Experience gained with the use of a high rate data logger in wind turbines

José San Leandro Ros
Director, Automated Computing Machinery SL, +34605876615
E-mail: Jose.sanleandro@acm-sl.com, www.acm-sl.com

***Abstract* – SCADA data in wind farms are normally not enough to discover the inner behaviour of all the subsystems in a wind turbine. This paper shows what we learned in the design of a specific data logger to collect data in the field for long periods (months) and their processing.**

The data gathering had time resolution enough to characterize the short term variations of the wind and its relation to rotational speed in the generator axis, the energy produced, pitch evolution, etc.

This experience was used to find real functional parameters for each of the subsystems (pitch/hub, brakes, nacelle's orientation, etc) of a stall machine with dual generator. Those parameters were key for the development of a Simulator/Trainer for that wind turbine.

It is also the first step for the design of a complete condition monitoring system specialized for wind turbines.

***Index Terms* – wind turbine, data logger, condition monitoring.**

I. INTRODUCTION

The detailed evolution of the signals internal to a wind turbine is processed by its controller (some times PLC based) in order to fulfil the main requirement: to produce the maximum of energy for the available wind. With this target, every wind turbine is designed as an autonomous system. The SCADA system of the wind farm has more a function of data logging the reported behaviour of the wind turbines, rather than an actual control tool for the wind turbines. The SCADA is used to grant "permission to operate", as the final decision for connecting to the grid is solely, to reorient the nacelle to follow the wind direction, etc, are the responsibility of the wind turbine's controller.

Therefore it is understandable that these controllers do not transmit the whole bunch of the evolution of the internal signals to the SCADA: that is a job in itself, of similar demand as the control.

The final result is that the data gathered in a standard wind farm SCADA are not the source of information to understand the internal behaviour of the wind turbine. We needed to know this behaviour in order to create a Simulator/Trainer, that would help in a number of tasks both at the production stage (functional testing of subsystems of the wind turbine) as well as for learning how to operate a wind turbine (without going to the field) or for checking the influence of each parameter (hundreds of them) introduced in the controller.

II. THE CONDITIONS

In the case we were studying (BONUS now SIEMENS 1300KW), the wind turbine has a controller with two CPU's (bottom and top, separated about 70 m.) each one located near their respective connected signals: at ground level and at nacelle level. They are connected via a fibre optic channel.

III. THE PLATFORM WE USED

A. Hardware for data logging

Although the market is full

We designed a data logger around an industrial grade, single CPU computer: that warranties the solution to a key issue, the use of a single clock reference for all measurement's time-stamps. Data were collected through RS485 lines. These lines serve as backbones for a number of off-the-shelf converting devices. Finally we designed adaptation circuits for each signal to those devices, and that fit in the available space left. We developed also an industrial grade tachometer with two particular characteristics: a) high resolution (i.e., in production there is a big power difference between 1500.1 rpm and 1500.4, so we needed a resolution better that 0.1 in 1500 or 1 in 15.000); b) a measurement per generator cycle (i.e., no integrating device), as we could not find one in the market with these characteristics and small enough to fit in the available space.

The link between top and bottom converting devices was done using a separated fibre optic channel: that solved the problem of locating the converting devices very near to the signal's origin, also a key issue due to the high EMI existing inside the tower.

For the electrical measurements we choose an off-the-shelf calibrated four-quadrants tree phases device.

In order to make alarm reports, there is a GPRS modem that allows: a) sending SMS messages; b) be a low speed communication channel for configuration (adquisition scheduling channel and general status verification).

B. Software for data logging

The data logger itself is running on a Linux derivative, with a particular threads management library. The setup is designed so that the data logging process starts automatically after power-up. The control of the data logger is done

through a Windows based program, executing on a separate portable PC, using a specific GUI interface (fig 1).

Due to installation environment and the fact that we planned to leave it working in the field for a long period of time (months) we were forced to solve problems related: a) detection of broken RS485 lines; b) detection of broken converting devices; c) self connecting broken RS485 lines once operational; d) self connecting converting devices once operational.

We designed the software so that it needs no programming, only setup through a simple XML file. Through this file we can set the baud rate of each line, the name of the signals and the sampling period for groups of signals, the type of signal, etc.

The data are stored in disk. The algorithm used for storing information considered several issues:

1. Store only the changes

This technique is common to reduce the size of the files, especially for digital signals, but implies that every sample of every signal, or groups of signals, must be stored with the absolute time-stamp. That potentially implies a size increase for each sample, to overcome that we used a way of coding the full time-stamp (year-month-day_hour:minute:second.millisecond) with resolution better than tenths of millisecond in just four bytes.

