Intelligent Nodes for Distributed Sensor Network

A. Sachenko, V. Kochan, V. Turchenko, V.Tymchyshyn, N.Vasylkiv

Institute of Computer Information Technologies
Ternopil Academy of National Economy
3 Peremoga Square, 282004 Ternopil UKRAINE
Phone: +380 (352) 33-0830 Fax: +380 (352) 33-0024
E-Mail: sachenko@cit.tane.ternopil.ua

Abstract

Neural networks models and their training algorithms on central computer with reference to previously developed distributed sensor network are considered. The requirements to its intelligent node are formulated. Also node's structure is offered which realise such intelligent functions, as sensor and other measuring channel components drift prediction using remote reprogramming.

1. Introduction

The quality management of modern industrial systems requires usage of intelligent instrumentation. Their sensors mostly have low accuracy because of technological restrictions and absence of built-in correction means. Usually the measuring modules work on non-changed algorithms, their output information is transferred by analog or raw digital signals. It requires large communication and computational power for distributed sensor systems and networks construction. The works [1, 2, 3] are devoted for these defects elimination. However the problem is still matter of current interest – the majority of produced measuring modules and systems [4] do not provide sensor error correction.

In [5] a hierarchical structure of intelligent distributed sensor network (IDSN) (measuring modules (IM), intelligent nodes (IN), central computer (CC)) is offered which realises some intelligent functions. Below is given essence of main IDSN component construction – intelligent node, which enable to use effectively a neural networks for measuring channels errors correction using remote reprogramming. There are two ways of reprogramming [6]:

1. Storing in node's non-volatile memory all previously debugged subroutines and supervising software of their choice (new constants values are loaded in RAM in particular). Loading the software in C, Forth, Basic languages into external microcontoller's RAM with its execution by program-interpreter [6] (the data loaded for program stored in ROM) also is variant of this;
2. Remote loading of new software (during node's functioning) into microcontroller's RAM (like IBM PC remote boot [7], etc.). It requires system loader presence in microcontroller's ROM [6] (for routine storing and their addresses modification before execution), network server support and the shared code and data memory addresses (von Nouman architecture).

The first way disadvantages are speed reduction by interpreter using, enlarging sizes routines library and its expansion difficulty during system exploitation. The second way disadvantages are routine debugging complexity because addresses updating and interruptions service time increasing. The specified defects strongly complicate using of both ways in IDSN. Therefore, the specialised IN development is expedient for measuring channels errors correction using neural networks.

2. Sensor error prediction using neural networks

For sensor and other IDSN components errors prediction single-layer and multi-layer perceptrons models [8] are used. In used models the current neuron condition is

$$s = \sum_{i=1}^{N} x_i \cdot w_i - T\,, \qquad (1)$$

where x_i – neuron synapses value,
w_i – synapses weight,
T – neuron threshold.
The neuron output is determined as function of its condition

$$y = f(s)\,. \qquad (2)$$

In single-layer perceptron model linear function is used

$$f(x) = K \cdot x, K = 1, \tag{3}$$

and in multi-layer perceptron model – non-linear sigmoid function

$$f(x) = \frac{1}{1+e^{-x}}. \tag{4}$$

The single-layer perceptron training was executed on following algorithm:

1. Initialise the weights and threshold to small random numbers;
2. Present to perceptron inputs the source pattern, and calculate its output, using equation (1), (2), (3);
3. If output is correct, go to step 4;
   Else calculate the difference between received and required output values
   $$\beta = y - d,$$
   where y – real (received) perceptron output,
   d – required output value.
   Update the perceptron inputs weight
   $$w_i(t+1) = w_i(t) - \alpha(t) \cdot \beta \cdot x_i,$$
   where: t and $t+1$ – current and next iterations numbers accordingly,
   i – number of perceptron input,
   $\alpha(t) = \dfrac{1}{1+\sum_i x_i^2}$ - adaptive learning rate [9].
4. Repeat cycle from step 2, while network error exceeds given value.

