International Journal of Engineering & Technology, 4 (1) (2015) 78-89
www.sciencepubco.com/index.php/IJET
©Science Publishing Corporation
doi: 10.14419/ijet.v4i1.4154
Research Paper

Smart metering application for power efficiency studies

Cristian Tudoran [1]*, Stefan Albert [1], Dorin N. Dadarlat [1], Carmen Tripon [1], Sorin Dan Anghel [2]

[1] *National Institute for Research and Development of Isotopic and Molecular Technologies, Cluj-Napoca, Romania*
[2] *Babes-Bolyai University, Faculty of Physics, Cluj-Napoca*
**Corresponding author E-mail: ctudoran@itim-cj.ro*

Copyright © 2015 Cristian Tudoran et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Improving the energy efficiency of our Institute's data center is an ambitious challenge for our research teams. Understanding how the energy is consumed in each segment of the system becomes fundamental in order to minimize the overall energy consumed by the system itself. In this paper, we propose an experimentally–driven approach to develop a simple and accurate power consumption and temperature monitoring system. In this work we focused our attention on the monitoring, measurement of the energy consumption patterns of our data center system, at INCDTIM Cluj-Napoca, Romania.

Keywords: *Control Design; Control Engineering Computing; Energy Consumption; Software Design.*

1. Introduction

In the last decade, "Green Networking" [1] has gained considerable importance for both commercial entities and researcher institutes that aim at understanding and reducing the energy consumption of computing and communication infrastructures. Several studies have shown that the computer network sector accounts for 2…7% of the global energy consumption [2] and it is also responsible for 2…3% of total emissions of CO_2. Moreover, it is important to remark that about 50% of the total energy used in the computing sector is consumed by the air conditioning systems [3]. Therefore, network operators and service providers currently compete to optimize the energy efficiency of their computer infrastructure in order to reduce both CO_2 emissions and operational costs.
The main objective of this work is to experimentally measure the energy consumption patterns of our data center, both at network level and at the air conditioning system. The data collected during the power measurements will help us gain an increased insight into the system's behavior, paving the way to the development of (i) realistic models for power consumption in computer networks and (ii) protocols and algorithms for decreasing the overall energy consumption of the system.

2. Smart metering software

In this section, we present the smart metering application that was written by the authors – a data logger software which records and presents in a graphic mode, in real time, the important data (electric parameters and the temperatures) used by us to perform power consumption diagnostics at the data center at INCDTIM Cluj-Napoca [5, 6].
The study of electricity consumption was performed using the information provided by a digital power meter type UPT-210, manufactured by Algodue Elettronica – Italy [7], and the temperatures used in the study were measured using a microcontroller-based module designed and built in INCDTIM.
The block diagram of the studied system is presented in Fig. 1. As you can observe from Fig. 1, we use two different measuring devices which send data to the data logger software on two different ports: the digital power meter sends the data using the serial port, and the microcontroller thermometer sends the data on the Ethernet port. The commercial data logging applications only communicate on a single port, usually the Ethernet, USB or serial port [8, 9, 10]. We could have used two applications running in the same time, one for each instrument, but the ideea was to gather the

information from the two different measuring devices in a unified file, for further analysis, without the need to use data conversion utilities or two or more applications. This is the main reason why we decided to write our own data logger application.

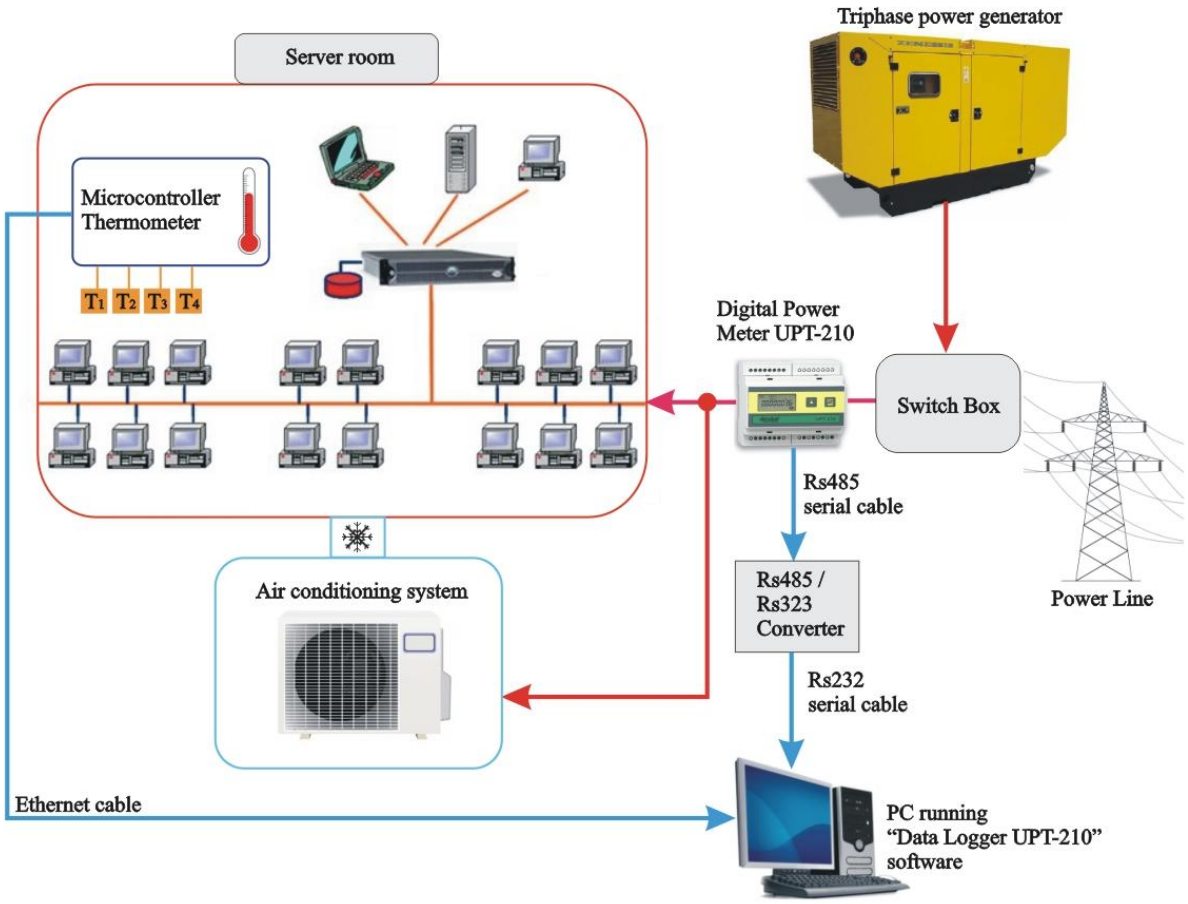


Fig. 1: The Block Diagram of the Experimental System.

