Brasília, 28 de novembro de 2007.

RELATÓRIO DE ATIVIDADES DE PROJETO DE PESQUISA
Novembro/2007

Edgar Jhonny Amaya Simeón

FINATEC
ELETRONORTE
Universidade de Brasília – UnB

1. Objetivo

Apresentar o trabalho feito no quinto trimestre do projeto de pesquisa que é parte integrante do projeto de modernização da área de automação de processos da usina hidrelétrica de Balbina e Samuel, desenvolvido pelo programa de pesquisa e desenvolvimento tecnológico da empresa de geração e transmissão de energia elétrica, ELETRONORTE, em parceria com a FINATEC e a UnB.

2. Atividades realizadas

Podem ser listadas as seguintes atividades específicas:

- ✓ Armazenamento das tags obtidas atraves do OPC no Banco de dados Simprebal, essas tags vão ser usadas para a analise de tendencias das grandezas dos devices existentes na usina.
- ✓ Verificação do estado de conexão do servidor ikernel com o servidor OPC e as DFI da Smar, o procesamento inteligente vai se efectuar sempre e quando existe conexão primeiro entre os dois servidores e depois verificar a conexão do servidor OPC con as DFI, tendo 15 DFI na usina, é importante conhecer a conetividade dos DFI e processar as regras por DFI.
- ✓ Integração dos dois servidores OPC que usavam diferentes librarias numa só, no desenvolviemento de I-kernel uso-se dois librarias do OPC, OPCCllient e Openscada, para uma melhor comparaçao das dois junto-se num só servidor configuravel atraves do arquivo config.ini.
- ✓ Posters apresentados no COBEM 2007 - 19th International Congress of Mechanical Engineering, “Different Control Strategies used in Didactic Plant PD-3 Of Smar Through OPC Technology” e “An Intelligent Kernel for the Maintenance System of a Hydroelectric Power Plant”.

3. Armazenamento das tags OPC no Banco de Dados Simprebal.

Para o armazenamento das tags obtidas do servidor OPC no banco de dados do simprebal, primeiro crio-se um banco de dados chamado simprebal localizado no computador 164.41.17.29, as configurações de conetividade foram escritas no arquivo de configuração config.ini.

```
[DBServers]
Assetview
Simprebal

[Simprebal]
user = alvares
password = eletronorte2003
DriveAddress = com.mysql.jdbc.Driver
ComAddress = jdbc:mysql://164.41.17.29:3306/simprebal
tags = DbsimTags
```

3.1 Arquivo de Tags

Os nomes das tags vão ser os mesmos dos grupos, o armazenamento no banco de dados vai ser feito por grupos.

```
[DBsimTags]
UGH1.GEP.GEP = real
UGH1.GEP.SRG = real
UGH1.GEP.SRT = real
UGH1.SMN.MGT = real
UGH1.SMN.MCB = real
UGH1.SMN.MGG = real
UGH1.SMN.SRLMCB = real
UGH1.SMN.SRLMGG = real
UGH1.THP.SVE = real
UGH1.THP.SRV = real
UGH1.THP.SAD = real
UGH1.THP.SLF = real
UGH1.THP.SDT = real
UGH1.SME.CTA = real
UGH1.SME.TF = real
```

```
[SELECT]

; ------------  Tags of the Simprebal Database ---------------

UGH1.GEP.GEP = SELECT value FROM tagteste WHERE nro = 1

UGH1.GEP.SRG = SELECT value FROM tagteste WHERE nro = 2

UGH1.GEP.SRT = SELECT value FROM tagteste WHERE nro = 3

UGH1.SMN.MGT = SELECT value FROM tagteste WHERE nro = 4

UGH1.SMN.MCB = SELECT value FROM tagteste WHERE nro = 5

UGH1.SMN.MGG = SELECT value FROM tagteste WHERE nro = 6

UGH1.SMN.SRLMCB = SELECT value FROM tagteste WHERE nro = 7

UGH1.SMN.SRLMGG = SELECT value FROM tagteste WHERE nro = 8

UGH1.THP.SVE = SELECT value FROM tagteste WHERE nro = 9

UGH1.THP.SRV = SELECT value FROM tagteste WHERE nro = 10

UGH1.THP.SAD = SELECT value FROM tagteste WHERE nro = 11

UGH1.THP.SLF = SELECT value FROM tagteste WHERE nro = 12

UGH1.THP.SDT = SELECT value FROM tagteste WHERE nro = 13

UGH1.SME.CTA = SELECT value FROM tagteste WHERE nro = 14

UGH1.SME.TF = SELECT value FROM tagteste WHERE nro = 15


[INSERT]

UGH1.GEP.GEP = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.GEP.SRG = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.GEP.SRT = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SMN.MGT = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SMN.MCB = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SMN.MGG = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SMN.SRLMCB = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SMN.SRLMGG = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.THP.SVE = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.THP.SRV = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.THP.SAD = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.THP.SLF = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.THP.SDT = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SME.CTA = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')

UGH1.SME.TF = INSERT INTO tagteste (nro, label, id, value, quality, status) VALUES( %s, '%s', '%s', '%s', '%s', '%s')
```