The training of multi-layer perceptron was executed on back propagation error algorithm [10]:

1. Initialise the weights and thresholds to small random numbers;
2. Present to network inputs the source pattern, and calculate last layer outputs, using equation (1), (2), (4);
3. If output is correct, go to step 4;
   Else calculate last layer neurons error
   $$\beta_j = (y_j - d_j) \cdot \frac{dy_j}{ds_j}, \tag{5}$$
   where y_j – real output condition of neuron j,
   d_j – required output condition of this neuron,
   $\dfrac{dy_j}{ds_j}$ – sigmoid function's derivative for neuron j.
   Update of the last layer neurons weights
   $$w_{ij}(t) = w_{ij}(t-1) - \alpha(t) \cdot \beta_j \cdot y_i, \tag{6}$$
   where β_j – error of j neuron of current layer,
   y_i – previous layer neuron output values.
   $$\alpha(t) = \frac{\sum_j \beta_j^2 \cdot y'_j}{\left(1+\sum_i y_i^2\right) \cdot \sum_j \beta_j^2 \cdot \left(y'_j\right)^2}, \tag{7}$$
   - adaptive learning rate [9].
   To calculate new weights for all previous layers using equations (5), (6), (7).
4. Repeat cycle from step 2, while the network error exceeds given value.

IDSN components errors prediction was executed by two methods:

- *The multi-step prediction* – on next prediction step is used not real, but predicted value, received by own neural network on the previous prediction step. The method is used at the long-term prediction for determination of trend and its characteristic points. The error analysis of multi-step prediction permits to evaluate increase rate of inter-calibration interval and its possible risk;
- *The one-step prediction* – on next prediction step is used real value instead predicted. The method is used for determination error prediction quality.

Analysis of neural network training and use has shown that they are incommensurable in computing resources. Typical training time of neural networks on PC Pentium-200 is up to several minutes and operational time (prediction) takes microseconds. For simple models of neural networks it allows to exclude approximation of individual sensor drift error function and to use previously trained on CC neural network in intelligent node. However it imposes the additional requirements to IN considered below.

3. Intelligent node's requirements and structure

The structure of offered IDSN hardware contains three levels of measuring information processing [5]:

- The first level (measuring modules) realises sensor signal transformation into a code, and, in most cases, simple calculations. The processing algorithms usually are realised by hardware or are recorded in the IN microcontroller's ROM;
- The second level (intelligent nodes) realises the majority of formulated in [5] intelligent functions concerning measuring process;
- The third level (central computer server) provides adaptation and self-training procedures for intelligent nodes and supports all IDSN elements functioning.

The offered IDSN structure features by passing to INs the functions of measurements results correction (continuous prediction of the correction and correction error for measuring channel elements, in particular, sensor, by individual simplified mathmodels, as well as measurements results correction). It considerably reduces a network traffic at the expense of exchange between network components by processed measurements results only (with pre-set errors). As the first and third levels elements devices considered in [5] should be used. IDSN advantages, provided by such functions division, are determined by IN opportunities. Therefore problems of its construction are considered below in more details.

The IDSN universality requirements assume a possibility to integrate heterogeneous sensors, measuring modules, etc. in a single network. Thus the intelligent node according to [5] must realise the following procedures:

1. The support of necessary number of input/output channels (requirements of their protocols are difficult to define beforehand);
2. Realisation of measuring information processing functions which correspond to individual parameters of used measurement channels (depending on measured physical quantity, sensor type, measuring module, interface);
3. Realisation of testing and calibration functions for used measuring channels and their components;
4. Sensor drift correction calculation and, if necessary, other measuring channel components;
5. Including correction into measurement results;
6. Calculation of prediction error for sensor drift correction and, if necessary, other measuring channel components;
7. Measurement results output according to users requests.

Expediently also is to provide an opportunity of direct connection to IN the nearest sensors and effectors and external devices control signals generation. Besides, the procedures in items 1..3 depend on variable structure of channel and its elements according to measuring task requirements. Procedures in items 4..6 depend on neural network models (calculating the corrections and errors) which in general are individual for each channel. Thus, in a simple case, the use of one model with individual parameters is possible, and in a complex case it is necessary to use the individual models. It is clear, that the requirements to intelligent node are rather contradictory and it is hard to define them before the IDSN design stage. Moreover, IDSN should have possibility to change 4..6 procedures at its normal functioning during adaptation and training processes. It can be supplied by remote reprogramming (reconfiguration) of IN software.

