure (9). The aim of this suggested method is to satisfy that: Neural network learning the industrial process and check it every Time, when PLC program is running, Neural network can work a controller for some tasks inside the Process and Neural network may separate between PLC and the process and run the process by its learning knowledge when necessary.

#### 3.1 The interface card

The interface card can be designed and built its component which including two major parts: the first is an electronic board to receive the signal from plant component: pressure sensor, level sensor, temperature sensor and flow sensor. The second part is A/D card to send all this signal to N.N software. The block diagram of interface card can be described in figure (10). The component of interface card is: processor unit (Intel 8052, IBM 1987) where it connected to 8 channel digital I/O, feed signal to A/D converter to 8 channel analog input, 2 channel analog output and use TXD, RXD to feed serial port communication into computer to deal with the neural network software. Power supply where the input is 16-0-16 AC volt to feed +12, 0, -12 dc volt to the component of the card. To have analog output, op amp must be used. The communication was designed to use Rs-232 serial port to be compatible with PLC unit and computer.



Figure(7): Basic block diagram



Figure(9): Hardware structure

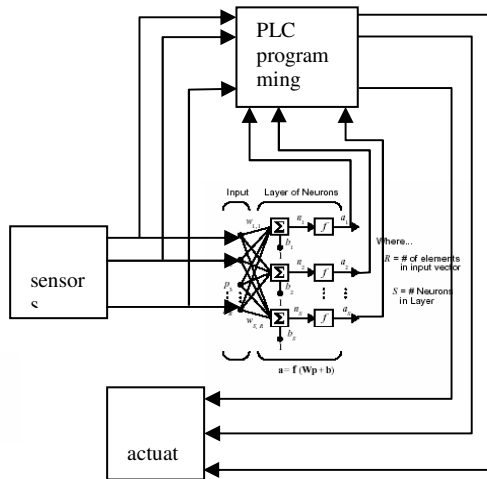
#### 3.2 Neural network software:

The suitable software is neural solution software designed by neurodimension inc.[10][26] the characteristic of this software is:

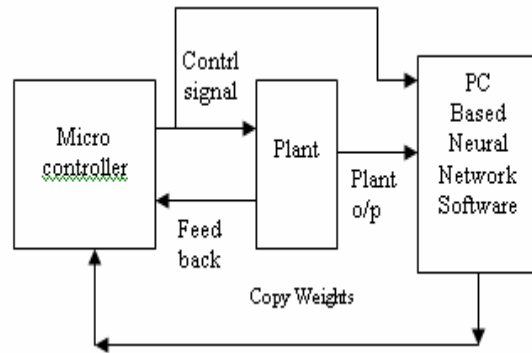
1-Interface NeuroSolutions to the External World: NeuroSolutions, like all other commercial neural network software is file driven. Data is read from files and results are written to files. NeuroSolutions provides a rudimentary windows-based interface through Object Linking and Embedding (OLE). This interface was improved and an OLE controller was written to act as in interface between the PC and controller.

2-Remote Control of NeuroSolutions: The above mentioned OLE component not only passes data to NeuroSolutions, but also acts as a "remote control" for NeuroSolutions. The OLE component actually opens the appropriate neural network in NeuroSolutions, sends data to it, and tells it to train. When the training is done, the OLE component asks for the trained weights and sends these weights back to the controller. This allows NeuroSolutions and the controller to remain in synchronization at all times. The main block diagram for connection between computer, software and plant can be drawing as shown in figure(11): The advantages of this arrangement are the following: The plant is always under control with a set of weights which are optimal or near optimal, Since the training is done separately from the control of the plant, the stability of the control can be maintained using appropriate checks before updating the weights of the controller. And finally This scheme can be easily integrated with the existing hardware.

In this research, Keyence is an industrial PLC unit[23][24][25] which was used in running the research applications and training neural network