2. Splitting the data files

Because we planned to leave the data logger unattended for a long period of time, we came to the idea of splitting the data gathered among several time-contiguous files, so that in case of problems with the disc occur, most part of the information could be recovered. Data files were named after the first time-stamp inside it, so they can be easily processed afterwards.

3. Data format

We decided to store the data in XML format as this format shows clearly the data in a readable form.

4. On line compression

A low priority process, running in background, monitors when a data file close, and make a compressed version of it, using a standard algorithm (bzip), to counteract the implication of using XML format.

IV. Limitations

The plan was for data logging every signal that arrives to or leaves from the controller. Among them, the fastest are related to the thyristor gate control. We had developed previously a specific data logger for these signals, that also monitored the voltages and currents of the grid. In the case of the wind turbine we were studying, these (6) signals consisted of pulse trains of 5 ms with each pulse. We decided that these signals did not added information for the purposes of our study, so we did not logged them. We logged, though, the conduction angle for these signals.

In total we have 46 analog and 63 digital signals. We made tests for speed of the overall system and found that the logger was able to cope with sampling periods of 300 milliseconds. After studying samples with this sampling period we decided that a good compromise between time resolution and meaningful data was around 1000 milliseconds. We found also that the digital signals time-stamp uncertainty was below 50 milliseconds.

V. Operation

As said previously, the software of the data logger starts up automatically after the operating system boots, and starts data logging using the specified configuration file. Downloading of the data files is done through ftp, without stopping the data logging process.

VI. Data exploitation

In order to characterize the data gathered, we decided to design an easy to use tool, able to present graphically big amounts of data, and to present them so that we could understand and, ultimately, learn how the evolution of the dependent signals was. For the exploitation, we choose to pass the data to a standard SQL data base. Typically, for a month period, the compressed data out of the data logger is below 100 Mbytes (109 signals, 1 second resolution for analog, time stamp with milliseconds resolution) , that transforms to around 1 Gbytes in data base format, including indexing of the time-stamps.

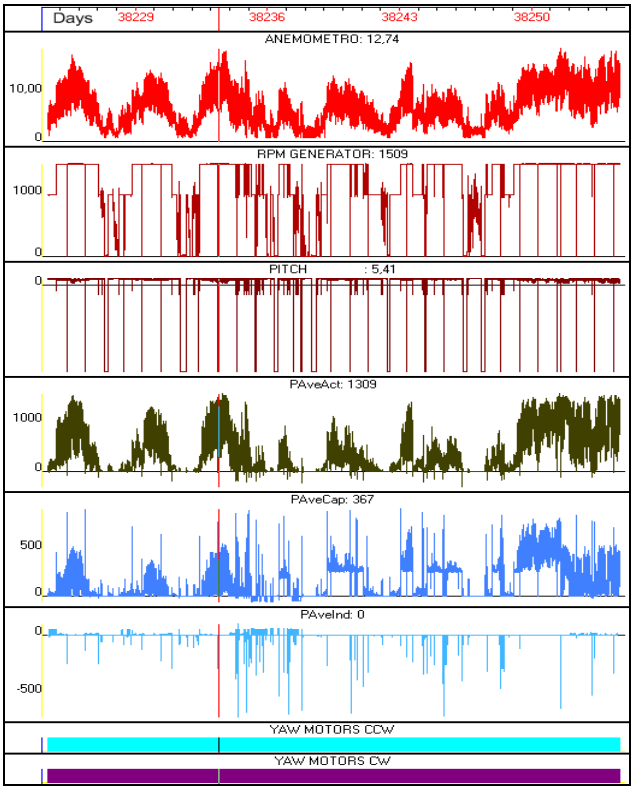

Figure 1

Figure 1 shows the wind speed, rpm (generator), pitch angle, active power, capacitive power and inductive power for this period (30 days).

Note: in the figures the signals is located on top of the oscilloscope channel. Values near to them are the instantaneous value at the cursor (red vertical line). ANEMOMETRO : *Wind Speed*; PAveAct: *Total Active Power (kW)*; PAveCap : *Total Capacitive Power* (kW); PAveInd: *Total Inductive Power* (kW).

Figure 2 shows the evolution of generated currents, compared to evolution of wind and pitch control.