The smart metering application called "Data Logger UPT210" was developed in order to conduct a comprehensive energy consumption study of the grid network at the Computing Center, INCDTIM Cluj-Napoca, in order to obtain a complex algorithm that will allow us to considerably reduce the electricity consumption. Although there are available data logging applications [8], [9], [10], we couldn't find a data-logger that can communicate with two different measuring devices on two different ports, record and simultaneously display the data in real time (we record and display 30 different parameters in real time). Another issue that made us writes our own software was the necessity to display and analyze sets of data with very small variations. The small variations cannot be observed using the usual plotting programs, like MS Excel, for example. This is how the idea of "Auto Zoom" plotting was born (please see paragraph 2.3). Our energy economy plan is based on two concepts:

1) The power consumption of a modern computing system is situated at approx. 60% of its maximum value, even when the system is in idle mode – thus, stopping the systems that do not perform calculations or functions in that moment, can significantly reduce the total electricity consumption of the network.
2) The computer systems in the network are operating in a controlled environment in which the internal temperature is maintained at a predetermined value regardless of changes in the outside temperature. The power consumption of professional air conditioning installations is in the order of tens of kW per unit [6]. By switching the operation of the air conditioning systems as a function of the outside temperature and the load of the computer network (the amount of dissipated heat), we can achieve a considerable reduction in the consumption of electricity.

The "Data Logger UPT210" software records and graphically displays the captured data from the two mentioned devices, allowing real-time tracking of both parameters: the power consumption / electrical parameters and temperature values, chosen for study. The selected temperatures are the following: outside ambient temperature, water temperature at the outlet of the main air conditioning compressor system, and two measured temperatures in the server room, called "GRID site rack temperature" and " HPC cluster rack temperature." The user interface of the application "Data Logger UPT210" consists of a main window that provides all the information sought in the survey (electrical parameters and temperatures) and the settings necessary for the study (communication parameters with the measuring equipment, information about recorded data and operation of the program). Fig. 2 shows a screenshot of the application interface.

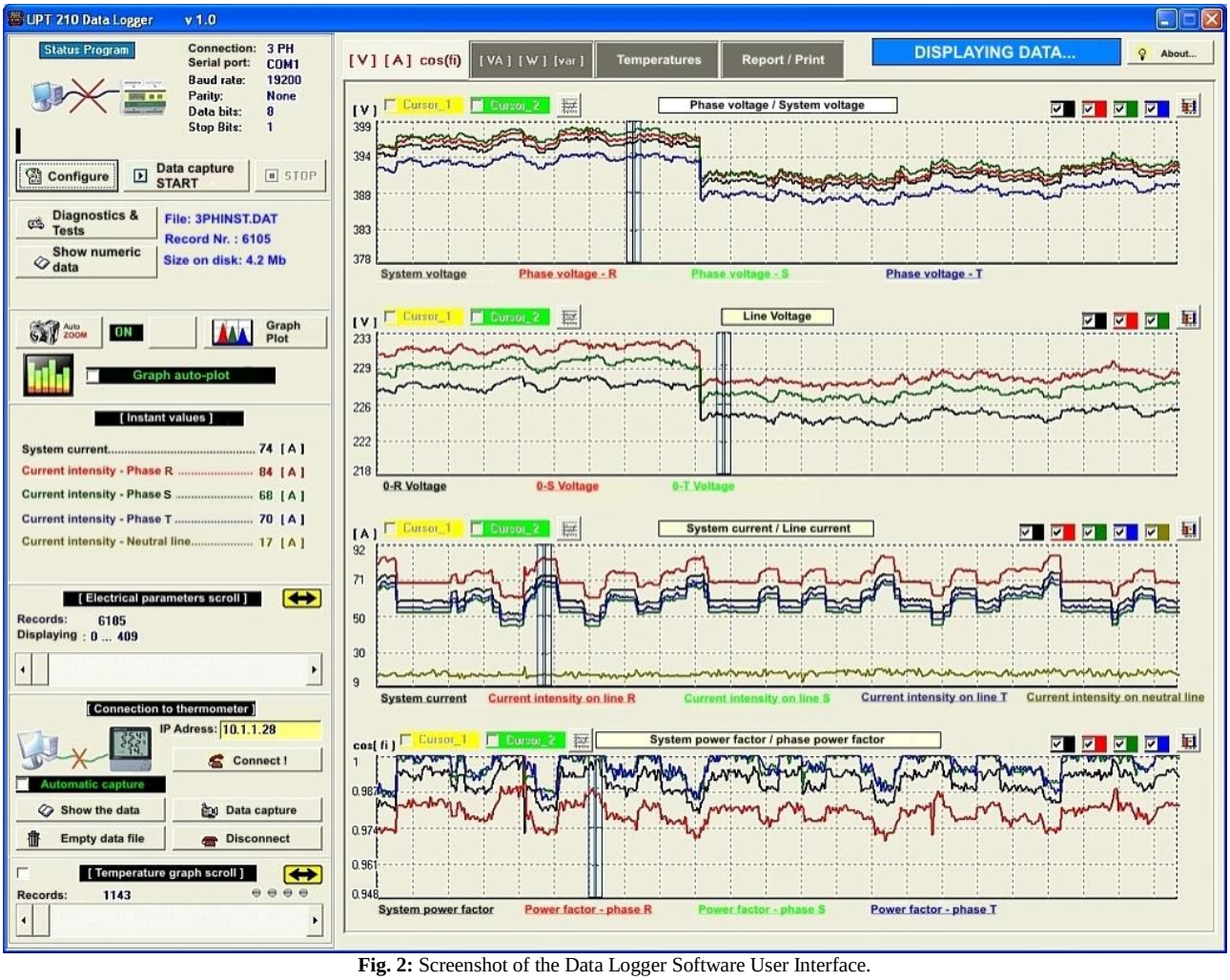


Fig. 2: Screenshot of the Data Logger Software User Interface.

As it can be seen from Fig.2, the application provides an interactive and user friendly interface, which contains the necessary tools for studying the electrical parameters and the temperatures: sliders that allow the user to display the instantaneous values measured directly from the displayed graphs, a report generator that allows the user to select a desired time domain and graphically display the selected area in a magnified fashion.

The main application window is divided into two panels: on the left panel are shown the communication parameters with the measurement equipment, and on the right panel are graphically displayed in real time, the following physical parameters: line voltage, phase voltage, rated current, power factor, apparent power, active and reactive power for each phase and four recorded temperatures ("GRID site rack temperature", "HPC cluster rack temperature", output temperature of the air conditioning system compressor, and outside ambient temperature).

In order to help with the study of instantaneous measured values, the application contains a set of interactive cursors by which the user can "read" directly on the graphs, the instantaneous values and time moments at which they were recorded.

