International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 02, Issue – 11, November – 2017, PP – 67-75

Demands' Manager at Arduino Platform

Moraes, Breno Gabriel Macedo[1], Rangel, Yan Ricardo Damasceno[2], Duarte Filho, Moisés[3], De Almeida, Luciana Lezira Pereira[4]

[1]*Electrical Engineer, Department of Electrical Engineering, Estácio de Sá University, Brazil*
[2]*Graduate Student, Department of Electrical Engineering, Estácio de Sá University, Brazil*
[3]*Professor of Undergraduate in Electrical Engineering Department, Estácio de Sá University, Brazil*
[4]*Professor of Undergraduate in Electrical Engineering Department, Estácio de Sá University, Brazil*

Abstract: In proposed paper is introduced the development of an electricity consumer demand manager for low voltage consumer with monitoring by SCADA system, with purpose to apply the demand side management (DSM). For such use the micro integrated Arduino controller with current and voltage sensors, those providing analog signals that are processed to obtain the active power information, and reactive power factor, and thus enabling the system to evaluate and control the loads electrical non-essential, aimed at managing consumption. The information obtained and subsidiaries are exposed in software with supervisory function graph on a monitor, enabling the visualization of the system by the end user.

Keywords: Arduino, Demand Side Manager, Energy efficiency, Microcontroller, SCADA System.

1. INTRODUCTION

Society is progressively concerned with climate change, and in this way, the rational and efficient use of energy becomes a constant challenge for the whole community. Energy efficiency is one of the best alternatives for reducing the rate of expansion of the electrical system.

The increase of the waste of electric energy necessarily implies in the increase of the installed power of generation. This increase represents a high cost, both environmental and equipment investments.

To have the possibility of monitoring and storing the data of the electrical parameters of residences, buildings, factories and companies, such as active power, reactive power, power factor, peak and actual values of voltages and currents, In order to reduce waste and the cost of consumption, it meets the policy of rational consumption of electricity.

According to the Program to Combat Electric Energy Waste, the consumption of electric energy in buildings corresponds to about 50% of consumption billed in the country (Brazil). It is estimated a potential reduction of this consumption by 50% for new buildings and 30% for those that promote reforms that contemplate the concepts of energy efficiency in buildings.

In order to reduce costs and reduce electricity consumption, the proposed electric demand manager is a system capable of managing this consumption, predicting and warning users when power consumption peaks occur, evaluating the scenario and automatically performing the Cut of the consumption of devices, with respect to the scale of priorities and hierarchy pre-defined, there is also the possibility to manually resume the use of some device if it is necessary.

The scale of priorities and hierarchies is based on the getting and classification of loads, that is, the loads covered by the system created are organized according to the degree of continuity they demand from the sources.

According to ISONI (2016), the loads can be classified into the following groups.

- Permanent Loads: Cannot be interrupted and are related to the operational continuity of the main system and to the safety of people and facilities.
- Essential Loads: Supports short interruptions. Shutting down such loads for longer periods can affect system reliability.
- Non-Essential Loads: Supports interruptions for a longer time.
- Emergency Loads: This group includes beacon and safety lighting, provided in case the Alternating Current Auxiliary Services system is lost.

This paper proposes the creation of a prototype electronic measure demand of electric energy consumption demand that calculates the active power, reactive, power factor and energy, allowing the easy visualization of its data by the supervisory system, also making possible the manual selection of loads, In order to facilitate end-user management.

2. OBJECTIVES

The objective of this paper is to create a manager demand ofenergy consumption that allows the acquisition and visualization of information about active power, reactive power, power factor, energy and the control of electric charges through the Arduino microcontroller and the SCADA system.

- Measure current and voltage;
- Calculate active, reactive power and power factor;
- Monitor the power consumption through the Arduino microcontroller;
- Perform management on the demand side;
- Develop the application of the SCADA system, in order to make available and control the system according to the information.

3. MATERIALS AND METHOD

3.1 MATERIALS

- Arduino MEGA;
- Arduino programmer software;
- Computer;
- Communication drivers;
- Current Sensor;
- Ethernet Shield;
- OPC server;
- Relay module, 8 channel 12v;
- Supervisory;
- Voltage Sensor;

3.2 METHODOLOGY

In this prototype, the measurement circuits are responsible for providing information for the Arduino Mega's inputs, and how the relay assembly was integrated with the function of disconnecting the loads after demand-side management.

The current and voltage variables are read by the analogue inputs of the Arduino, and in this way a logic was created for the active, reactive, real and power factor calculations, with this information the energy variable was calculated.

It has prepared a computer to be the programming terminal, where they are contained: the Arduino programmer, the OPC server, supervisory and communication drivers. The variables created in the Arduino are made available in the supervisor via the OPC communication protocol, so the "OPC Server for Arduino" is installed in the programming terminal.

Through the OPC library, Arduino publishes the variables of the developed program, the OPC server "consults" the serial port of the program, and verifies which variables are published and makes these variables visible.

An OPC client, which in this case is the supervisory communication driver, communicates with the OPC server, and verifies which variables are visible, so that it can read and write to them. This process is exemplified in Fig. 1.

