Wind maintenance system using network synchronization techniques based on open-source software

INÁCIO FONSECA[1]; JOSÉ TORRES FARINHA[2]; FERNANDO MACIEL BARBOSA[3]
Instituto Superior de Engenharia[1,2]; Faculdade de Engenharia[3]
Instituto Politécnico de Coimbra[1,2]; Universidade do Porto[3]
Rua Pedro Nunes - Quinta da Nora, 3030-199 Coimbra[1,2]; Rua Dr. Roberto Frias, 4200-465 Porto[3]
PORTUGAL[1,2,3]
inacio@isec.pt[1]; tfarinha@isec.pt[2]; fmb@fe.up.pt[3]

***Abstract**:* - The use of open-source software in many institutions and organizations is increasing. However, a balance should be considered between the software cost and the cost of its technical support and reliability. In this article, a maintenance system for wind farms will be presented. It is connected to an information system for maintenance, called SMIT (Terology Integrated Modular System) as a general base to manage the assets and as a support strategic line to the evolution of this system, which incorporates on-condition maintenance modules, and the support to the research and development done around this theme. The SMIT system is based on a TCP/IP network, using a Linux server running a *PostgreSQL* database and *Apache* web server with *PHP*, and *Octave* and *R* software for numerical analysis. Maintenance technicians, chiefs, economic and production management personnel can access SMIT database through SMIT clients for *Windows*. In addition, this maintenance system for wind systems uses also special low cost hardware for data acquisition on floor level. The hardware uses a distributed TCP/IP network to synchronize SMIT server master clock through Precision Time Protocol. Usually, the manufactures construct, deploy and give the means for the suppliers to perform the wind system's maintenance. This is a very competitive area, where companies tend to hide the development details and implementations. Within this scenario, the development of maintenance management models for multiple wind equipments is important, and will allow countries to be more competitive in a growing market. For on-condition monitoring, the algorithms are based on Support Vector Machines and time series analysis running under *Octave* and *R* open-source software's.

Key-Words: - Wind energy; renewable energy; predictive maintenance; clock synchronization.

1 Introduction

A wind turbine is a complex system with several components changing constantly and supporting strong forces. By consequence, it can experience many problems, such as vibrations, electrical failures and many other kinds of faults [8-12]. Additionally, wind farms are usually far from cities and from companies supporting their maintenance. Technical assistance is expensive and the combination of on-condition maintenance with the best practices of operational research to minimize distance costs is extremely important. Within this work, the main objective is to implement a maintenance plan using, namely, on-condition maintenance through on-line data instrumentation, acoustic techniques, vibration techniques [1-2], infrared images, stress measurement, zero crossing current analysis and artificial intelligence, in a coherent and synergetic way.

Another strategic objective of this work is to build the entire system with open-source and low cost hardware [3]. The reliability of actual microcontrollers and open-source software is of great importance for market penetration. The increment on sustainable ecological energy is in mind of all political persons in the planet.

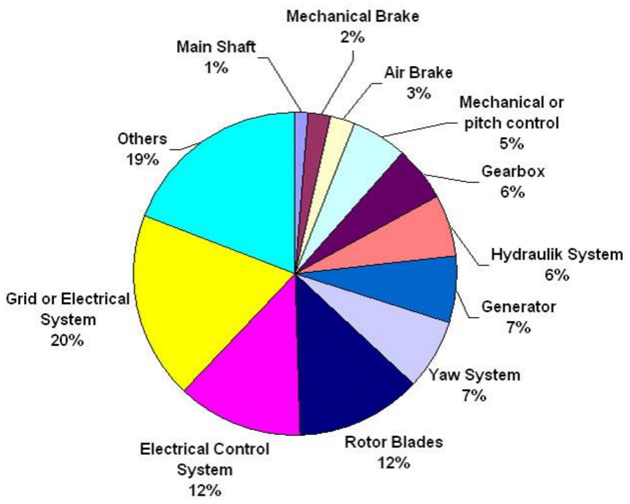


Fig. 1 - Share of components in the total failures in a Wind Turbine [8]

Nowadays, green energy is a very important subject and countries believe it is the good direction to the ecological and oil savings. This is the case, not only for wind energy, but also for other kind of green energy like photovoltaic solar energy, sterling solar energy, and geothermal energy, naming only a few.

Countries are now betting on the installation of wind power generation. This factor is not only important for green energy production, for companies and for less oil energy dependency, but also for economical reasons. The financial money paid for oil does not leave the country and, in alternative, can be applied in the construction of new wind farms, thereby leveraging an energy resource. Some authors have reported different types of failures in wind turbines, which presume the existence of an area of great economic importance, such as the financial costs about maintaining these facilities. Some of these reported failures can be seen on Fig. 1. In this scenario, maintenance problems will continue to grow in the future because of the oldness of equipments and, in a reasonable way, it is important to study this problem to increase knowledge.