The movement of an interactive cursor on the graph has the effect of displaying the measured values corresponding to its position on the chart. Fig.3 shows an example of using an interactive cursor on the "current intensity plot" on each phase. The corresponding values are displayed at the cursor's position. The application can generate a report when the user whishes to study the selected values in a specific time interval. Fig.4 shows an example of the chart generated for the current intensity, using the "Autozoom" function of the application. For additional analysis of the recorded data, the application allows the exporting of the numeric data, as columns in a standard text file. The data can then be imported by other applications (Excel, Origin, Matlab, MathCAD, etc).

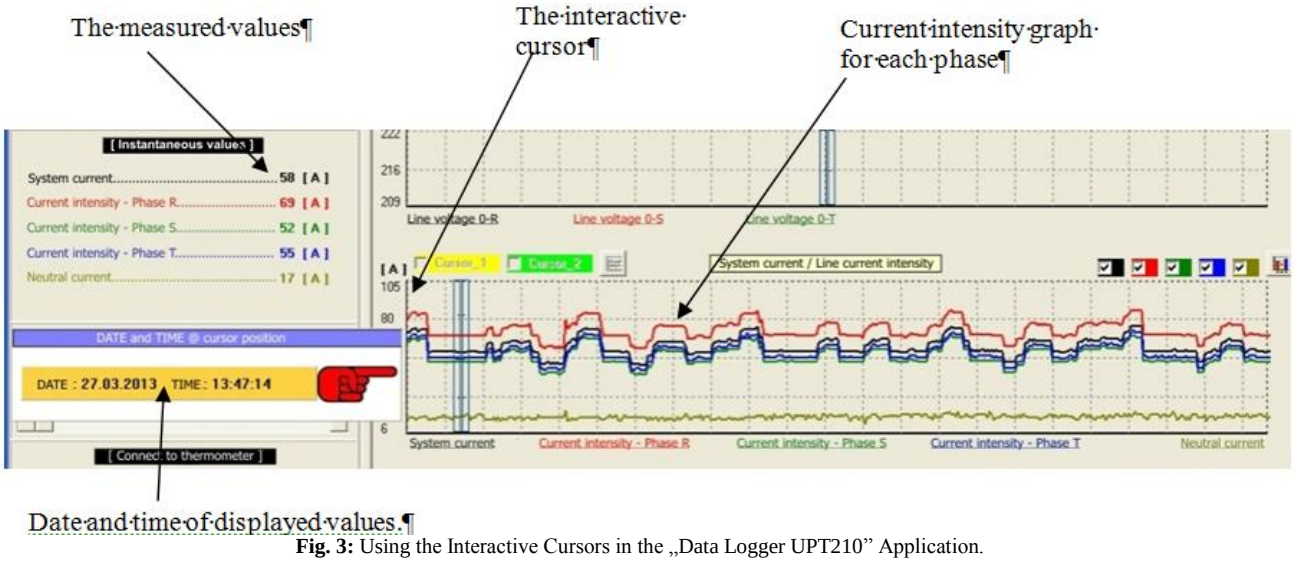


Fig. 3: Using the Interactive Cursors in the „Data Logger UPT210" Application.

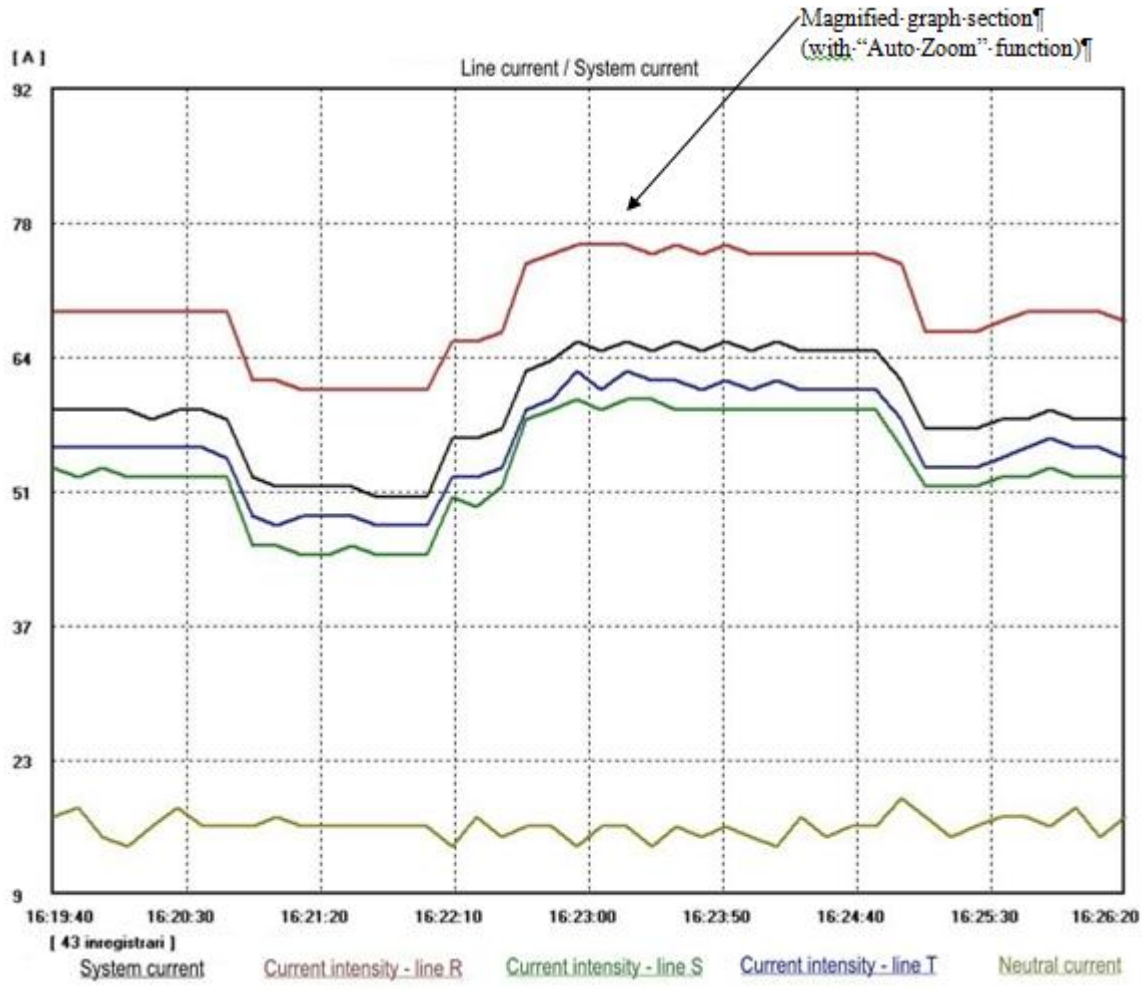


Fig. 4: Graph-Report Generated with Our Application, in „Autozoom" Mode.