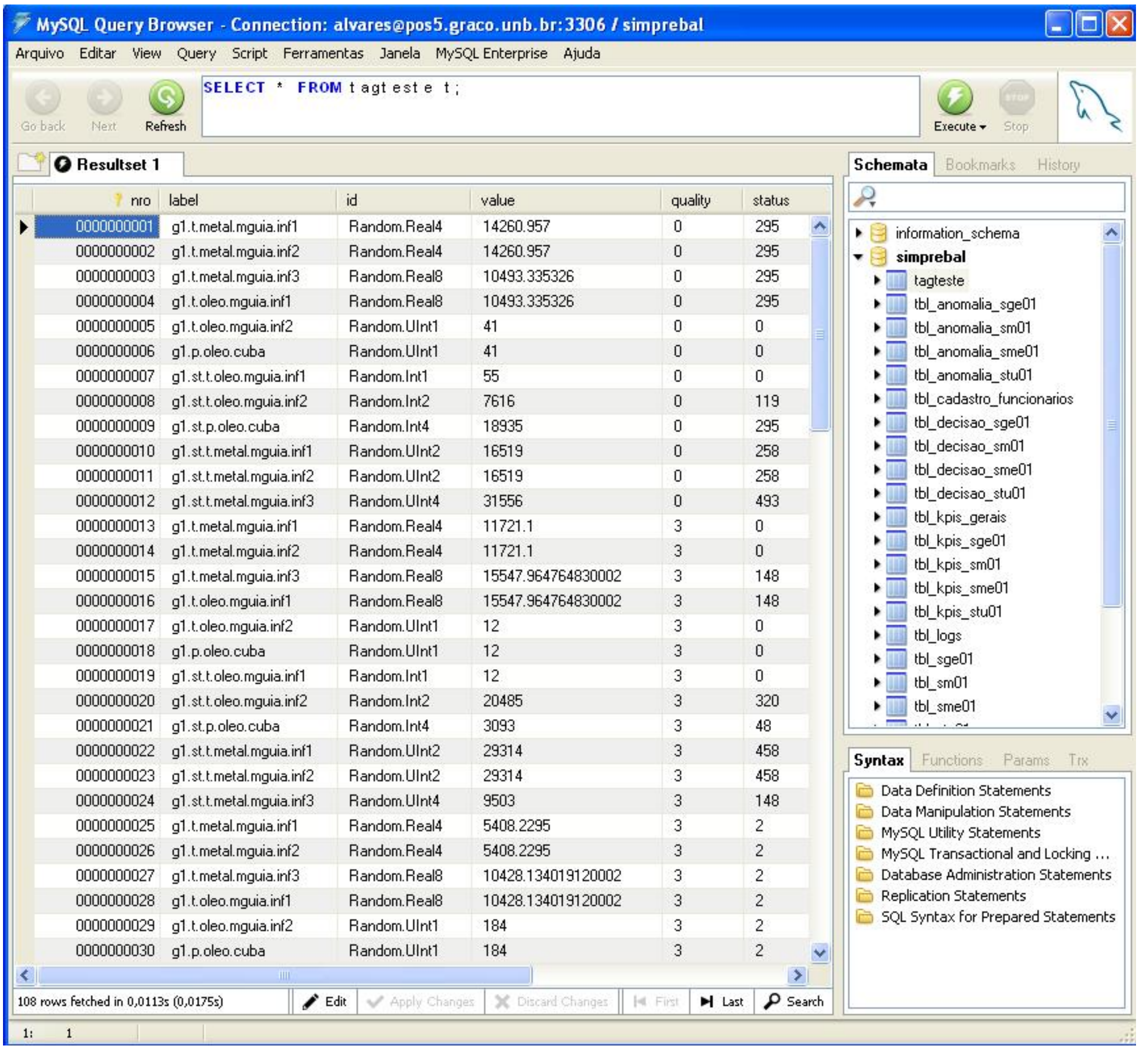


Fig. 1 – tags armazenadas nas tabelas do Banco de dados Simprebal

4. Verificação do estado de conexão do servidor OPC e as DFIs

4.1 Estado de Conexão do Servidor Ikernel com os Servidores OPC

Para a avaliação do estado de conexão do servidor ikernel com o servidor OPC é feito um metodo em java que envia um ping no IP configurado no arquivo config.ini este processo é feito pra cada um dos servidores OPC achados na usina.

```
public String checkopcserver() {
    String estado = "";
    if (Conf.OPCLibrary == 1){
        for (OPCServerjopc opcs : Conf.ListofOPCServersjopc) {
            try {
                if (InetAddress.getByName(opcs.host).isReachable(5000))
                    opcs.status = "Connected";
                else
                    opcs.status = "Disconnected";
            }catch (Exception e) {
                opcs.status = "Disconnected" + e;
            }
            estado = opcs.status;
            logger.Log("HOST = "+opcs.host + " ; "+"STATUS = "+estado);
        }
    }
    if (Conf.OPCLibrary == 0){
        for (OPCServerosc opcs : Conf.ListofOPCServersosc) {
            try {
                if (InetAddress.getByName(opcs.host).isReachable(5000))
                    opcs.status = "Connected";
                else
                    opcs.status = "Disconnected";
            }catch (Exception e) {
                opcs.status = "Disconnected" + e;
            }
            estado = opcs.status;
            logger.Log("HOST = "+opcs.host + " ; "+"STATUS = "+estado);
        }
    }
    return estado;
}
```

4.2 Estado de Conexão do Servidor Ikernel com os Dispositivos DFIs