Figure(8) : NN and PLC architecture



Figure(11) : Connection between software and plant

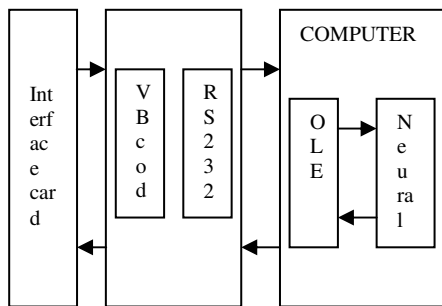
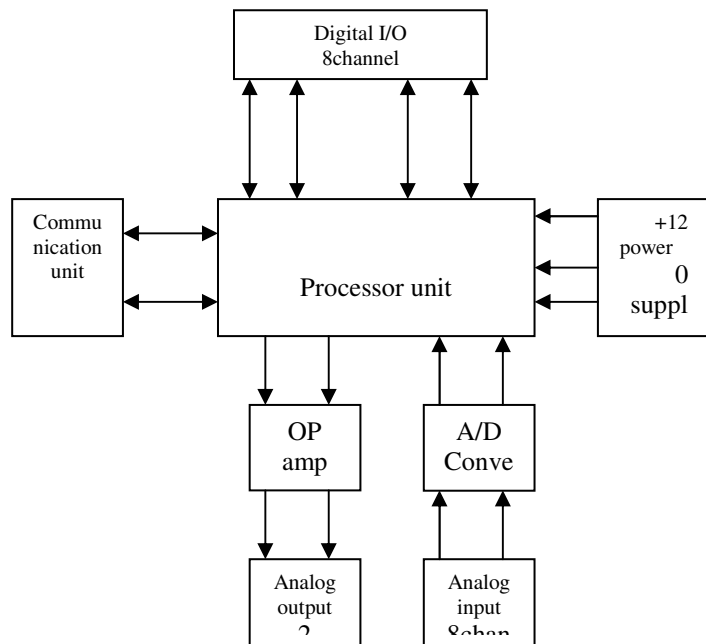


Figure (12) : The connection between interface card and computer



Figure(10) Block diagram of

### 3.3 Interface card – neural network connection:

NeuroSolutions software is a fully-compliant OLE Automation Server. This means that NeuroSolutions can receive control messages from OLE Automation Controllers, such as Visual Basic, Microsoft Excel, Microsoft Access, and Delphi. By writing a fully functioning Visual Basic program to set a network's parameters, run the network, and then retrieve the network's output. The interface card needs to send its signals to computer and receives output signal across RS232 serial port which must identify also .Figure (12) describe the connection and software relation between both of interface card and computer which include neural solution software.

The visual basic code must satisfy the next points:

- 1-Check for interface card response and serial port communication becomes OK. This step will repeated every time when the card receive or transmit signal. The "E" code for request attention. The code check the mscomm port if it is true ,the default port is port no 1 and it can be changed. if there is an error ,the message of fault will appear. The rule is : signal value AND mask for communication = result.
- 2-Send command for analog out (one commend for channel, the other for the value) by "A" code.
- 3-Get and receive command analog input by "B" code.
- 4-Send command for digital inputs/output by "C" code.
- 5-Get command for digital input/output by " D" code.
- 6-A code for open training neural network.
- 7-The form windows for this code is shown in figure(13).

## 4.0 NN PLC application

To design a neural network PLC (NNPLC) system, the research can be classified into two categories: mathematical design and practical design. *Mathematical* design is the important step in the design. There are three phases: The first is to Create some new rules, components and special networks to satisfy the function of PLC components for example: logic, counter, timer... The advantage of this step is reducing the hardware components of the electrical system and the operation will be depended on the neuro thinking. Self maintain circuit (start-stop circuit of motor) needs NO point in the contactor to hold the Power on, by using neural network. The second phase is N.N learns the process from actual PLC unit. The third step executes After training N.N, the parameter weights will be freezing, then N.N can control the plant of system by it self. The general architecture for N.N is shown in figure(14). According to the basic phenomena of N.N, the desired output can be calculated by the equations:

$$\text{Hidden neuron 1} = N_1 = W_{11}X_1 + W_{12}X_2 + \theta_1$$

$$\text{Hidden neuron 2} = N_2 = W_{21}X_1 + W_{22}X_2 + \theta_2$$

Where  $W_{11}, W_{12}, W_{21}, W_{22}$  are the weights of hidden 1 synapse,  $W_{31}, W_{32}$  are the weights of output synapse and  $\theta_1, \theta_2$  are the weights of the hidden axon and  $\theta_3$  is the weight of output axon.

$$\text{Output neuron} = f(\text{OUT}) = W_{31}f(N_1) + W_{32}f(N_2) + \theta_3$$

Where  $f(N_1), f(N_2), f(\text{OUT})$  are the transfer function of the neural network.

To have the correct output the  $\theta_1, \theta_2, \theta_3, W_{11}, W_{12}, W_{21}, W_{22}, W_{31}, W_{32}$  must be adjusted for every case.  $\theta_1, \theta_2, \theta_3$  value will be assumed zero value unless noted in the component. The N.N learning the behavior of logic from plc. By using interface card, N.N training to adjust its parameter, when the output becomes correct, creates N.NPLC symbols.