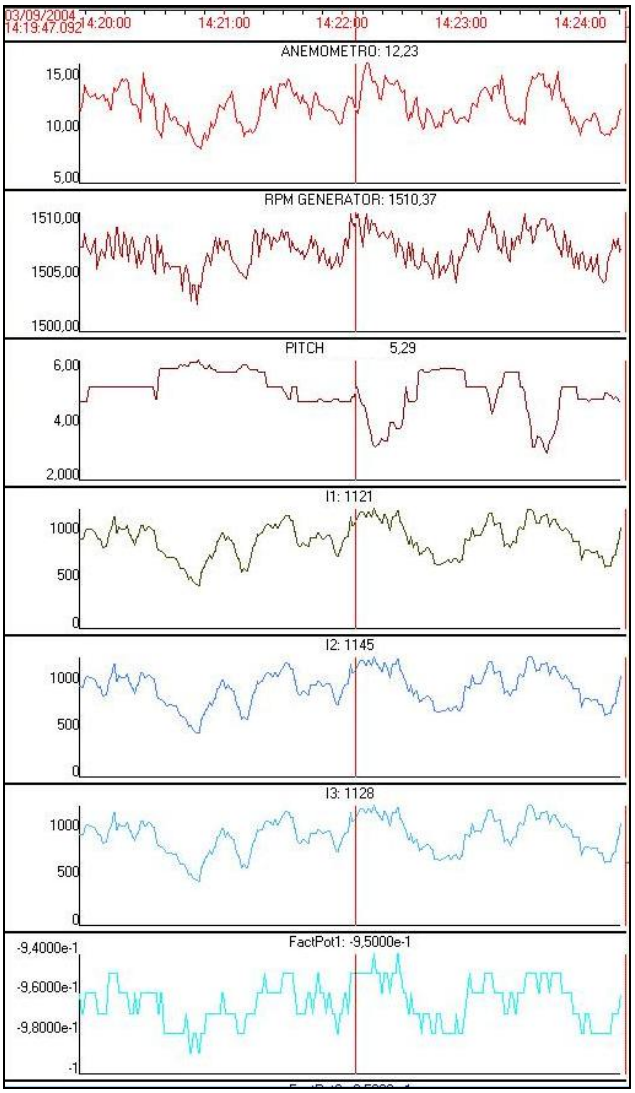

Figure 2

Figure 3 shows the run up period, starting after brakes are liberated (red line), the pitch evolution controlling this run up, till cutin, with the generation of initial capacitive (generation) and inductive powers (consumption). With wind speed around 12 m/s, the whole operation takes around 100 seconds.

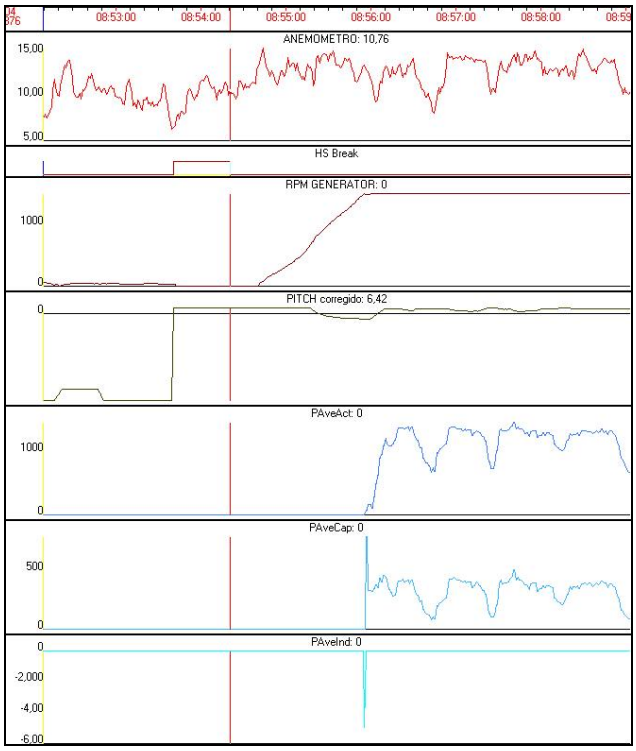

Figure 3

In important behaviour is shown in figure 4: after the speed of rotor arrives to around 800 rpm (aprox 60 seconds after the start of the run up), the controller reduces the pitch value to get out of the exponentially growing speed area, and keep a linear approach till the cutin region.