No arquivo de configuração config.ini deve ter todos os dispositivos DFI que estejam conetado ao servidor OPC, cada DFI é identificado por um numero de IP

```
[DFIDevices]
dfi1a = 164.41.17.130
;dfi1b = 164.41.17.131
;dfi1c = 164.41.17.132
;dfi2a = 164.41.17.133
;dfi2b = 164.41.17.134
;dfi2c = 164.41.17.135
;dfi3a = 164.41.17.136
;dfi3b = 164.41.17.137
;dfi3c = 164.41.17.138
;dfi4a = 164.41.17.139
;dfi4b = 164.41.17.140
;dfi4c = 164.41.17.141
;dfi5a = 164.41.17.142
;dfi5b = 164.41.17.143
;dfi5c = 164.41.17.144
```

O programa desenvolvido em java abre o socket na porta 23 em cada um dos DFI existentes na lista do config.ini, este processo é executado em cada tick do servidor ikernel.

```java
public String checkdfis() {
    String estado = "";
    boolean ed = false;
      for (DFIDevice dfi : Conf.ListofDFIDevices ) {
         try {
            Socket socketDfi = new Socket(dfi.ip, 23); // criacao do socket no cliente
            if (socketDfi.isConnected()){dfi.status = "Connected";}
            socketDfi.close();
         }
         catch(UnknownHostException e) { dfi.status = "Not Found";}
         catch(java.io.IOException e) { dfi.status = "Disconnected";}
         estado = dfi.status;
         logger.Log("DFI = "+dfi.ip + " ; "+"STATUS = "+estado);
     }
     return estado;
  }
```

5. Integração das duas Librarias num só Servidor OPC

No desenvolvimento do Ikernel, esta-se usando 2 librarias para conexão com OPC server, JOPCClient e Openscada, cada um das librarias tem classes e metodos diferentes, o objetivo foi criar um só servidor que use as duas librarias configuravel via o arquivo config.ini isto vai ser usado pra comparar o desempenho de cada um das librarias e para ter um só servidor ikernel em desenvolvimento.