2.1. The application's architecture

Our data logger application records the data from the measuring instruments (digital power meter and microcontroller thermometer) using the following communication diagram:

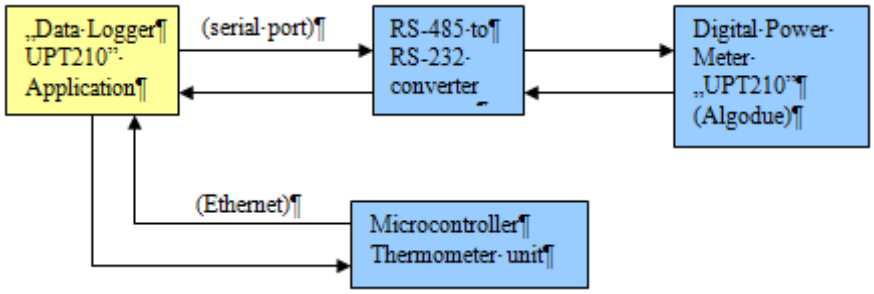


Fig. 5: The Communication Diagram for the „Data Logger UPT210" Application.

As shown in Fig.5, the application communicates simultaneously with two measuring equipments connected to two different ports of the computer: RS-232 serial port (for the digital power meter "UPT210") and the network port (Ethernet) - for the microcontroller thermometer module. The microcontroller thermometer module's description is presented in reference [4]. Fig.6 presents the logical block diagram for the operation of the "Data Logger UPT210" application.

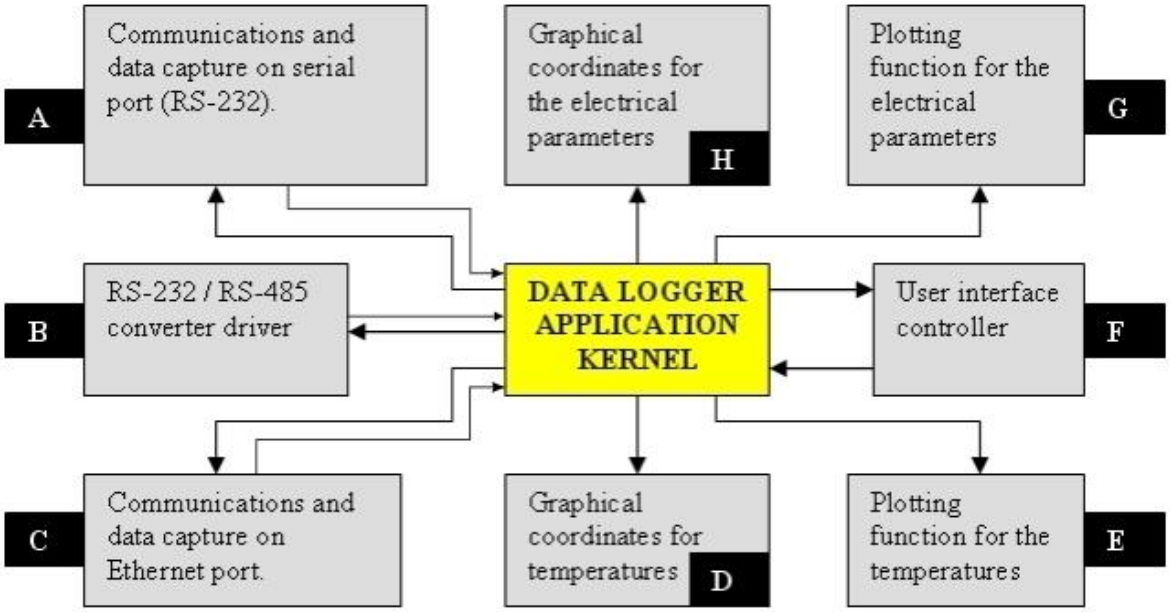


Fig. 6: The Logical Block Diagram for Our Data Logger Application.

The "Data Logger UPT210" application is based on a "central kernel" or „core" which has the function to drive the other software modules: communication and data capture modules (A and C, Fig.6), calculation of coordinates for the graphical display of the measured data (H and D, Fig.6), modules for the graphical displaying of the values (E and G, Fig.6), the command module for the RS-232 / RS-485 converter (B, Fig.6) and the user interface module, based on the interaction of the user's command inputs and the program (F, Fig.6). The application's kernel controls the data capture modules (A and C, Fig.6) periodically with a period set by the user (in our tests, we chose a temperature capture rate of 10 seconds and 2 seconds for the electrical parameters capture rate). The operation of the modules that make up the application will be further described using flowcharts and pseudo-code sequences for a detailed understanding of the internal software mechanisms. The variables associated with the electrical parameters and the recorded temperatures are presented in Table 1.

Table 1: The Variables Associated to the Recorded Parameters.

Recorded parameter	Variable name
The system voltage, phase voltages	UF1, UF2, UF3,Usys
Line voltages	UL1, UL2, UL3
Line current intensities, system current intensity	IL1, IL2, IL3, Isys
Power factor for each phase, system power factor	Cos φ1, Cos φ2, Cos φ3, Cos φ_{sys}
Aparent power on each phase, system aparent power	PAF1, PAF2, PAF3, PAsys
Active power on each phase, system active power	PACTF1, PACTF2, PACTF3, PACTsys
Reactive power for each phase, system reactive power	PRF1, PRF2, PRF3, PRsys
Recorded temperatures.	T1, T2, T3, T4

At the first start after the installation of the program, the user is required to set the serial communication parameters: data transmission speed (baud rate), parity, data bits, stop bits, serial port name (COM1, COM2, etc.), and the IP address of the microcontroller thermometer, which is connected to the local network. This data is then saved in an initialization file (*.INI) that is read each time the program starts. The application's initialization sequence is presented in Fig. 7.