2 A Wind Maintenance System

Fig. 2 gives an idea of the system overview. SMIT [6] system uses open-source software to achieve a good performance between cost and reliability.

For operating system, Linux *Slackware* was used, with *PostgreSQL* database. The mathematical support is based on *R* and *Octave* algorithms (see Fig. 3) [13], using Time Series Analysis to follow trends on key variables from the field [14-23]. At the time the Linux distribution supporting SMIT was Slackware version 12.1 and also FreeBSD version 7.1 [3].

For remote access by browser technology an Apache server running PHP is used, versions 2.2.4 and 2.5.1, respectively. Some pieces of the maintenance software modules are available for maintenance intervention requests and information exchange with third parties using web services PHP technology.

Stored procedures are written in PL/pgSQL a native language similar to Oracle's stored procedure language. SMIT database includes 149 tables and 156 PL/pgSQL stored procedures. The SMIT users can interact with the main system using windows interface modules programmed in Delphi 7. This is the single no open-source programming tool used. However, especial design as been done to easily translate all source code with minor changes to be compiled by freepascal [4] and the RAD development with Lazarus project [4], as soon as its reliability on certain components is assured. Reports are also developed with a commercial program, Crystal Report (integrated in Delphi) and some others with PHP, both stored in special tables in SMIT database. Its portability to open-source is also possible and will be achieved by PHP reporting tools.

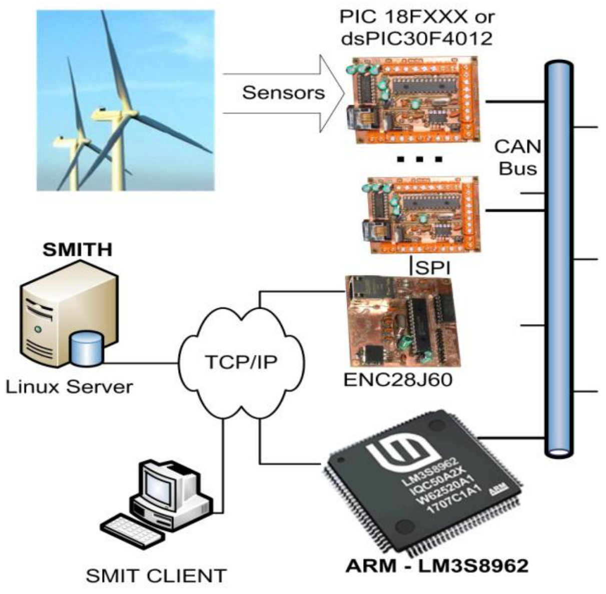


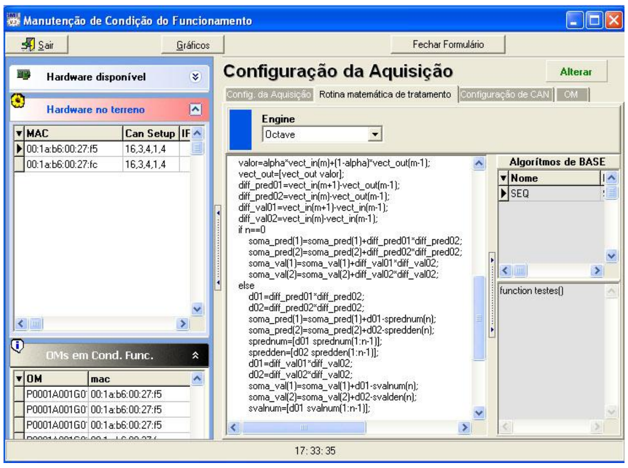
Fig. 2 - Global overview of SMIT system from software to hardware components

To protect this intellectual property around the development, particularly in the PHP and SQL postgreSQL scripts, two software tools for encryption had been created. For SQL scripts, only the PL / pgSQL functions are encrypted; for the PHP the entire contents of the program are encrypted. Obviously, to accomplish this task, there was a need to change the source of their distributions. Obviously, to carry out this task, there was the necessity to modify the source code of the respective distributions (PHP and PostgreSQL). With this solution, the administrator password of server SMIT can be provided to third parties if required. Another security feature is related to all the external connections to SMIT's database that are made with SSL sockets implemented through OpenSSL.