5.1 Arquivo de Configuração

No arquivo config.ini. tem um campo que é OPCLibrary pudendo ser 0 = Openscada, 1 = JOPCClient, depende do qual das duas librarias vai ser preciso usar.

[OPCLibrary]

; 0 = Openscada, 1 = JOPCClient.

0

6. Ferramenta de configuração e monitoramento.

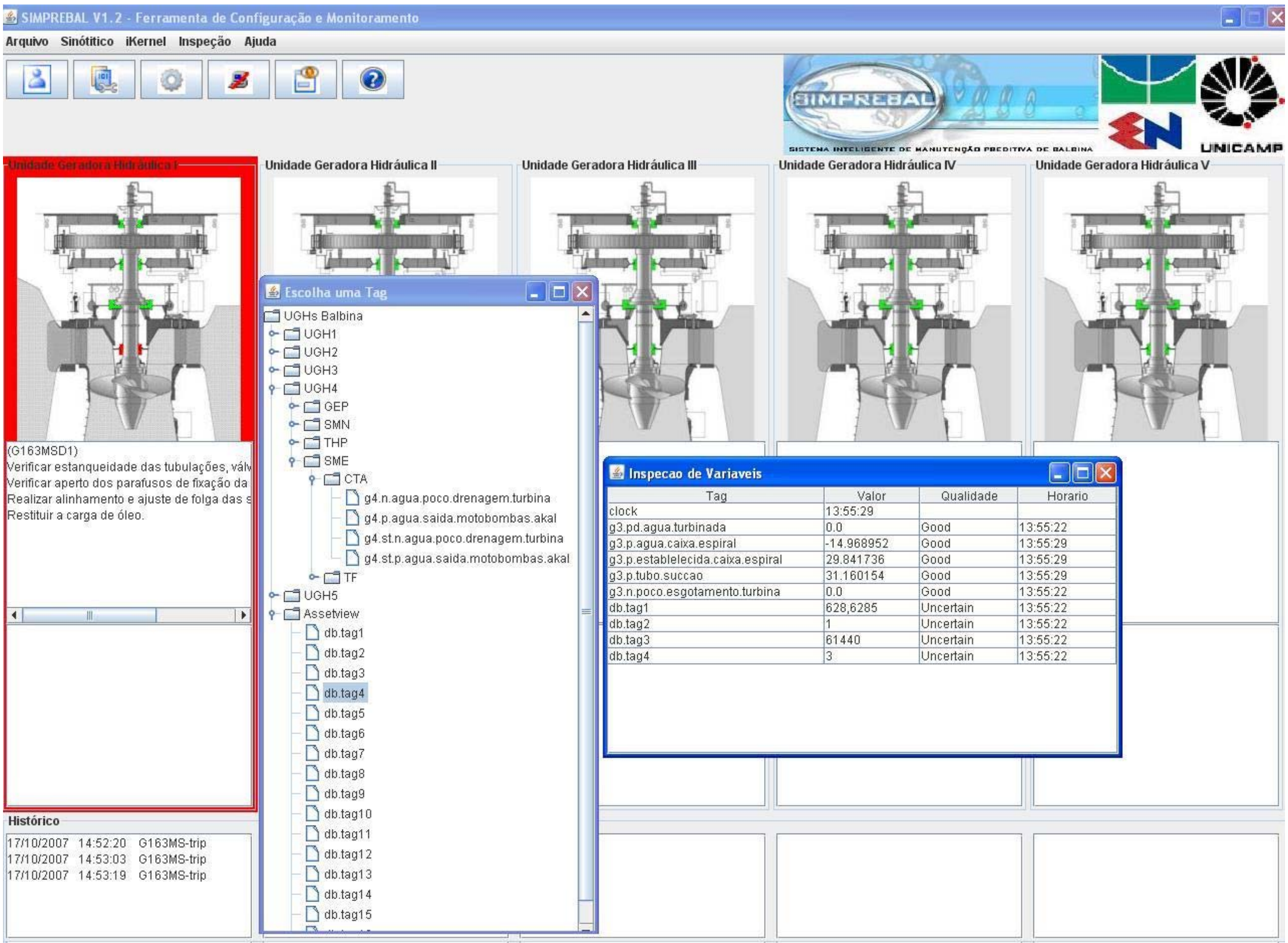


Fig. 2 – ferramenta configuração e monitoramento

Anexo 1

Posters Apresentados no COBEM2007

ABCM
ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA E CIÊNCIAS MECÂNICAS

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

AN INTELLIGENT KERNEL FOR THE MAINTENANCE SYSTEM OF A HYDROELECTRIC POWER PLANT

Authors:

Edgar Jhonny Amaya Simeón[1], Rosimarci Pacheco Tonaco[1], Alberto José Álvares[1], Ricardo Ribeiro Gudwin[2], Rodrigo de Queiroz Souza[1].

[1] Universidade de Brasília (UnB), Departamento de Engenharia Mecânica e Mecatrônica, Grupo de Inovação em Automação Industrial (GIAI), Brasília, DF, Brasil.

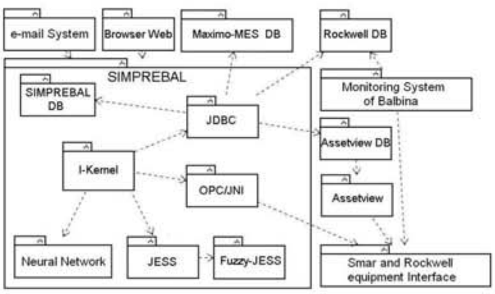
[2] Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação, Departamento de Engenharia de Computação e Automação Industrial, Campinas, São Paulo, Brasil.

Abstract

In this work, we present the conceptualization of an intelligent system to be used on the predictive maintenance of a hydroelectric power plant. It is a real application, being developed in the context of the project "Modernization Processes of Automation Area of the Hydroelectric power stations of Balbina and Samuel", with the support of Eletronorte. The work described in this paper corresponds to the development of an intelligent kernel, which is planned to be the heart of a major methodology, called SIMPREBAL (Predictive Maintenance System of Balbina), to be implanted in the plant of Balbina, near to the city of Manaus in Brazil. The overall specification and design of this intelligent kernel is described here using UML (Unified Modeling Language) diagram. This intelligent kernel, called here I-kernel, will be providing expert systems functionality, besides fuzzy logic rules processing and neural networks in an operational cycle where data is collected from equipments and databases, and will be used to both give support to the operational team and to the maintenance team at the plant. I-Kernel will get the data from supervision system by means of both JDBC access to databases and JNI (Java Native Interface) getting data from the OPC (OLE - Object Linking and Embedding - for Process Control) server. In order to provide expert systems and fuzzy logic functionality to the system, we will be using the tools JESS (Java Expert Systems Shell) and Fuzzy-JESS. The SIMPREBAL methodology is based on RCM (Reliability Centered Maintenance) concepts, being used to analyze the manners and effects of fails on the Hydraulic Generator Units of Balbina.