The optimal base for IN construction on executed functions volume/price ratio is single-chip (for example, AT89C51 type) [6]. Such microcomputer contains built-in serial interface, timers and interrupt request inputs, also it supports external and internal ROM and RAM modes, etc. However, addresses division on commands and data spaces (Harvard architecture) does not allow direct facilitating the two ways of remote reprogramming described in Introduction. Also the necessity of loaded code addresses and interrupt vectors modification complicates the debugging [6]. For elimination of these defects loading the IN work software into external RAM from zero address by system loader (located in internal microcomputer's ROM [11]) is offered. After software loading the microcomputer is switched to external memory operation mode and begins program execution. Thus any changes in the loaded software is not required. The structure of this software corresponds to [5] and contains three subsystems: compensation, supervisor and communicator (as the loader is not accessible in microcomputer's external memory operation mode).

The central element of IN in the general IDSN structure (Fig. 1.) is developed Base Controller. It interacts with server through the network interface (NI) and measuring (MB) and control (CB) expansion boards are connected through the system bus interface (SBI). Sensors and effectors are connected to MB and CB. Digital code producing measurement modules MM interact with IN through the input-output interface boards (IB). Basic input-output ports (BIOP) located on controller's board provide seven-segment or LED display and simple 16 button or 101-key IBM PC keyboard and printer connection to IN, as well as organise interaction with MMs without expansion IBs use. The offered IN structure is reasonably universal, makes possible to realise above listed procedures and can be used in wide range of IDSN applications.

4. IN realisation

The IN structure on single-chip 89C51 type with electrical reprogramming ROM (flash-memory) is shown on Fig. 2 [11]. On the IN board there is a low addresses register (AR), random access memory (RAM) with control circuit (CS), system decoder (DC), network interface adapter (NI), clock generator (Clk), interrupt circuit (IC), bus former (CD), microcomputer reset circuit (Res), as well as BIOP registers. The trigger (Tr) is added to the circuit for maintenance of remote routine reconfiguration. RAM chips up to 32 Kbytes can be used. The addresses range is equally divided between the code (begins with zero address) and data (other dividing is possible). CS permits passage of write signal (WR) on RAM at choice of any addresses during remote routine reconfiguration and forbids write in routine addresses range, as well as permits passage of commands choice signals (PSEN) and data reading (RD) during routine execution. Non-masked interrupt signal (NMI) arrives directly on the microcomputer (MCU) input (INT0) with maximum priority, the other interrupts are generated by IC on the input INT1.