2.1 Hardware

Many manufactures use industrial PLCs to control and acquire sensorial data in the wind turbine equipments. In Portugal many examples can be found. It is also normal to have a higher failure rate in these components due to instability in the voltage. These aspects give another argument to use low cost hardware, also envisaging the possibility of supplying the hardware with photovoltaic panels or batteries avoiding the problems with instable voltage.

Our low cost acquisition system includes PIC18F2685, ENC 28J60 for Ethernet connectivity, a dsPIC30F4012 for high speed acquisition [7], a board using Microchip digital potentiometers to implement a Butterworth low pass filter of 4th order with cut-off frequency between 100 Hz and 50 kHz and two cascade amplifiers with gain range between 0.1 and 10. The frequency and gain are programmed by software and, finally, a board based on the microcontroller LM3S8962, an ARM-Cortex-M3 architecture with support for Ethernet packet time stamping in hardware, with two interfaces Ethernet at a 10/100 Mbps full/half duplex and a CAN 2.0B. All the programming tolls for this microcontroller as been constructed based on GNU GCC tool chain for ARM-Cortex-M3 version 4.3.3, Binutils 2.19.1 and newlib-1.17.0 under Gygwin. This is the most expensive component of the system; however, it is priced by 12 €/unit and, in addition, an operational board runs with a few elements. This board will be a gateway between the traffic from the CAN network and the Ethernet network.



Engine: Octave

MAC	Can Setup	IF
00:1a:b6:00:27:f5	16,3,4,1,4	
00:1a:b6:00:27:fc	16,3,4,1,4	

```
valor=alpha*vect_in(m)+(1-alpha)*vect_out(m-1);
vect_out=[vect_out valor];
diff_pred01=vect_in(m+1)-vect_out(m-1);
diff_pred02=vect_in(m)-vect_out(m-1);
diff_val01=vect_in(m+1)-vect_in(m-1);
diff_val02=vect_in(m)-vect_in(m-1);
if n==0
   soma_pred(1)=soma_pred(1)+diff_pred01*diff_pred02;
   soma_pred(2)=soma_pred(2)+diff_pred02*diff_pred02;
   soma_val(1)=soma_val(1)+diff_val01*diff_val02;
   soma_val(2)=soma_val(2)+diff_val02*diff_val02;
else
   d01=diff_pred01*diff_pred02;
   d02=diff_pred02*diff_pred02;
   soma_pred(1)=soma_pred(1)+d01-sprednum(n);
   soma_pred(2)=soma_pred(2)+d02-spredden(n);
   sprednum=[d01 sprednum(1:n-1)];
   spredden=[d02 spredden(1:n-1)];
   d01=diff_val01*diff_val02;
   d02=diff_val02*diff_val02;
   soma_val(1)=soma_val(1)+d01-svalnum(n);
   soma_val(2)=soma_val(2)+d02-svalden(n);
   svalnum=[d01 svalnum(1:n-1)];
```

Nome
SEQ

```
function testes()
```

OM	mac
P0001A001G0	00:1a:b6:00:27:f5
P0001A001G0	00:1a:b6:00:27:f5
P0001A001G0	00:1a:b6:00:27:f5

17: 33: 35

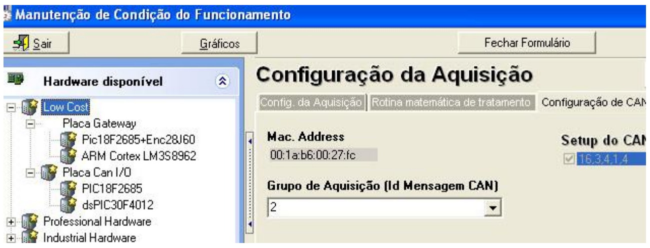
Fig. 3 – *Octave script using time series algorithms, for numerical data analysis*

3 System overview

The acquisition timings are saved in SMIT's database where the global acquisition network is designed and stored. At a first step, a SMIT user will configure the acquisition network, choosing the boards on the field. The gateway board will use DHCP protocol to acquire an IP number, and the setup for CAN network is also stored in the database. Under this perspective it is very easy to change CAN network velocity by changing the parameters associated with the CAN network, as can be seen in Fig. 4.

In the second step, the information related to frequency sampling is indicated, and also the temporal interval like the starting date and final date of the acquisition, Fig. 5. These parameters are downloaded by the gateway board to control the sampling rate in the CAN bus, generating control signals for the I/O boards.