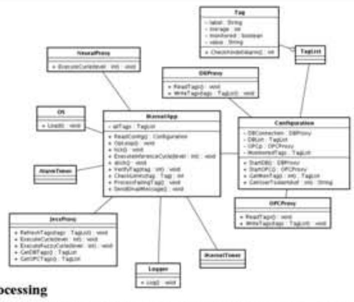
I-Kernel Subsystem Specification

The I-Kernel subsystem is a software component, to be developed in Java language, responsible for the data acquisition from databases and process equipments through OPC, and its processing through conventional rules, fuzzy logic and neural network, in N layers. Many of the functionalities required for I-Kernel are already available as out-of-the box software components. The components to be used are the following: JESS - Java Expert System Shell, developed by Sandia National Laboratories, responsible by the rule processing and main inference machine to the expert system part of I-Kernel; Fuzzy-JESS, developed by the NRC Institute for Information Technology, responsible for the processing of fuzzy rules, in a way similar to JESS; JDBC responsible for the access to the databases and JNI, responsible for the access to the OPC servers which give direct access to the equipments. The I-Kernel application needs to access different databases, in order to integrate to the computational environment of Balbina's power plant. This includes the database for the Rockwell's Monitoring System, Smar's Asset Management System - Assetview, IBM/MRO's Operating and Maintenance Management System MAXIMO-MES (Manufacturing Execution System), and also the own SIMPREBAL database which will be storing all the knowledge required by the different intelligent algorithms. The Neural Network module will be developed almost from scratch, based on some code available at UNICAMP, built according to the scientific literature. A general view of this architecture, in terms of a UML diagram, is available in the next figure.



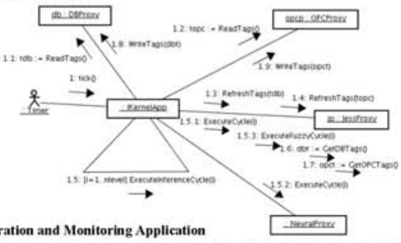
I-Kernel Application

The I-Kernel application is a server that doesn't have an user interface. In this way, the only thing that the user can do is to start the application. After that, a timer previously programmed sends periodic "ticks" that start each operational cycle of intelligent processing and make alarms and alerts verification.



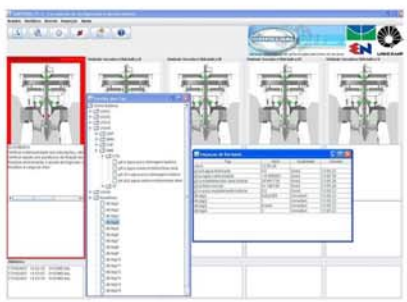
Intelligent Processing

The intelligent processing is the heart of the application. It starts with a "tick" from the IKernelTimer, which trigger a sequence of operations: first it asks the DBProxy for the tags to be read from the database, then it asks the OPCProxy for the tags to be read directly from the Process Equipments, then both tags from database and from OPC servers are fed into JESS. After that, the system starts an n-level loop, where for each loop, the system asks the JESS engine to perform one operational cycle, which first execute JESS rules, then Fuzzy JESS rules and then the neural network.



Configuration and Monitoring Application

The Configuration and Monitoring application is a web application, destined to promote the configuration of parameters of the I-Kernel, as well as the Monitoring of the variables under control of the system and the request of shutdown of the I-Kernel. This monitoring can be a direct supervision of the state of some variable (by the selection of the variable among all those which are available), or it can be a monitoring by synoptic. In the direct variables monitoring, the user chooses the variable to be monitored, and the system exhibits its state directly.



Conclusions

We briefly presented in this paper the requirements and design of an intelligent kernel (I-Kernel) to be used in the construction of a Predictive Maintenance System of a Power Plant. The modelling complies with RCM (Reliability centered maintenance) concepts. The system basically consist of the ConfMonitToolApp module and the I-kernelApp module. This application is actually going to a coding phase, and the next steps are to fully test the application within the power plant. Despite its use in this particular application, the concept of an intelligent kernel goes beyond simply this project, being useful for the construction of similar applications in the future.

Acknowledgements

We acknowledge the support of the Eletronorte and Manaus Energia provided by the Reserch and Development Program under contract number 4500052325, project number 128 "Modernization of Processes Automation Area of the Hydroelectric power stations of Balbina and Samuel", that has as a technical responsible Prof. Alberto José Álvares of the UnB, and we acknowledge too to the engineer Antonio Araujo from Eletronorte, who played a significant role in this work for the information provided that was important to develop this project.

ABCM — ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA E CIÊNCIAS MECÂNICAS

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Mecânica

DIFFERENT CONTROL STRATEGIES USED IN DIDACTIC PLANT PD-3 OF SMAR THROUGH OPC TECHNOLOGY

Authors:

Edgar Jhonny Amaya Simeón[1], Alberto José Álvares[1], Victor Rafael R. Celestino[1], Carlos Frederico Maciel Silveira[1].