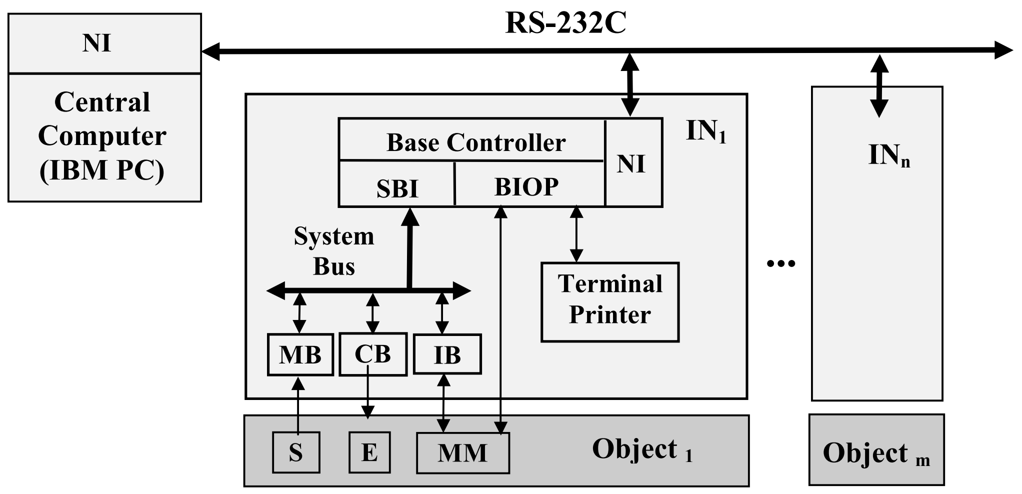

Fig.1. General IDSN structure

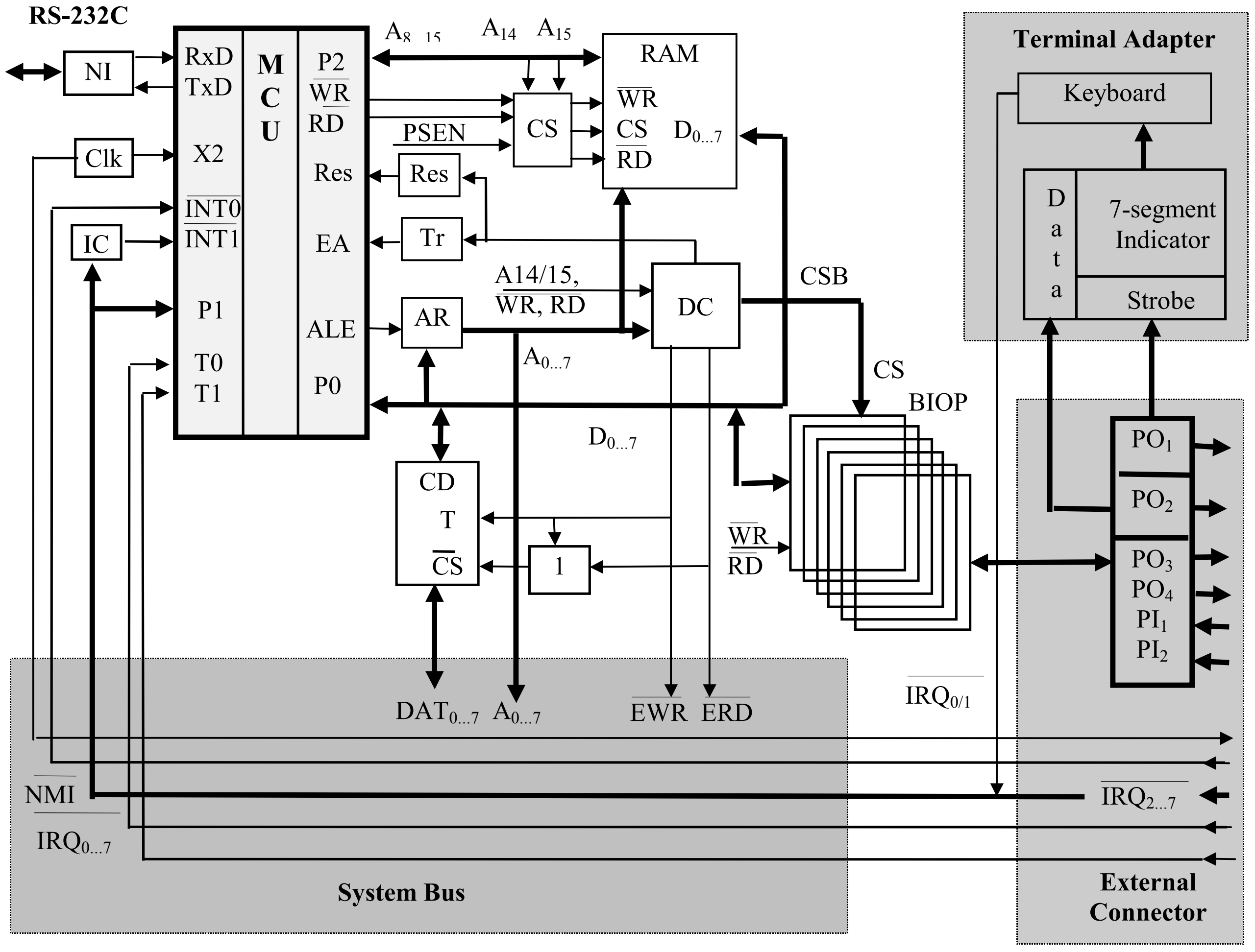

Fig.2. IN structure

P1 port is used for identification of interrupt source. Up to six BIOP registers are stipulated in IN and four of them are outputs and two of them are inputs. For connection of single expansion board set the SBI is realised containing data $DAT_{0...7}$ signals buffered by CD, low addresses $A_{0...7}$ and write EWR and read ERD signals, as well as clock signals, interrupt requests NMI and IRQ and timer inputs. The circuit NI is executed on transistors, optrone separation is stipulated. For expansion of use NI realises conventional three wires network (RS-232C standard), as well as offered in [12] two-wire, enabling IN to work in network up to1.5 km.