Another goal is to synchronize the acquisition boards through time propagation from the SMIT server to PIC microcontrollers in the CAN bus using SNTP or PTP running in a cooperative way in the Ethernet-CAN gateway [24, 25]. Under this feature it is possible to ensure that different devices placed in different wind turbines perform signal acquisition at the same time. This aspect makes possible the comparison among the same data in different wind turbines; it is guaranteed that the gap between the acquisition times is less than 10 micro seconds. The CAN slaves, in setup mode, will auto-baud the communication velocity until a valid CAN message is received. After this stage they will start the normal cycle, waiting for a message asking for an acquisition and forwarding packets for measuring CAN propagation delay time, or receiving messages from firmware.


Manutenção de Condição do Funcionamento
Sair
Gráficos
Fechar Formulário
Hardware disponível
Configuração da Aquisição
Config. da Aquisição
Rotina matemática de tratamento
Configuração de CAN

- Low Cost
  - Placa Gateway
    - Pic18F2685+Enc28J60
    - ARM Cortex LM3S8962
  - Placa Can I/O
    - PIC18F2685
    - dsPIC30F4012
- Professional Hardware
- Industrial Hardware

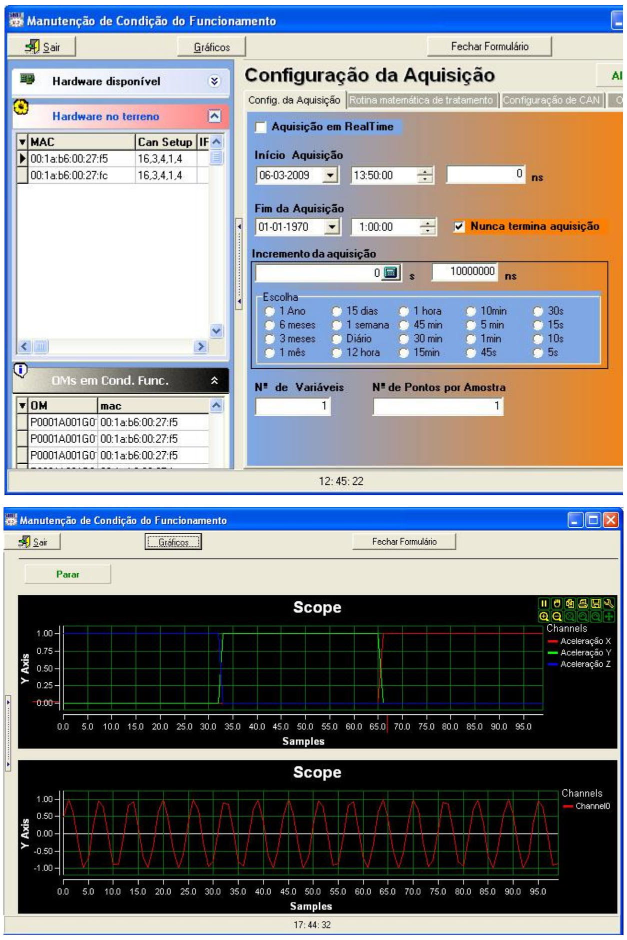
Mac. Address: 00:1a:b6:00:27:fc

Setup do CAN: ✓ 16,3,4,1,4

Grupo de Aquisição (Id Mensagem CAN): 2

Fig. 4 - Choosing hardware: in low cost mode a gateway is always needed as also an I/O CAN board.

The CAN bus devices are restricted to the global acquisition rate. If the number of nodes together generate data with a flow rate higher than the CAN network speed, the nodes must be divided by two (or more) different CAN networks. For synchronizing the master gateway Real Time Clock the system uses SNTP – Simple Network Time Protocol (PIC with CAN and ENC28J60 device) - and the board based on the LM3S8962 microcontroller uses PTP (Precision Time Protocol). The use of different protocols is justified by the Ethernet packet time stamping facility of the microcontroller that is fully explored by the PTP. The SMIT server runs an SNTP and/or PTP daemons servers.



MAC	Can Setup	IF
00:1a:b6:00:27:f5	16,3,4,1,4	
00:1a:b6:00:27:fc	16,3,4,1,4	

OM	mac
P0001A001G0	00:1a:b6:00:27:f5
P0001A001G0	00:1a:b6:00:27:f5
P0001A001G0	00:1a:b6:00:27:f5

Configuração da Aquisição

Config. da Aquisição | Rotina matemática de tratamento | Configuração de CAN

☐ Aquisição em RealTime

Início Aquisição: 06-03-2009 13:50:00 0 ns

Fim da Aquisição: 01-01-1970 1:00:00 ✓ Nunca termina aquisição

Incremento da aquisição: 0 s 10000000 ns

Escolha: 1 Ano, 6 meses, 3 meses, 1 mês, 15 dias, 1 semana, Diário, 12 hora, 1 hora, 45 min, 30 min, 15min, 10min, 5 min, 1min, 45s, 30s, 15s, 10s, 5s

Nº de Variáveis: 1

Nº de Pontos por Amostra: 1

Fig. 5 – Top: Programming the acquisition temporization for each low cost I/O board, in the SMIT's on-condition module;
Bottom: An example when the system is running.