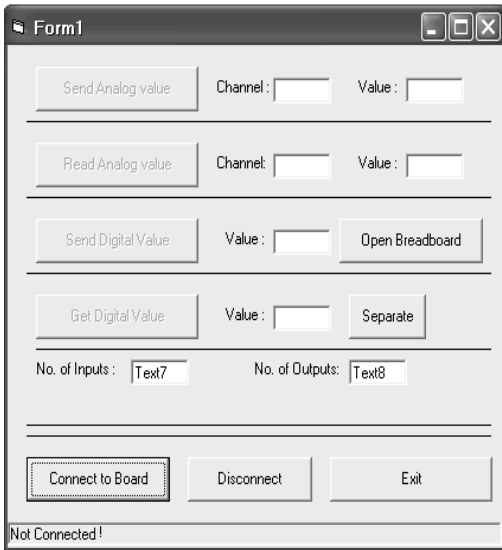


Figure (13): Form windows for VB code

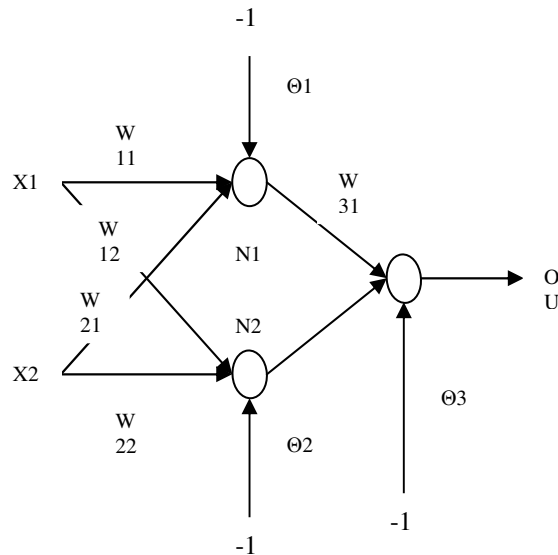


Figure (14): General architecture for NN

Where PLC expression refers to basic programmable logic control, N.N expression refers to basic neural network and N.NPLC expression refers to our created network.

Three of the most commonly transfer function (activity) can be applied to the network to activity its input.

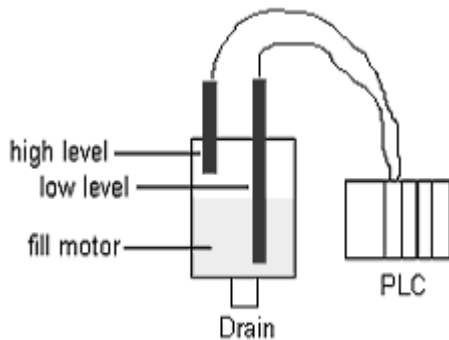
$$\text{Sigmoid function } f(x_i, w_i) \equiv \frac{1}{1 + \exp(-x)} \quad \& \quad \text{tanh function } f(x_i, w_i) = \tanh(x_i)$$

$$\text{Linear function } f(x_i, w_i) = \beta x_i + w_i$$

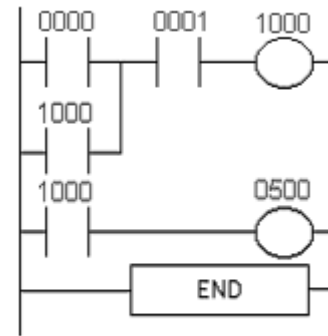
Note that  $\theta_1, \theta_2, \theta_3$  used as bias values, which added to the product of input and weight, the bias, has a constant input of 1. And  $W$  and  $\theta$  are both adjustable scalar parameters of the neuron; the central idea of neural networks is that such parameters can be adjusted so that the network *exhibits* some desired or *interesting* behavior. Thus, the network can be trained to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end. The second category is the Practical test which depended on: Successful connection between plant and NNPLCN architecture, The plant of the system that need to be controlled and The interface card between plant and NNPLC. In general cases, an input /output unit must be designed to provide the interface between the system and N.N software through N.N interface card. To put the aim of this search in a practical case, a control system will be design.