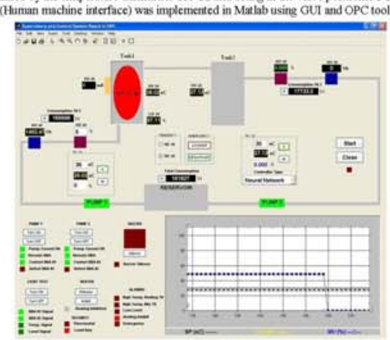
[1] Universidade de Brasília (UnB), Departamento de Engenharia Mecânica e Mecatrônica, Grupo de Inovação em Automação Industrial (GIAI), Brasília, DF, Brasil.

Abstract

In this work, three different control strategies, neural network, fuzzy logic and PID (Proportional Integral Derivative), were implemented in Matlab, acting on a Smar PD3 didactic plant by means of OPC (OLE - Object Linking and Embedding - for Process Control) technology. The PD3 plant employs Foundation Fieldbus protocol and configuration tools of Smar System 302. The PD3 process chosen for testing control strategies consists to control the temperature on the mix water tank , which has a constant flow input of constant temperature hot water and an input consisting of ambient temperature water controlled by a pneumatic valve. The PID control was developed using differences equation and adjusting the parameters using a Ziegler- Nichols method. The Neural Network, that has a structure (3-20-1) capable to learn the dynamic of the PID controller. The training was accomplished using a backpropagation method that is least Square Method (LMS). The Fuzzy Logic Controller was developed using the toolbox of the Matlab, membership function was edited in the FIS editor. In addition to the controllers a supervisory was developed using GUI of Matlab.

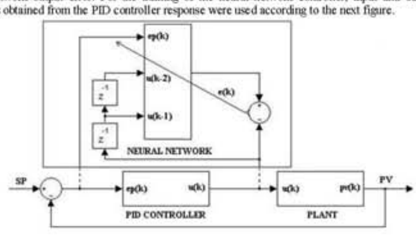
Smar Didactic Plant PD3

The PD3 didactic plant contains two flow loop: The flow of the loop 1 is generated by pump1, from water reservoir tank to tank1 (hot water tank), all the hot water is transferred to the tank2 (mix water tank). The TIC-31 is a single PID controller that is commissioned to control the water temperature in the tank1, temperature measure is obtained by the temperature transmitter TIT-31 and acting in the static converter TY-31. The flow of the loop 2 is generated by pump2, from water reservoir tank to tank2. The TIC-32 can be any of the control strategies implemented in Matlab, like Neural Network, Fuzzy Logic or PID controller, that's located in the micro computer, is commissioned to control the water temperature in the tank 2, temperature measure is obtained by the temperature transmitter TIT-32 and acting in the valve positioner FY-32. The HMI (Human machine interface) was implemented in Matlab using GUI and OPC toolbox.



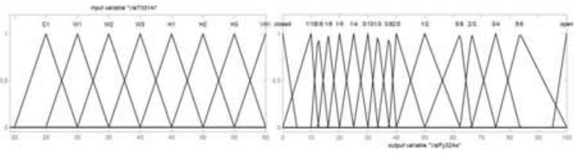
Artificial Neural Networks (ANN)

The neural network structure: 3 inputs, 20 neurons in the hidden layer and 1 neuron in the output layer. The activation function in the hidden layers is the hyperbolic tangent function and in the output layer is the sigmoid function. The backpropagation method, utilized in this work, belongs to the family of Least Mean Square (LMS) methods, in which the minimum value of the quadratic error function is sought, based in the gradient method. The training algorithm basically consists in correcting in the output layer and in the hidden layers synaptic weights, on the basis of the error occurred in each output of the neural network. Weight correction is done by means of the minimum quadratic means method, aiming to find values for the weights which minimize the network output error. For the training of the neural network controller, input and output values obtained from the PID controller response were used according to the next figure.