3.1 Synchronous Data Acquisition

To ensure synchronous acquisition in different geographic locations, it is used the PTP protocol.

The protocol works in the following form: each slave synchronizes with the RTC of the master through a set of specific messages (*Sync*, *Delay Request*, *Follow Up* and *Delay Response*). In summary, the operation of the protocol is as follows [5, 25]:

a) The master sends a multicast *Sync* message, noting time (T_{m1}) and slaves, after receiving the message, record the internal time (T_{s2}). The frequency of messages is two seconds, $T_{Sync} = 2s$;

b) The master sends a *Follow UP* message with its time (T_{m1}) for slaves;

c) The difference between the two times is the delay from master to slave: $dm2s = T_{m1} - T_{s2}$;

d) The slaves send *Delay Request* messages, saving the time of transmission (T_{s3}) and the master receives the message, records the time of receipt (T_{m4}). The frequency of these messages is uniformly distributed between 2 to $30 \cdot T_{Sync}$ units;

e) Then, the master sends a delay response with time T_{m4};

f) The difference between the two times is the delay from slave to master: $ds2m = T_{s3} - T_{m4}$;

g) The propagation in the network is given by $T_{prop} = (dm2s + ds2m)/2$;

h) The clock in the slave should be carried out according to the quantity $\Delta t = dm2s + T_{prop}$.

3.2 Using PTP for clock synchronization

Formally, consider a master clock to which we want to synchronize in absolute terms; the period is $T_m = 1ns$, giving rise to two variables, seconds and nanoseconds, i.e., Ms: Mns. Each clock cycle, the nanoseconds variable, is incremented and if it overflows is normalized to the seconds variable. Consider, as an example of a node, one of the PIC microcontrollers, operating at a frequency of $f_{no} = 40Mhz$, with clock period of $T_{no} = 25ns$. The objective is to keep in this node two variables Ns:Nns in absolute synchronization with Ms:Mns variables. The maintenance of these variables is usually done through a system outage called *Tick Interrupt System* with a 10ms that is an adequate value for this timing.

Thus, the value of Nns is incremented by 10 ms (10.000.000ns) and, in the case of overflowing it will be normalized to the second's variable. In microcontrollers, when this interruption occurs, it is necessary to plan a counter hardware (CNT_TICK), as described, with the value CNT_TICK_MAX = 400.000 = 10ms/25ns. The counter starts at zero and counts up to limit (400,000), generates an interruption (to carry out maintenance on Ns:Nns), restarting to zero again. To determine the correct time, it is necessary to read the value of CNT_TICK and add it to the variables Ns:Nns. The accuracy of this system will be in the ceiling of the order of 25ns, whose value is impossible because it is the execution time of an instruction.

Assuming differences in the frequency of oscillation of both clocks, which is perfectly acceptable with reference to the sender, the model of this differential can be designed, without loss of generality, designated here as the clock error of the receiver, by

$$T_{no} = 25 \cdot T_m + \Delta T_{no} = 25ns + \Delta T_{no}$$

For a Δt_{no} positive it means that the period of the slave is higher than the one of the master. As a consequence, it will be increasingly slowly over time, ie introduces a negative error in counting the clock given by:

$$Error_{10ms} = -CNT\_TICK\_MAX.\Delta T_{no}$$

$$Error_{1s} = 100 \cdot Error_{10ms}$$

As the master frequency is Tm = 1ns that means:

$$\Delta T_{no} = T_{no} - 25 \cdot T_m$$

From this description it is clear that it is necessary to require $\Delta T_{no} \rightarrow 0$ and to change the value CNT_TICK_MAX. This control is not more than a PLL function, in this case implemented by software. Figure 6 presents the evolution of the clock node slave according to the PTP messages, considering the error displayed. Applying the changing technique to variables Ns:Nns, it appears to be impossible to take the error value equal to zero, as the diagram of Fig. 6 shown.