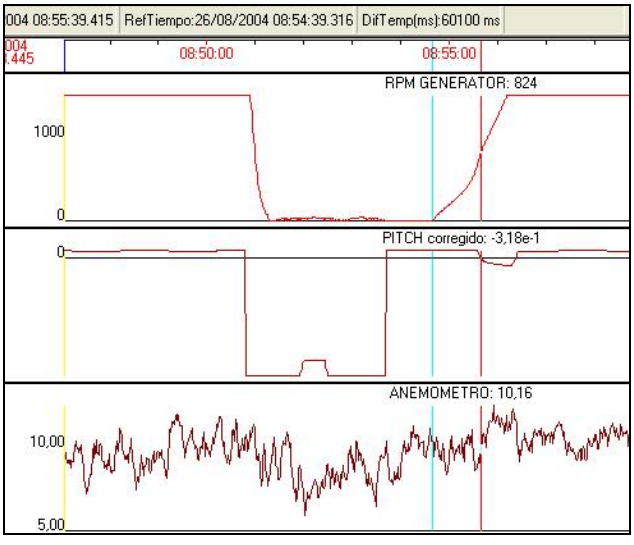

Figure 4

Approaching the cutin region the control reduces the slope for a softer connection to the grid. That can be seen in the next figure, together with the detailed evolution of pow-

ers (active, capacitive, inductive) immediately after the cutin.

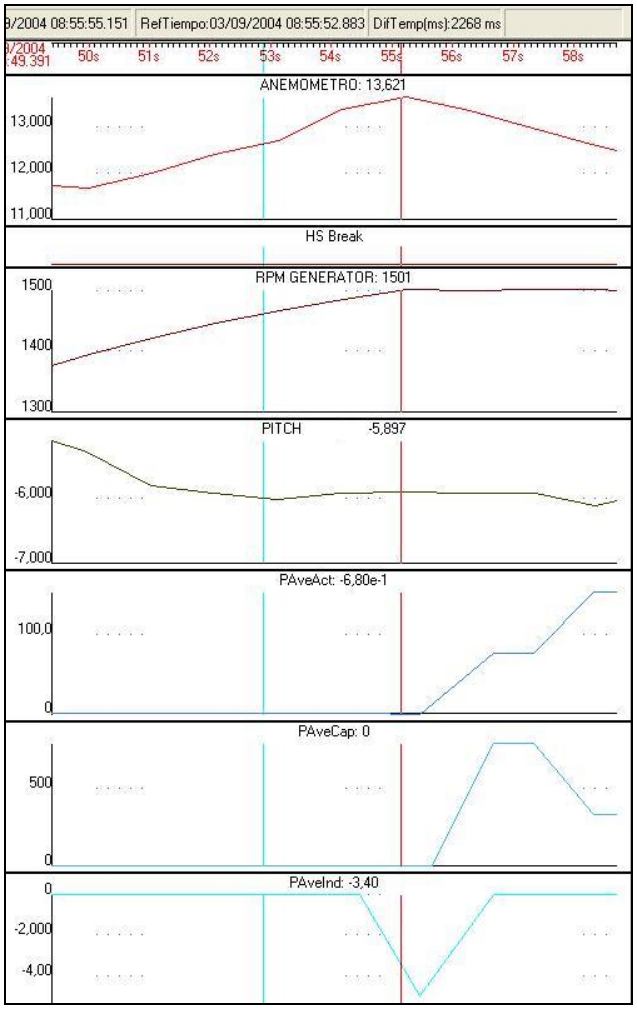

Figure 5

Another interesting operation is the change of the generator operating configuration (4 to 6 poles). Figure 6 shows the effectiveness of this approach (use to squeeze the maximum of the wind energy at low wind speeds). The pitch is controlled, first to slow down the rotor speed, just a little below the target speed (1.000) and then to make a bottom-up approach as a normal run up. These pictures can be used to properly choose the run-up, run-down stages as to minimize them while incurring a minimum mechanical stress in the wind turbine's elements.