At using 61256 RAM in IN the microcomputer's addresses spaces (64 Kbytes) is divided as following:

- Code memory (write protected during execution) - 0000... 3FFF;
- Data memory - 4000... 7FFF;
- Expansion board addresses, connected to a system bus - 8000... 80FF;
- BIOP addresses - C0F1... C0D0;
- Addresses for switching to external (stored in RAM) program - C040;
- Addresses for switching to internal (stored in ROM) program - C080.

Such IN, having large flexibility in application, allows according to processor and memory loading, to provide various volume of correction functions of measurements results (for example, in one of the most difficult cases - combination of calibration with prediction [13]). Thus the IN supervising routine downloads required functions by reading from CC and stores into RAM necessary subroutines. Inparticularly the stage of network training using neural networks requires large computing and timing resources. At the same time the errors prediction stage of sensors and other IDSN components does not require significant resources, because it is reduced to calculation of prediction source data results.

5. Sensor error prediction results

Described above neural networks models were used for prediction of distributed sensor network components errors. For neural network training mathmodels of concrete sensors types and other components were used. It made possible to separate from particular devices in certain measuring channels. For example, function

$$y = a\sin(bx + c) + d\sin(ex + f) + g \qquad (8)$$

modelling measuring channel errors without sensor, which include such IDSN components errors as switchboards, ADC, etc. For these errors prediction the multi-layer perceptron model is used (according to (1), (2), (4)), which consists of five input and four hidden and one output neurons. The training sample consists of 50 values with step 0.1. Prediction was executed on a interval {0 ..5}. For 536 thousand training iterations (37 minutes of training on PC Pentium-200) sum-square error $1\times10E^{-4}$ was achieved, that has allowed to receive maximum error 0.56% on the one-step prediction and 11.25% on the multi-step prediction (Fig.3).

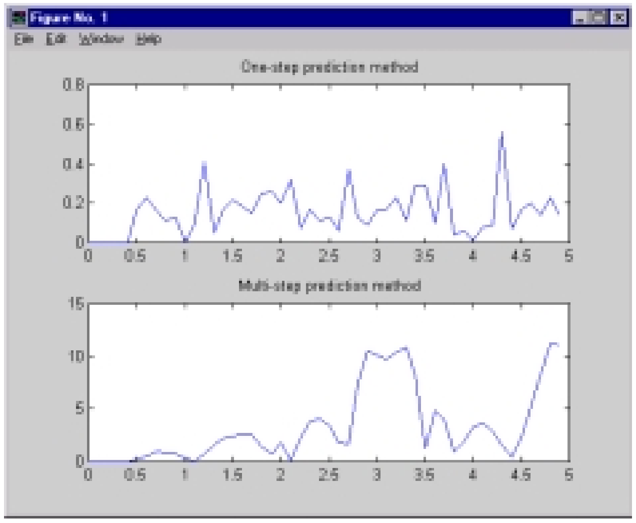

Fig 3. Predicted percentable error for function (8)

Other function

$$y = ax^2 + bx + c\sin(x) \qquad (9)$$

modelling industrial thermocouple drift [14] according with calibration and testing errors. For drift prediction was used six-input single-layer perceptron model according to (1), (2), (3). The training sample consists from 100 values with step 0.4. Prediction was executed on a interval {40 ..80}. For 3.6 million training iterations (44 minutes of training on PC Pentium-200) sum-square error $1\times10E^{-8}$ was achieved, that has allowed to receive maximum error percent 0.000004% on the one-step prediction and 0.012% on the multi-step prediction (Fig.4).

6. Conclusion

Testing of developed IN in complete set with precision switchboard and low voltage sensitive ADC [15] has shown, that the error briefly and intermediate term prediction correction on industrial thermocouples drift does not exceed specified in [5] 0.3 .0.5°C. Developed IN can be successfully used in IDSN, which require high measurement accuracy at frequent change of its structure, executed tasks and operational conditions of sensors.