## 4.1 level application:

Now, let's process a program like PLCs do to enhance the understanding of how NNPLC can be designed, this example aims to prove the suggested method. Let's consider the level application where controlling lubricating oil being dispensed from a tank. This is possible by using two sensors. One near the bottom and one near the top, as shown in the picture (15). Here, the fill motor pumps lubricating oil into the tank until the high level sensor turns on. Figure(16) shows what the ladder diagram will actually look like. Notice that an internal utility relay was using in this example. Where it can use the contacts of these relays as many times as required. Here they are used twice to simulate a relay with 2 sets of contacts. These relays DO NOT physically exist in the plc but rather they are bits in a register that it can use to SIMULATE a relay. The signals from sensors read automatically and send to the N.N interface card. Notice the kind of sensor is a probe, which becomes open when no liquid found. In which the two inputs are normally open , the internal relay signal must be neglected and training N.N without its signal. The force of NNPLC is that it decrees the hardware and supervises of all the inputs and output signals every time. Therefore, the training signals are the statuses of input signals and the output signal which feeding to Neurosolution, figure (17) shows the weights of the system.



Figure(15): Dispensing oil from a tank



Figure(16) : Ladder diagram

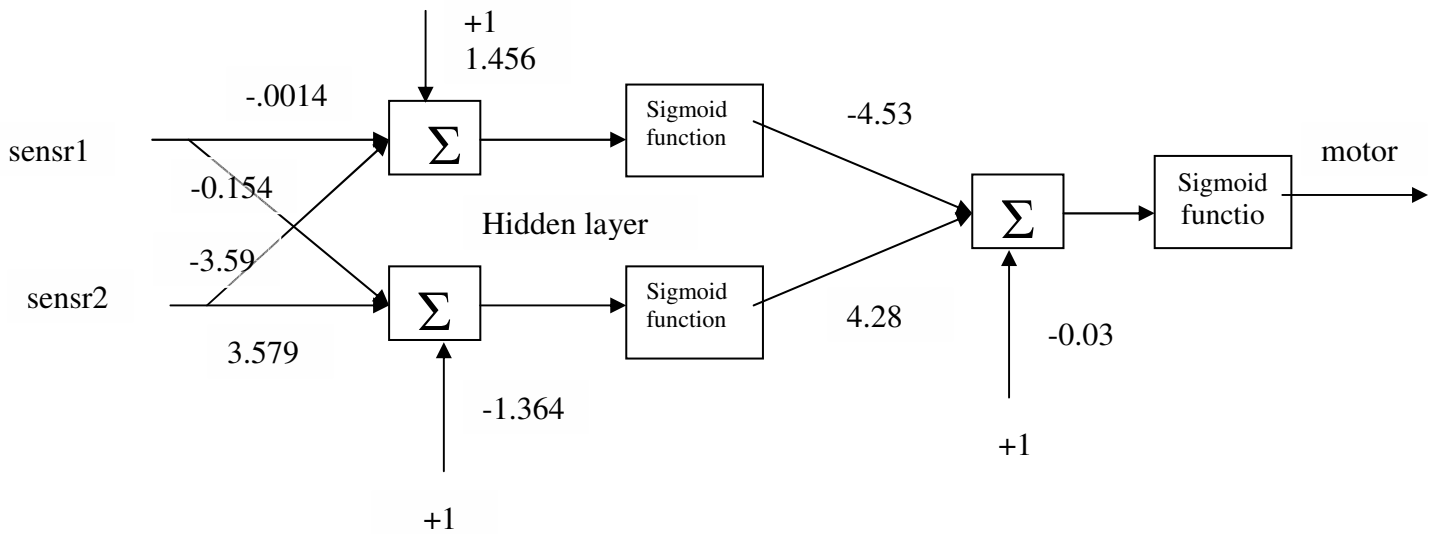


Figure (17) : The weights and bias of Level application after training from PLC program

## 5.0 Conclusion

**The** function of any control system is to automatically regulate the output and keep it at the desired value. The desired value is the input to the system. If the input is changed, the output must respond and change to the new set value. If something happens to disturb the output without a change to the input, the output must return to the correct value. In order to regulate this output, the error between output and input must be determined. This paper suggests a new method for controlling industrial processes by using neural network. It aims to: Create a new control system help to reduce error between desired and actual signals, React speedily with any change in the input and dealing with the disturbance in the system, Use neural network in control system. It wasn't used widely in the field of control technology. Neural network has the strong ability to supervisor and dealing with any data, Make a combination between neural network and

programmable logic controller. The function of this mixing is training neural network to adjust its parameter and solving the problem of delay time in PLC. Neural network learning the industrial process and check it every Time, when PLC program is running. Neural network can work as controller for some tasks inside the process. Neural network may separate between PLC and the process and run the process by its learning knowledge when necessary. To satisfy these points, the present paper has three phases: The first phase is Create mathematical design for suggested method. The second is establish the required hardware and software. And finally apply the suggested system in to practical systems.