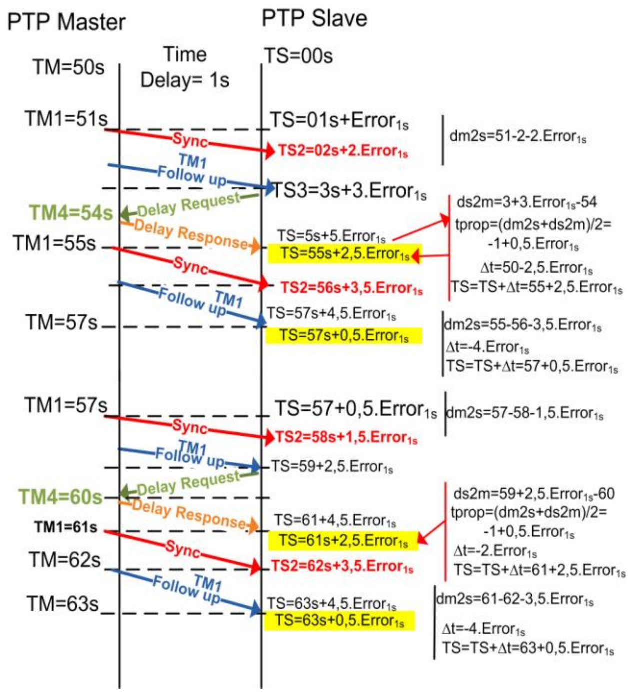


Fig. 6 – Timing diagram that considers a shift frequency in the slave node

The parameter CNT_TICK_MAX should be controlled and modified when calculating the error Δt in the receiving messages *Follow UP* (if not sent a *Delay Request* message) or at the reception of *Delay Response* messages, or both. This update of the base period of *System Tick Interruption* is directly related to the PLL operation. One additional note refers to the value of CNT_TICK_MAX which has the following limits: it can not be negative nor exceed the maximum allowable size of microcontroller register. This means that applying a threshold to the value calculated before will reflected in the CNT_TICK_MAX parameter.

Fig. 7 shows the diagram of the local clock integrated process control. The experiences carried on refer to the limits alteration of the threshold function and to the filtering function.

3.3 Simulation with Matlab

To study the behavior of the system, ie, filter and gain, as illustrated in Fig. 7, a simulator was implemented in Matlab. The difference with reality relates to the following aspects:

a) the messages in the network take 45ms to be transmitted (changeable value);
b) the times data that are sent and received are recorded properly, without errors;
c) the duration of the calculations in control is zero and does not influence the result; therefore, it is equivalent to stopping the time during those moments;
d) the time of the simulation is 1s, varying to 45ms when there is a message to be sent or received between the master and slave nodes.

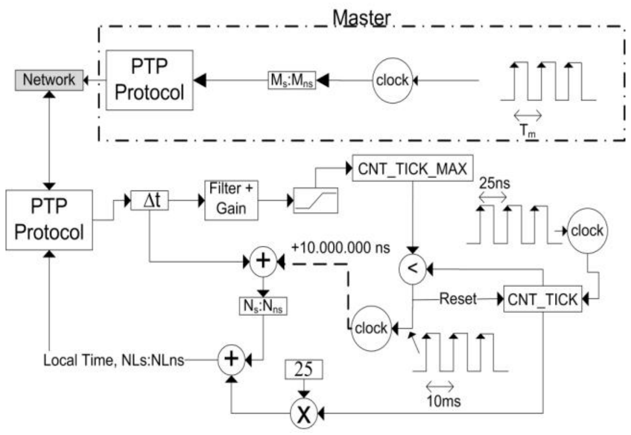


Fig. 7 – Diagram to update local Slave clock with Master local clock information, using PTP protocol

3.4 Simulation results

This section presents the results of Matlab simulations. The data considered for the simulation is: the master clock period, that is below to 1ns; and in the slave node the frequency clock is f=40MHz (25ns period) where only the variables Ns:Nns and the value of CNT_TICK_MAX can be changed. The most important aspect in the simulation is to consider a shift of 5,263% in the 40MHz frequency in the slave, which introduces the need to change CNT_TICK_MAX parameter using the PLL shown in Fig. 7.