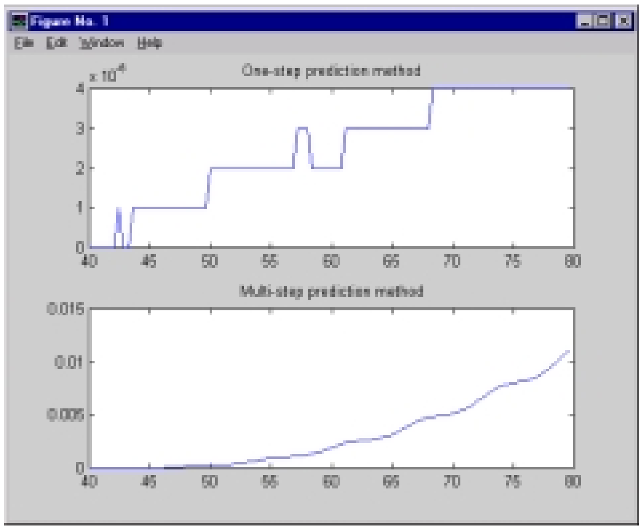

Fig 4. Predicted percentable error for function (9)

Acknowledgement

Authors would like to thank INTAS, grant reference number INTAS-OPEN-97-0606.

References

[1] E. J. Brignell, "Digital compensation of sensors", *Scientific Instruments*, vol. 20, No 9, 1987, pp.1097-1102.

[2] L. Finkelstein, "Measurement and instrumentation centre", *Measurement,* vol. 14, No 1, 1994, pp.23-29.

[3] A. Sachenko, "Application of the simulation for accuracy improvement in the measurements of non-electrical quantities", *First International Conference "Modelling in Measurement Processes"*, Wroclaw, Poland, 1993, pp. 148-151.

[4] Fluke – Philips, *Test and Measurement Catalogue,* 1997.

[5] A. Sachenko, V. Kochan, V. Turchenko, "Intelligent distributed sensor network", *IEEE Instrumentation and Measurement Technology Conference,* St. Paul, USA, vol.1., 1998, pp. 60-66.

[6] URL http://www.atmel.com/atmel/products/prod20.htm

[7] W. Feibel, Novell's Complete Encyclopedia of Networking, Novell Press, Sybex, 1995.

[8] A.K. Jain, J. Mao, K.M. Mohiuddin, "Artificial neural networks: a tutorial", *Computer*, March 1996, pp. 31-44.

[9] V. Golovko, V. Dimakov, V. Gladishchuk, J. Savitsky, "A neural system for intelligent robot navigation", *International Conference on Technical informatic, Proceedings Automation and Industrial Informatics*, Timisoara, Romania, vol.1, 1996, pp. 63-70.

[10] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing*, vol. 1, Cambridge, MA: MIT Press, 1986, pp. 318-362.

[11] V.Kochan, V.Tymchyshyn, "Controller with remote reconfiguration", *Journal of Ternopil State Technical University*, Ternopil, Ukraine, vol. 3, No 3, 1998, pp. 82-88.

[12] V.Kochan, V.Tymchyshyn, "Construction of distributed information measurement systems on the basis of modified RS-232C interface", *Proc. of the 10th IMEKO TC-4 Symp. On Development in Digital Measuring Instrumentation and 3rd Workshop on ADC Modelling and Testing*, Naples, Italy, 1998, pp. 723-726.

[13] A. Sachenko, B. Maslyjak, V. Kochan, "Design of measurement instruments on the basis of transducers with self-calibration", *Proceedings of the Third IFAC-Symposium on the Low Cost Automation*, Vienna, Austria, 1992, pp. 139-141.

[14] A.Sachenko, V. Milchenko, V. Kochan, M. Chyrka, A. Karachka, "Experimental researches of non-stability of thermocouples scale characteristics for Cr-Al", *Measurement Technique*, Moscow, USSR, 1985, № 10, pp.28-29.

[15] V.Kochan, V.Tymchyshyn, "Precision ADC module for IBM PC", *41.Internationales Wissenschaftliches Kolloquium*, Ilmenau, Germany, 1996, pp.668-672.