## 6.0 References

- [1] Albus, J.S., 1975, "A new approach to manipulator control : cerebellar model articulation control (CMAC), Trans. ASME, J. of Dynamics syst., Meas. And Contr., 97, 220-227
- [2] Albus, J.S., 1979, " Mechanisms of planning and problem solving in the brain." Math. Biosci., 45, 247
- [3] Cesare Alippi, A. Ferrero, V. Piuri , 1998, " Artificial intelligence for instruments and measurement applications" , IEEE Instrumentation & Measurement Magazine , June 1998, 9-17.
- [4] Clark S. Lindsey, 2002, "Neural Networks in Hardware: Architectures, Products and Applications" Guest Scientist, Royal Institute of Technology, Stockholm, Sweden [www.particle.kth.se](http://www.particle.kth.se), August 26, 2002 - Update]
- [5] Duc, 1995, "Neural networks for identification, predication and control", Duc trung pham and Xing liv , university of wales cardiff, UK, Springer- verlag newyork , 1995
- [6] E. van Keulan, S. Colak, H. Withagen, H. Hegt, 1994, "Neural Network Hardware Performance Criteria", Proc. of the IEEE Conference on Neural Networks, III, 1885-1888,
- [7] I.F. Goyal, "Industrial application of neural networks", I.F. Goyal, J.P. Mason, Project ANNIE hand book , Springer – verlag press 1992
- [8] James, 1995, "An Introduction to Neural Network", James. A. Anderson , 2<sup>nd</sup> edition, Massachusetts Institute of Technology
- [9] John , 1991, "Introduction to the theory of neural computation", John Hertz, Anders Krogh , Richard G. Palmer, Addison – Wesley publishing company 1991
- [10] Jose C. Principe, Neil R., W. Curt, 2000, "Neural and adaptive systems : Fundamentals Through Simulations" [www.nd.com](http://www.nd.com)
- [11] Kasko, 2000 "Neural network and fuzzy systems", Bart Kasko, university of southern california, Prentice hall of India 2000
- [12] Kawato, M., Uno, Y., Isobe, 1988, "Hierarchical neural network model for voluntary movements with application to robotics" , IEEE Contr. Mag., 8-15
- [13] Kraft, L.K and Campagna D.P , 1989 "a Comparison of CMAC Neural network and traditional adaptive control systems, 1989, American control conference , Pittsburgh, PA, June 21-23
- [14] Mille III, W.T. Glanz, 1989, " Real time application of neural networks for sensor based control of robots with vision", IEEE Trans. Syst. Man, and Cybern., 19(4), 825-831
- [15] M. Holler, "VLSI Implementation of Learning and Memory Systems: A Review", Advances in Neural Information Processing Systems 3, Morgan Kaufmann, San Mateo, Ca. 1991.
- [16] Mukul, 1997, "Asystematic classification of neural network based control ", Mukul Agarwal , April 1997 , IEEE 3<sup>rd</sup> conference on control applications in Glasgow
- [17] Phil Metore, 2003 , "learn PLC" , [www.PLCS.net](http://www.PLCS.net)
- [18] Psaltis, D., Sideris, A. and Yamamura, 1988 " Amultilayered neural network controller" IEEE Control systems Magazine, 17-21
- [19] Robbert, 1988, "Soft logic A guide to using a pc as a programmable logic controller", Robbert Carrow, McGraw – hill 1988
- [20] Thomas, 2000, " Industrial electronics applications for programmable controllers , instrumentation and process control" , Thomas E. Kissell , Terra community college, by prentice hall , inc 2<sup>nd</sup> edition
- [21] T. Sejnowski and C. Rosenberg, "NetTalk: A Parallel Network that Reads Aloud", Tech. Rep. JHU/EECS-86/01, The Johns Hopkins Univ. Elect. Eng. and Comp. Sci. Dept. 1986
- [22] Valluru, 1995 , " C++ neural networks & fuzzy logic ", Valluru Rao & Hayagrivarao Mis press , 1995, 2<sup>nd</sup> edition
- [23] W. Bolton, 1999 " Programmable logic controller", W. Bolton, Plant Tree, 1999 , 4<sup>th</sup> edition.
- [24] [www.kyenece.com](http://www.kyenece.com)
- [25] [www.plcs.net](http://www.plcs.net)
- [26] [www.nd.com](http://www.nd.com)