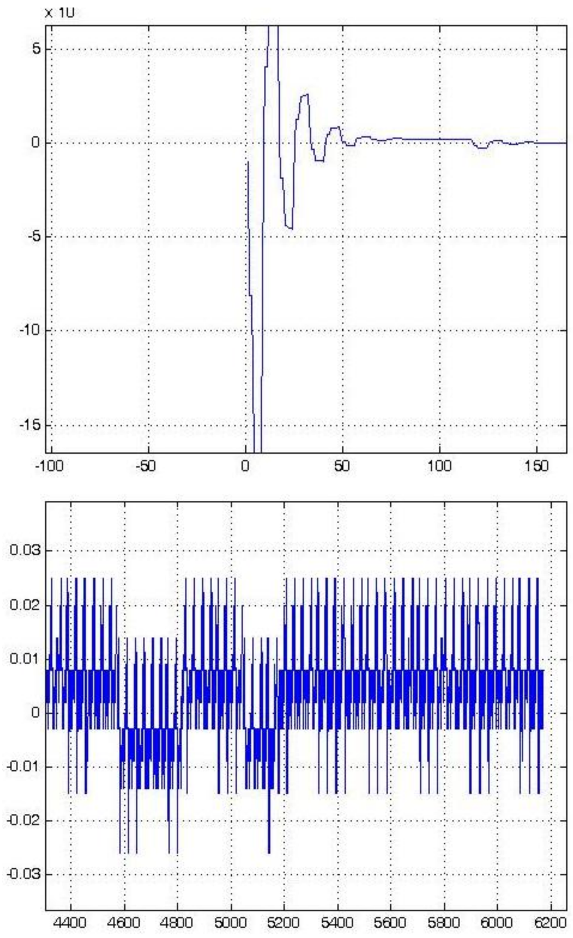


Fig. 8 – Experience 1: Difference between the Master and Slave clocks (microseconds). Top: initial moments; Bottom: end instant.

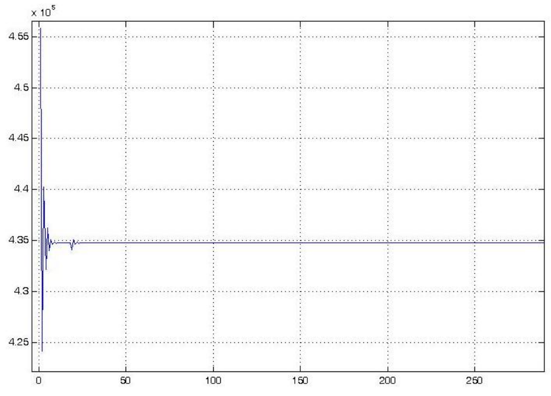


Fig. 9 – Experience 1: Evolution of the CNT_TICK_MAX value.

Figures 8 and 9 outline the simulation results for the PTP functioning duration through large seconds. It can be observed that a convergence of the clocks exists and, after 50 timing messages, at the end, occurs a maximum deviation of 50 ns. In this experiment it was considered two decimal accuracy places in parameter CNT_TICK_MAX; its final value was estimated at 434782.61 for a real value of 434782.608696. In the end, the clock slave was late in 8 ns. *Delay Request* messages are sent randomly between 2 and 30 TSync.

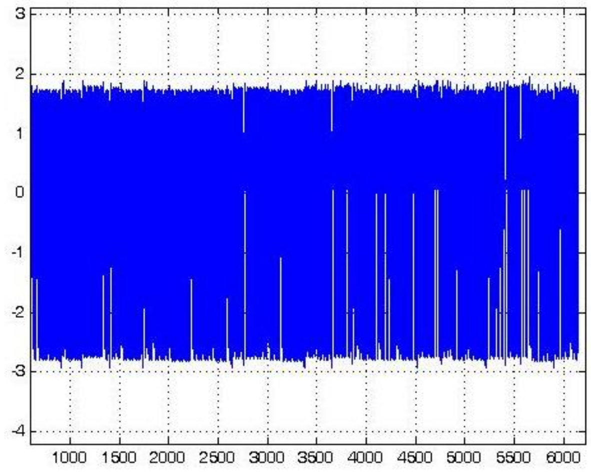


Fig. 10 – Experience 2: Difference between the Master and Slave clocks (microseconds)

Fig. 10 shows the same simulation changing the precision of parameter CNT_TICK_MAX to zero decimal places and, at the end, it was estimated the value of 434783.0. The clock slave was, after simulation, late of 1783 ns. The final error was about 4000ns.

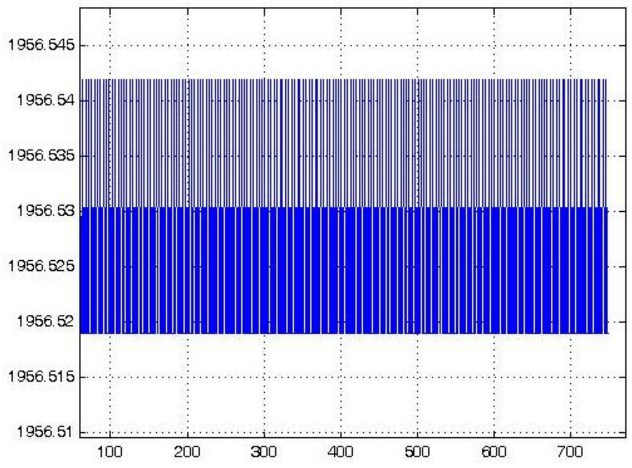


Fig. 11 – Experience 3: Difference between Master and Slave clocks (microseconds)

Fig. 11 shows the simulation results for 2 decimal precision places on CNT_TICK_MAX (similar to Experience 1), but the Slave only sends one *Delay Request* message in the beginning. The result is a constant absolute error in the difference between clocks. The precision is equal to experience 1, ie, about 50 ns; CNT_TICK_MAX was 434782.61.