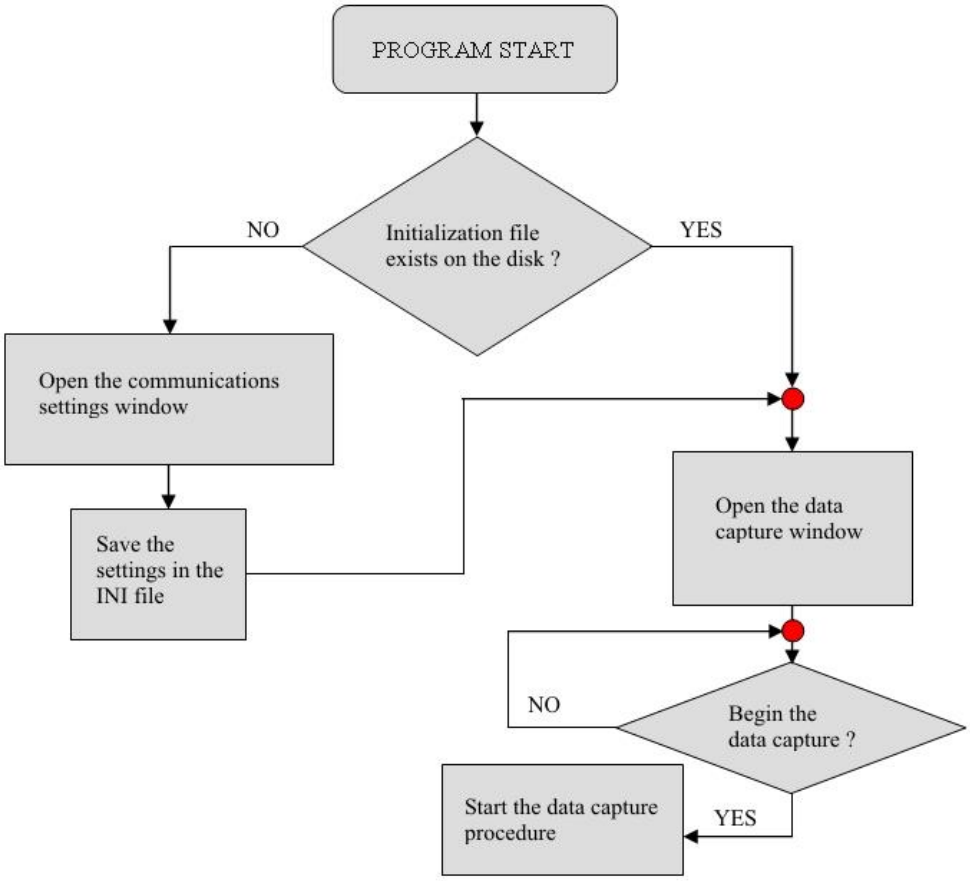


Fig. 7: The Initialization Sequence for the „Data Logger UPT 210" Application.

2.2. The data capture procedure

The data capture procedure includes modules A and C of Fig. 6. This procedure reads the data sent by the digital power meter "UPT210" and the microcontroller module for the temperature measurements, checks their integrity, and in afirmative case, it saves the data in a file on the hard drive.
Both measuring equipments respond to a "data request" type command, sent by the data capture procedure through a series of numeric values separated by separator character #32 (Space). The digital power meter "UPT-210" requires a converter from RS-485 to RS-232, whose operation (in transmit mode (TX) or receive mode (RX)) is controlled by the data capture procedure by using the DT_READY control bit (Data Terminal Ready) on the serial port. The communication on the serial port is based on a simple idea: we declared the serial port as a binary file and we used the WinAPI - WriteFile [11] and ReadFile [12] functions for reading or writing the data:
WriteFile(ComFile, data_string, Length(data_string), BytesWritten, nil);
ReadFile(ComFile, data_string, Length(data_string), BytesRead, nil);
where, ComFile is the binary file attached to the serial port.
The communication on the Ethernet port is performed using the standard functions provided by the WINSOCKET library provided by the Windows operating systems [13]. This library provides the following functions for communication on a network port: Accept, Close, Connect, GetData, Listen, PeekData and SendData. The functional description in pseudo-code for the data capture procedure is given in Appendix 1.

2.3. The procedure for the calculation of graphical coordinates

This procedure reads the numeric values stored in the data files („ELECTRICAL_VALUES.DAT" and „TEMPERATURES.DAT") and calculates the pixel coordinates for plotting the graphs. The calculated values are saved to disk in a dedicated file (called „NUMBERS.DAT"). Coordinates calculation is performed according to the

option chosen by the user: display in "normal mode" or display with "auto-zoom" mode. In the case of "normal" mode of display, the procedure searches for the maximum value for each data (electrical parameter and temperature), and the coordinate in pixels is obtained using equation (1):

$$Y_{pixel} = \frac{\left(N \cdot Y_{MAX}\right)}{M} \quad (1)$$

where: Y_{pixel} is the coordinate in pixels for display on the chart.
N is the numerical value read from the data file
Y_{MAX} is the image height (in pixels) for the chart.
M is the maximum numerical value read from the data file.
Considering the top left corner of the computer screen as the origin (0,0), a correction of the pixel value must be made according to equation (2) before displaying it on the screen:

$$Y_{grafic} = Y_{MAX} - Y_{pixel} \quad (2)$$

where: Y_{grafic} is the final value in pixels for display on the screen and is saved in the file named "NUMBERS.DAT".
For the "auto-zoom" display mode the Y=0 and $Y=Y_{MAX}$ positions correspond to the minimum, respectively, maximum values found in the data file. These values grant the effect of "magnifying glass" or "zoom" on the graph that allows the observation of the smallest variations in the displayed values. The "auto-zoom" mode is particularly useful if the data presents small variations, such as the display of power factor for each phase, for example. Fig.8a shows an example of display in "normal" mode, and Fig.8b shows the same graph displayed using the "auto-zoom" option.

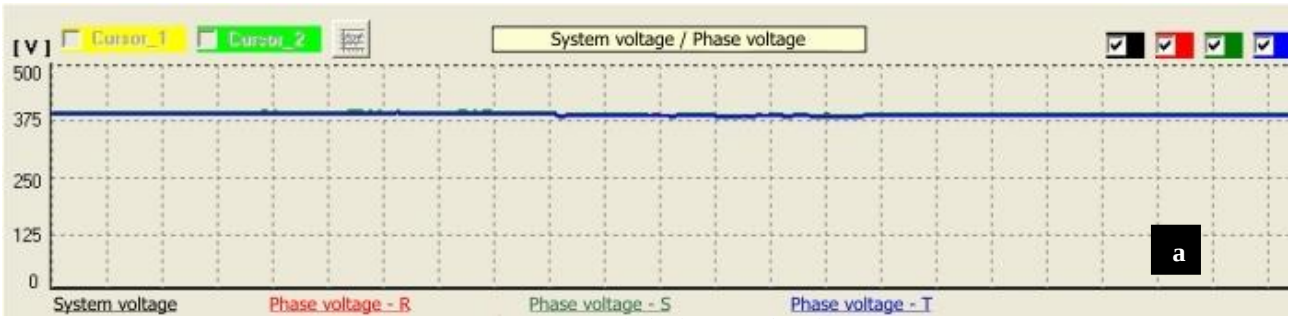


Fig. 8: A. Graphic Display in „Normal Mode".