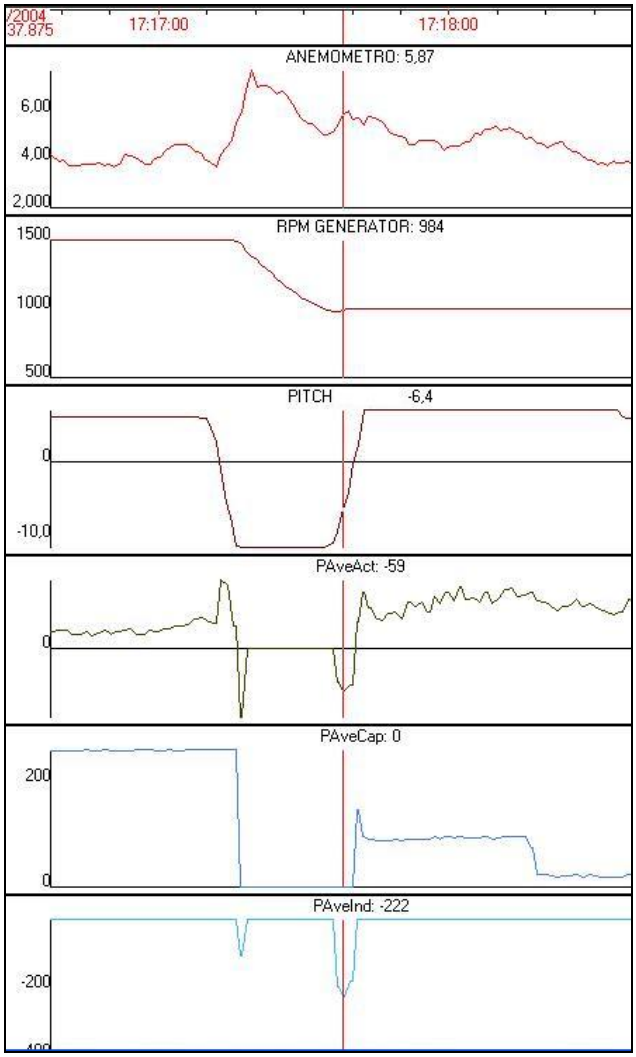

Figure 6

Figure 7 shows the evolution of the powers (active, capacitive and inductive), the power factor (centered around 1, bigger than 1 capacitive, lesser than 1 inductive), and how the correction capacitors banks were controlled.

Legend: there are five capacitor banks name "BANCO CONDEN 1" to "BANCO CONDEN 5". Negative logic.

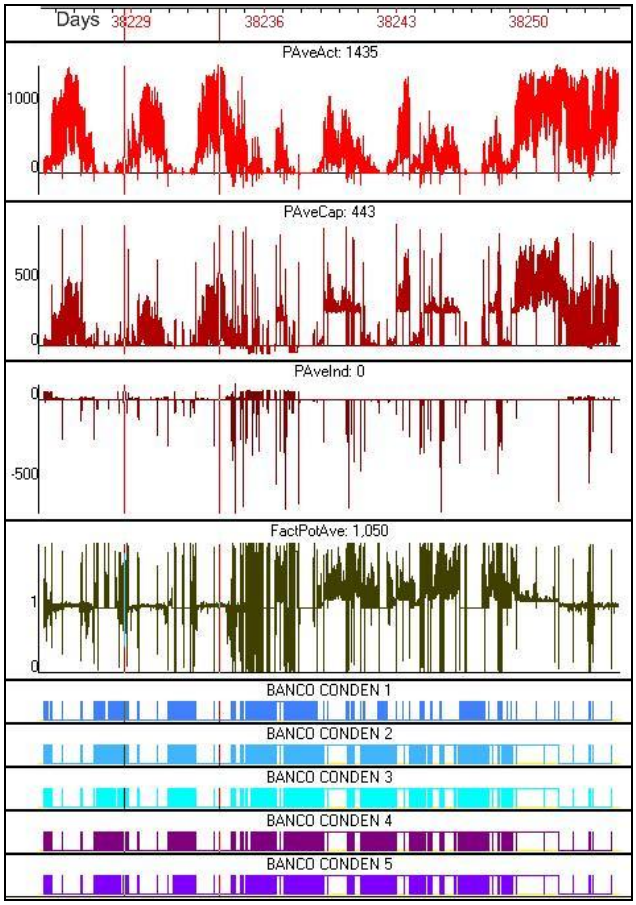

Figure 7

VII. CONCLUSIONS

1. *For the purpose of analysing the behaviour of wind turbines, sampling rates of around 1 second is enough. SCADA data is not enough to understand the internal behaviour of wind turbines.*
2. *Typical lab instruments are not suitable, in general, due to number of signals to monitor, space, connections layout and environmental constraints.*
3. *High EMI and topology considerations force a special design on how de measurement units are connected to the signals sources (as near as possible).*
4. *Despite of these constrains, we constructed and deployed a data logger that logs all the wind turbine's internal signals , using low cost specialized measuring units, installable by the maintenance personnel at the wind farm.*
5. *Detailed analysis of the log data shows a number of interesting points, such as: a) rapid changes in the electrical load that, most probably, create high mechanical stress in components (gearbox); b) the complex policy of the wind turbine's controller regarding the reactive compensation capacitor's banks; c) the study of the controller's setup parameters that governs the run-up and run-down evolution times; etc.*
6. *One unforeseen application for this data logging is for discovering intermittent problems in devices, connections, etc.*