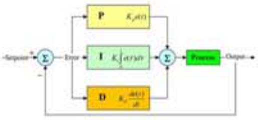
Fuzzy Logic Controller

The fuzzy logic controller applied in this work is composed of a set of inference rules of a type such as If <expression> AND <expression> Then <result>, which define control actions as a function of the several value ranges, which plant input variables can assume. 225 inference rules were implemented. Using the Fuzzy Logic Toolbox of matlab, the controller was implemented using FIS editor. The controller was built based on Mamdani model and max-min composition was used for operations with fuzzy numbers: min operator for implying and max for aggregation. The centroid method was used for defuzzification. The controller uses three input variables, which are extracted from plant output: water temperature in the hot water tank TIT-31, water temperature in the mix water tank TIT-32 and set point of the water temperature in the mix water tank TIT-32SP. The output is the open percent value of the control valve FY-32. Input variables assume values: cold (C1), warm (W1, W2, W3), hot (H1, H2, H3), and very hot (VH1, VH2, VH3). The membership functions have the form presented in the next figure.



PID Controller

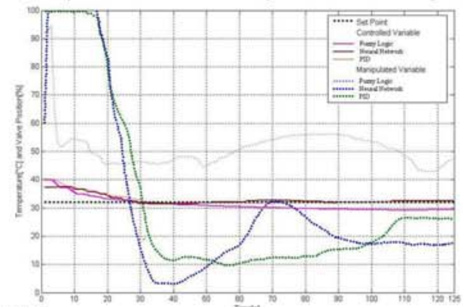
For the tuning of the PID controller, a Ziegler-Nichols method was used, where an experimental adjustment of PID parameters was done. The method works in the following manner: A step function is placed as set point and PID is turned-on, with Tr equal to infinity and Ti equal to zero; Kp gain is adjusted until reaching marginal stability condition. This gain, de Kpc, is named critical gain. The resulting sinusoid period as marginal stable output is indicated by Ppc. The final Kp value is set as 0.6 Kpc, Ti as 0.5 Ppc, and Td as 0.125 Ppc. Using the algorithm above and applying a small empirical adjustment on top of values found, the following parameters were used for the implementation: Kp = 3.7; Ti = 9; Td = 1. For all tests and results obtained, it was used T = 2 s.



	Neural Network	PID	Fuzzy Logic
Overshoot[%]	1.75	0.78	8.6
Overshoot Time[s]	34	54	110
Settling Time[s]	90	85	>120
Rise Time[s]	20.4	21.3	20

Tests and Results

Defined the test for functional analyzes of these controllers, the temperature of the hot water tank with its automatic temperature control loop around 38°C and the temperature of the cold water from the reservoir around 26°C, the set point of the controllers were at 32°C, the process start with the water temperature at 38°C, the mission of the controllers is to arrive to 32°C and maintain in this value. The Neural Network controller with its weights learned starts the control loop. In the table before is ilustrated the comparison of the three Control Strategies Response.



Conclusions

In this paper, a comparative study was presented using a neural network, fuzzy logic and a conventional proportional integral derivative controller, PID. The controllers were applied to a didactic plant of Smar. In the tests were observed that the Fuzzy Logic controller arrive to the set point slowly because were considered few rules and membership function. The neural network envelop learning and training in this case was necessary to have as a teacher to the PID controller. The PID require mathematical equation and tests to tuning the constants Kp, Ki and Kd. The fuzzy logic controller demand human experience that know how to interact with the plant dynamic. The employed industrial protocol Fieldbus Foundation to communicate the supervisory system and Smar instrumentation permit to control the plant remotely. Future work includes the increasing dynamic to the fuzzy controller and to made neuro fuzzy controller in order to had better response behavior.

Acknowledgements

We acknowledge the support of the Eletronorte and Manaus Energia provided by the Reserch and Development Program under contract number 4500052325, project number 128 "Modernization of Processes Automation Area of the Hydroelectric power stations of Balbina and Samuel", that has as a technical responsible Prof. Alberto José Álvares of the UnB, and we acknowledge too to the engineer Antonio Araujo from Eletronorte, who played a significant role in this work for the information provided that was important to develop this project.

Edgar Jhonny Amaya Siméon – Aluno do curso de Sistemas Mecatrônicos – UnB

Prof. Alberto José Álvares – ENM/UnB (Orientador)