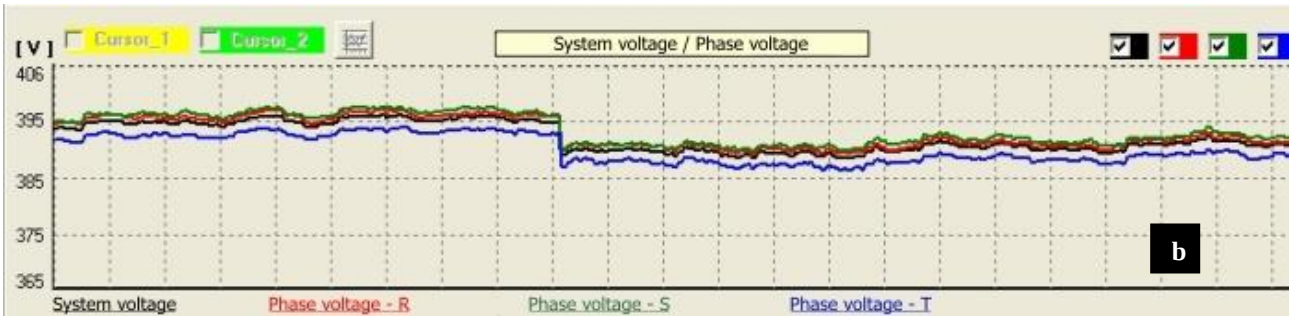


Fig. 8: B. Graphic Display in "Auto-Zoom" Mode.

The calculation of the coordinates for the "auto-zoom" displaying mode is as follows:

- For the physical value chosen, the data file is searched for the minimum value (MIN).
- Subtract the minimum value found (MIN) from all the values stored in the data file.
- Then search for the maximum (MAX) the values saved in the data file.
- For each value in the data file, apply equation (1).
- After applying equation (1), the correction is given by equation (2).

The description of the procedure in pseudo-code language is given in Appendix 2.
The procedure for calculating the coordinates is the same in the case of temperature values. The procedure runs repetitively for each physical quantity that is displayed graphically.

2.4. The procedure for graphic plotting of the recorded data

This procedure takes the calculated values from the coordinate file "NUMBERS.DAT" and draws on the screen each value (electrical parameters and temperature values). The coordinate values were calculated according to the option selected by the user: display with "auto-zoom" or "normally" and because of this, the procedure has no other role than to read the coordinate values from the data file and then draw the graphs.

At the initialization of the procedure, first the graphs grids are drawn and then the values are displayed on the ordinate. Plotting is carried out using standard procedures provided by the operating system: MoveTo(x, y) and Lineto(x, y). If the total number of records in the data file is larger than the width of the graphs (the width of the image), the display is performed using the "scroll" function. In this case, a horizontal scroll bar is activated and the the graphic is displayed in the following interval: (Scrollbar1.pozition - Scrollbar1.min) ... (Scrollbar1.pozition) , where: Scrollbar1.pozition is the function that returns the cursor position of the scroll bar, and Scrollbar1.min is the minimum range of the scroll bar. The size of the data file (the total number of records) is automatically assigned to the Scrollbar1.max value. Any movement of the cursor of the horizontal scroll bar has the effect to redraw the graphics in the range given by equation (3). The operational flowchart of the graphic display procedure is presented in Fig. 9. On the flowchart, in the first decision block, the variable called DFILESIZE (data file size = total number of records) is compared to the numerical value of 800, which is the width of the image in pixels (800 x 600 pixels).

2.5. Operating the user interface

The "Data Logger UPT210" application interface gives the user the following options for easing the research work:

- Interactive sliders for displaying the instantaneous values directly on the graphs
- Markers, for delimiting a desired domain on the graph
- A report generator.
- An option to save an interval of any graph as an image in bitmap format.
- An option to export data in standard text format (data in columns) to be used by another spreadsheet program.

At the functional level, the sliders are bitmap images of size 600 x 15 pixels (height x width) whose background color was defined as "transparent" for the application. These images are translated along the X axis of each graph, using the WinAPI functions that check the movement of the mouse pointer and the state of its buttons: MouseDown (checks if a mouse button was pressed), MouseMove (checks if the pointer was moved on the screen) and Mouseup (checks if the pressed mouse button was released). The interactive cursor movement involves the translation of the associated image on the X axis, following the mouse pointer, when the left button is pressed. The boolean variable associated with the state of the left mouse button was named suggestively "button_pressed". In the program, these functions are described as follows:

In case of the MouseDown procedure:

```
START Procedura MouseDown
IF NOT(button_pressed) THEN button_pressed = TRUE
END
```

The translation of the image (the cursor) on the graph and the displaying of the associated measured values is performed by the following piece of pseudo code:

```
1)  START  Procedure MouseMove
2)  IF (X < 0) THEN X = 0
3)  IF (X > 785) THEN X = 785
4)  cursor_image.pozx = X
5)  Poz_file = (cursor_image.pozx – 7)
6)  IF (display_scroll = TRUE) THEN
7)  Poz_file = Poz_file + (Scrollbar1.position – Scrollbar1.min)
8)  OPEN_FILE ("TEMPERATURES.DAT")
9)  READ_RECORD (Poz_file)
10) WRITE_RECORD.Temperature.T1
11) WRITE_RECORD.Temperature.T2
12) WRITE_ RECORD.Temperature.T3
13) WRITE_ RECORD.Temperature.T4
14) WRITE_RECORD_Time
15) CLOSE_FILE ("TEMPERATURES.DAT")
16) END
```