4 Conclusion

This paper describes an integrated computer system based on different components. The system is based on special firmware to acquire relevant data, being the integration trough Octave and R algorithms and SMIT database system relevant for this purpose.

The basic structure of the system uses data collecting devices connected through a CAN network. A PTP protocol is used to implement a set of control techniques in order to achieve clock synchronization. A simulation in Matlab shows a mechanism to maintain the synchronisation of the clock variables in microcontrollers, being possible to conclude that: a closed control loop is necessary; *Sync* messages are fundamental as *Delay Request* messages; the time period of *Delay Request* messages does not matter, they just need to be sent from time to time; a high precision master clock is essential; the precision of two decimal places is necessary to CNT_TICK_MAX in order to achieve good results.

References:

[1] Inácio Fonseca, J. Torres Farinha and F. Maciel Barbosa, *A Computer System for Predictive Maintenance of Wind Generators*. Proceedings of the 12th WSEAS International Conference on COMPUTERS, Heraklion, Greece, July 23-25, 2008. Pp 928-933. ISSN: 1790-5109. ISBN: 978-960-6766-85-5.

[2] Inácio Fonseca, Torres Farinha, F. P. Maciel Barbosa, *On-Condition Maintenance for Wind Turbines*, IEEE, PowerTech 2009.

[3] Online in: www.slackware.com; www.freebsd.org; www.postgresql.org; www.apache.org; www.php.net.

[4] Online in: www.freepascal.org; www.lazarus.freepascal.org.

[5] Kendall Correll, Nick Barendt e Michael Branicky. *Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol*.

[6] J. Torres Farinha. *An Terologic approach of Hospital Equipment Maintenance*. Phd Thesis, Faculdade de Engenharia da Universidade do Porto, Portugal, 1994.

[7] Microchip: www.microchip.com

[8] Timothy G. Gutowski Joseph T. Foley. *Turbsim: Reliability-based wind turbine simulator*, available online.

[9] Gotenborg, P.Caselitz, *Advanced Condition Monitoring System for Wind Energy Converters*, 1999

[10] A. Hameed et Al, *Condition monitoring and fault detection of wind turbines and related algorithms: A review*, Renew Sustain Energy Rev (2007).

[11] Scheffer, Cornelius; Girdhar, Paresh; 2004, *Practical Machinery Vibration Analysis and Predictive Maintenance*; Elsevier; ISBN 0-7506-6275-1.

[12] *Advanced Maintenance and Repair for Offshore wind farms using fault prediction and Condition Monitoring Techniques*, E.U. final report of project NNE5/2001/710.

[13] Octave and R softwares, www.gnu.org/software/octave/ and www.r-project.org/

[14] Cauwenbergh, Poggio, *Incremental and Decremental Support Vector Machine Learning*, Article, 2001

[15] Kecman, V, *Learning and Soft Computing*, MIT Press, Cambridge, MA. 2001.

[16] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J., Least Squares *Support Vector Machines*, World Scientific, Singapore, 2002.

[17] Scholkopf, B., Smola, A.J., *Learning with Kernels*, MIT Press, Cambridge, MA. 2002.

[18] Manabu GOUKO and Koji ITO, *An Action Generation Model Using Time Series Prediction*, Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

[19] Ricardo de A. Araújo, et all, *An Evolutionary Morphological-Rank-Linear Approach for Time Series Prediction*, 2007 IEEE Congress on Evolutionary Computation (CEC 2007)

[20] T. Rothenberg, *Univariate Time Series Models*, Econ 241b, Fall 2005.

[21] Anna Mikusheva, course materials for 14.384 *Time Series Analysis*, Fall 2007, MIT OpenCourseWare (http://ocw.mit.edu), Massachusetts Institute of Technology. Downloaded on [20-08-2008].

[22] Numerical Recipes, *Introduction to Series Analysis*.

[23] Mr. K Koteswara Rao et all, *Statistical Prediction Based on Estimation of Conditional Density*, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008.

[24] SNTP:www.cis.udel.edu/~mills/database/rfc/rfc4330.txt

[25] PTP:en.wikipedia.org/wiki/Precision_Time_Protocol#IEEE_1588-2008, and an implementation of PTPD in http://ptpd.sourceforge.net/