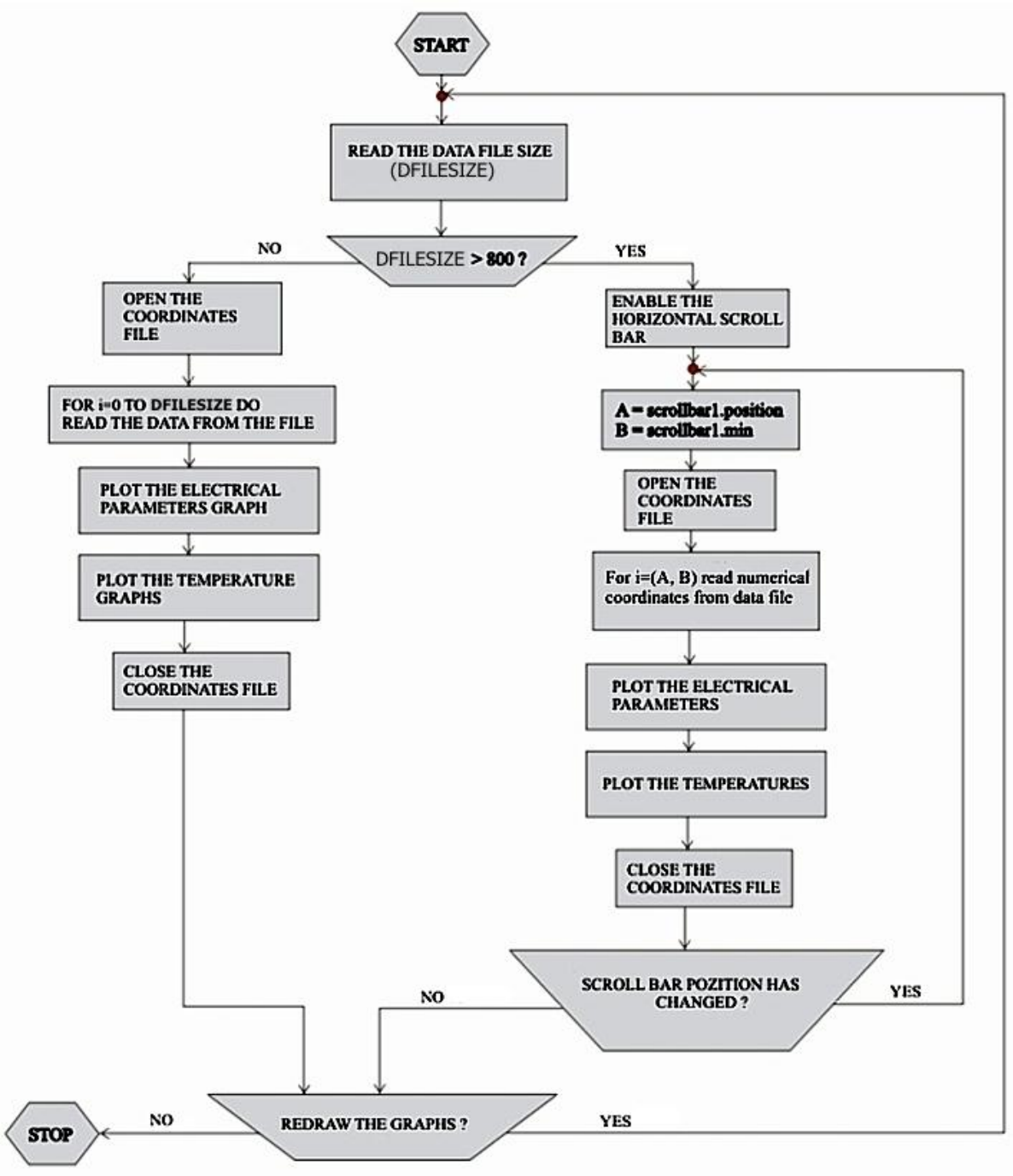


Fig. 9: The Operational Flowchart of the Graphic Display Procedure.

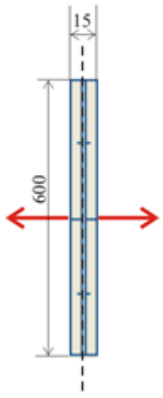


Fig. 10: The Interactive Cursor/Slider

In lines 2 and 3 of the pseudo-code above, we check the position of the X axis of the image associated with the cursor, and if the chart limits are exceeded (width is 800 pixels) , the cursor's position correction is performed (line 4). In this way, the cursor's movement is restricted to the area inside the graph. The address in the data file, associated with the cursor's position is calculated in line 5 of the pseudo-code (as the cursor image has a width of 15 pixels, we took as a reference point its center, thus the actual address will be the position X of the image - (15: 2)). If we have a larger number of records than the width of the graph, the lines 6-7 in pseudo-code perform a position correction in the data file according to the position of the cursor. Each graphic has an option which gives the user the option to define an interval of interest. For this purpose, two different positions of the interactive slider are marked with two markers, directly on the chart (Fig. 11).

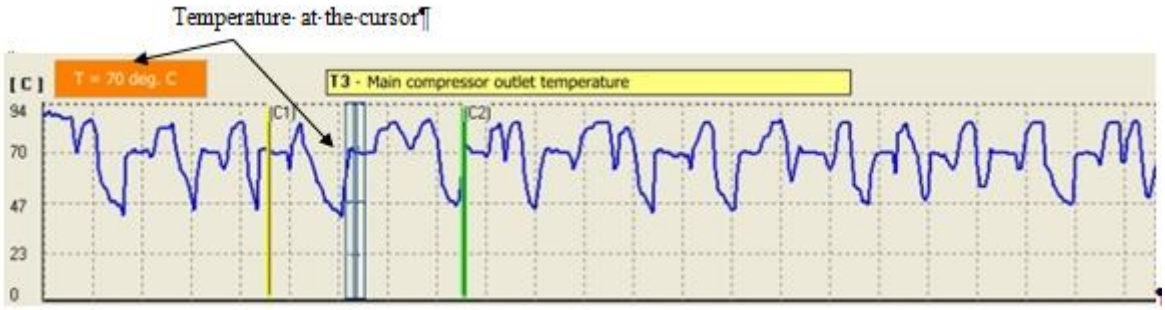


Fig. 11: Using the Interval Delimitation in the Data Logger Application.

3. Conclusion

In this work we presented a modern application that was developed by the authors, which can be used to track in real time all the important electrical parameters and power consumption of a complex system - in our case we studied the power consumption of the local data center.

As our measuring system uses two different devices connected to two different ports of the PC to measure electrical power and temperatures, we decided to write our own data logger application instead of using two commercially available data loggers in the same time - one for each instrument. Leaving aside the costs involved with purchasing the commercial applications, a data logger designed and written in-house has the major advantage of being fully customizable - we can add new functions or options at any given time, as needed, making our team completely independent. In case of a commercial data logger, this would only be possible if the user purchases "add-on" functions or "plug-in" packages. It took us about 2 months to write our data logger application and we made 5 revisions so far, but being independent and having the possibility to modify the inner working of the application, worth the effort.

Acknowledgement

This work was supported by a grant of the Romanian Ministry of Education, CNCS – UEFISCDI, project numbers PN-II-RU-PD-2012-3-0270 and PN-II-RU-PD-2011-3-0021.

References

[1] A. P. Jardosh, K. Papagiannaki, E. M. Belding, K. C. Almeroth, G. Iannaccone, B. Vinnakota, "Green WLANs: On-demand", WLAN infrastructures, Mob. Netw. Appl. 14 (2009), 798–814. http://dx.doi.org/10.1007/s11036-008-0123-8.

[2] K. Dufkova and, M. Bjelica, B. Moon, L. Kencl, J.-Y. Le Boudec, "Energy savings for cellular network with evaluation of impact on data traffic performance", Wireless Conference (EW), 2010 European, 2010, pp. 916–923.

[3] L. M. Feeney, M. Nilsson, "Investigating the energy consumption of a network interface in an ad hoc networking environment", in: Proc. of the IEEE INFOCOM, Anchorage, USA, 2001, pp. 1548–1557.

[4] M.R.C. Truşcă, Ş. Albert, F. Fărcaş, M. L. Soran, M. Abrudean "Implementation of an Additional Monitoring Device to Prevent Failures of the Cooling Systems", Automation Quality and Testing Robotics (AQTR), 2012 IEEE International Conference on, ISBN 978-1-4673-0701-7, pp. 319 - 322, doi: http://10.1109/AQTR.2012.6237724

[5] Felix Farcas, Radu Trusca, Stefan Albert, Izabella Szabo, and Gabriel Popeneciu, "Application of green IT for physics data processing at INCDTIM", AIP Conf. Proc. 1425, 69 (2012); doi: http://10.1063/1.3681969

[6] M.R.C. Truşcă, Ş. Albert, F. Fărcaş, M. L. Soran, M. Abrudean, "Power consumption monitoring in ITIM Datacenter", Proceeding - Romania Tier 2 Federation "Grid, Cloud & High Performance Computing Science", 25 – 27 oct. 2012, Cluj-Napoca, Romania, 93-95.

[7] Algodue UPT-210 power meter: http://www.algodue.com/index.php?alias=catalogue&act=product&id=24

[8] Alfa Energ data logger: http://www.alfaenerg.ro/Produse/Monitorizarea-varfurilor-de-putere-si-prodata/Prodata-si-Data-logger

[9] Thor Labs data logger: http://www.thorlabs.de/software_pages/ViewSoftwarePage.cfm?Code=PM100x

[10] Agilent, Bench Link DataLogger: http://www.home.agilent.com/agilent/editorial.jspx?cc=RO&lc=eng&ckey=843028&nid=-11143.0.00&id=843028.

[11] The "WriteFile" WinAPI function: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365747 (v=vs.85).aspx.

[12] The "ReadFile" WinAPI function: http://msdn.microsoft.com/en-us/library/windows/desktop/aa365467 (v=vs.85).aspx.

[13] WINSOCK function list: http://msdn.microsoft.com/en-us/library/windows/desktop/ms741394 (v=vs.85).aspx.

Appendix 1

```
{Note: the variables used in the pseudo-code are described in Table 1}
START Procedure Data_Capture
                    OPEN INI file “DATALOGGER.INI”
                    READ serial communication settings
                    CLOSE file “DATALOGGER.INI”
                    SET serial_port  {setting the transmission parameters}
                    SET transmission_speed
                    SET bit_date
                    SET bit_stop
                    OPEN serial_port
                    SWITCH interface_TX
                    SEND data_request
                    SWITCH interface_RX
                      READ data_string
                    CALCULATE c_r_c {transmission control character}
IF (crc = ok) THEN
              START
                 READ (data_string, UF1, UF2, UF3, Usys)
                  READ (data_string, UL1, UL2, UL3)
                 READ (data_string, IL1, IL2, IL3, Isys)
                                    READ (data_string, cosfi1, cosfi2, cosfi3, cosfisys)
                                    READ (data_string, PAF1, PAF2, PAF3, PAFsys)
                  READ (data_string, PACTF1, PACTF2, PACTF3, PACTsys)
                 READ (data_string, UF1, UF2, UF3, Usys)
                 READ (data_string, PRF1, PRF2, PRF3, PRsys)
                  OPEN file “ELECTRICAL_VALUES.DAT”
                 WRITE (file, UF1, UF2, UF3, Usys)
                 WRITE (file, UL1, UL2, UL3)
                 WRITE (file, IL1, IL2, IL3,Isys)
                  WRITE (file, cosfi1, cosfi2, cosfi3, cosfisys)
                 WRITE (file, PAF1, PAF2, PAF3, PAFsys)
                 WRITE (file, PACTF1, PACTF2, PACTF3, PACTsys)
                 WRITE (file, UF1, UF2, UF3, Usys)
                 WRITE (file, PRF1, PRF2, PRF3, PRsys)
                  CLOSE file “ELECTRICAL_VALUES.DAT”
              END
CLOSE serial_port
                    READ IP_adress
                    WINSOCKET.CONNECT(IP_adress) {OPEN Ehernet port}
                    WINSOCKET.SENDDATA (COMAND) {data_request}
                       WINSOCKET.GETDATA (data_string) {read data string}
                    READ (data_string, T1, T2, T3, T4) {extracting the numerical values}
                    OPEN file “TEMPERATURES.DAT”
                    WRITE (file, T1, T2, T3, T4)
                    CLOSE file “TEMPERATURES.DAT”
                    WINSOCKET.CLOSE {CLOSE Ethernet port}
        END Procedure Data_Capture
```

Appendix 2

```
        START Procedure_coordinate_calculation
        {coordinates calculation for line voltage UL1}
        OPEN file “ELECTRICAL_VALUES.DAT”
        IF (autozoom = false) THEN
             START
                {search for MAX value in the data file}
                OPEN (“ELECTRICAL_VALUES.DAT”)
                DIM = FILESIZE( “ELECTRICAL_VALUES.DAT”)
                MAX = 0
```

```
        FOR K = (0, DIM) DO
        READ("ELECTRICAL_VALUES.DAT", value)
                    IF (marime > MAX) THEN MAX = value
                    {calculate the coordinates}
                    OPEN file "NUMBERS.DAT"
                    FOR K = (0, DIM)
                   START
                       READ ("ELECTRICAL_VALUES.DAT", value)
                        Y_MAX = 600 {the image has 800 x 600 pixels}
                        Y = value * Y_MAX
                        Y_grafic = Y_MAX – Y {coordinate correction}
                        {saving the coordinates to file}
                        WRITE ("NUMBERS.DAT", Y_grafic)
                    END
                   CLOSE ("NUMBERS.DAT")
                   CLOSE ("ELECTRICAL_VALUES.DAT")
               END
           {coordinates calculations for "autozoom" mode}
IF (autozoom = true) THEN
START
               OPEN ("ELECTRICAL_VALUES.DAT")
               DIM = FILESIZE( "ELECTRICAL_VALUES.DAT")
               {search for the minimum value in the data file}
               MIN = MAX
               FOR K = (0, DIM)
                IF (value < MIN) THEN MIN = value
               FOR K = (0, DIM) DO value = value – MIN
               {search for the MAX value in the file}
               MAX = 0
                           FOR K = (0, DIM)
                           IF (value > MAX) THEN MAX = value
                          {now we apply equation (1)}
                          OPEN ("NUMBERS.DAT")
                           FOR K = (0, DIM)
                           START
                               READ ("ELECTRICAL_VALUES.DAT", value)
                               Y_MAX = 600 {the image is 800 x 600 pixels}
                               Y = value * Y_MAX
                               Y_grafic = Y_MAX – Y {coordinate correction}
                               {saving the coordinates to file}
                               WRITE ("NUMBERS.DAT", Y_grafic)
                           END
                          CLOSE ("NUMBERS.DAT")
                          CLOSE ("ELECTRICAL_VALUES.DAT")
                     END
END Procedure_coordinate_calculation.
```