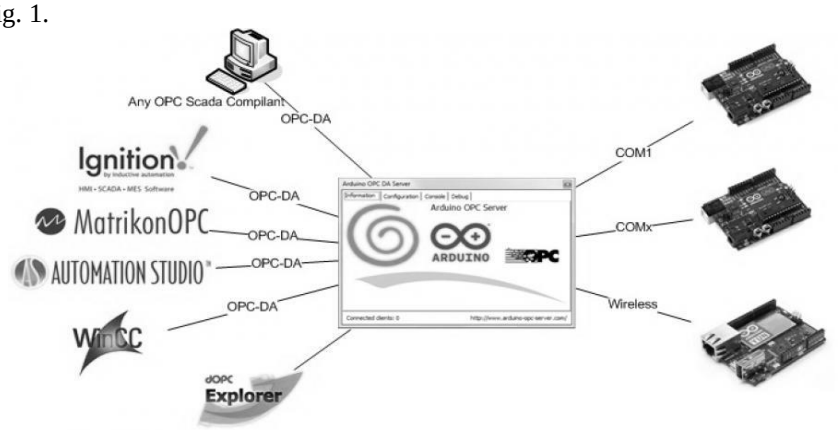

Figure 1 - OPC Server forArduino.

For the monitoring of the system, It was used the Wonderware Intouch supervisory system of the Schneide manufacturer. For communication with the OPC server, It was used the FactorySuite Gateway (also called FS Gateway), which is an application program that functions as a communications protocol converter. It can be used to connect clients and data sources that communicate using data from different access protocols, the information generated in the Arduino is available in the Intouch supervisor, where it is possible to monitor and control in real time the power consumption and the status of the loads . Fig. 2 shows the prototype with the Supervisor and Arduino controller.

Figure2 - Controller and Supervisory.

For communication of the Arduino with the MBTCP DAServer, using the Modbus TCP protocol Ethernet Shield was used to establish the TCP / IP communication with the programming terminal.

Fig. 3 shows the 8-channel relay module 12v, which, upon receiving the logic signal from the controller, will change the state of the relay, which in turn will act on the desired circuit to computational modeling.

In order to manage on the demand side using the peak cutoff method, encodings were developed in the Arduino and in the supervisory, which possessed the voltage and current information of the central circuit, could calculate and control the triangle of powers, Outputs, thereby making this information available to the supervisor via Modbus TCP. At the Arduino programming terminal, the logic model described below was developed to calculate the information required for the control of demand and made available through Modbus TCP.

Figure3 - 8-channel 12v Relay module.

Logical model:

```
//============================================================
#include < SPI.h>
#include < Ethernet.h>
#include <Modbus.h>
#include <ModbusIP.h>
#include "EmonLib.h"
//============================================================
const int DIGITAL3 = 200; // Recorder offset declaration for relay 1
const int DIGITAL4 = 300; // Recorder offset declaration for relay 2
```



```
const int CORRENTE = 3; // Recorder offset statement for current
const int TENSAO = 4; // Recorder offset statement for voltage
const int ENERGIA = 5; // Recorder offset statement for power
const int FPOT = 6; // Recorder offset statement for power factor
//============================================================
const int pinoE3 = 3; // Pine Statement
const int pinoE4 = 4; // Pine Statement
const int analogCA = A1; // Pine Statement
const int analogDDP = A2; // Pine Statement
//============================================================
float Irms; // Statement of the current variable
float Vrms; // Declaration of the voltage variable
float FatorPot; // Declaration of the variable power factor
float PotAparente; // Declaration of apparent power variable
float PotReal; // Declaration of the real power variable
float hTotal; // Declaration of the energy variable (wh)
unsigned long ltmillis, tmillis, timems; // Declaration of variable time base
//============================================================
ModbusIP mb; // Instance Creation
//============================================================
void setup() {
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };  // MAC Address
byte ip[] = { 10, 161, 0, 2 };   // IP address for the shield
mb.config(mac, ip);
//============================================================
emon1.current(analogCA, 56.5);  // Current: Pin, Calibration
emon1.voltage(analogDDP,104.5, 1); // Voltage: Pin, Calibration, Phase shift
 //============================================================
pinMode(pinoE3, OUTPUT);  // Configuring the Pin as Output
pinMode(pinoE4, OUTPUT);  // Configuring the Pin as Output
//============================================================
mb.addCoil(DIGITAL3, true); // Recorder's declaration and initial value
mb.addCoil(DIGITAL4, true); // Declaration of the register and initial value
mb.addIreg(CORRENTE, 0); // Declaration of the register and initial value
mb.addIreg(TENSAO, 0); // Declaration of the register and initial value
mb.addIreg(ENERGIA, 0); // Declaration of the register and initial value
mb.addIreg(FPOT, 0); // Declaration of the register and initial value
}
//============================================================
void loop() {
mb.task(); // Cyclic call of the modbus routine
//============================================================
digitalWrite(pinoE3, mb.Coil(DIGITAL3));
digitalWrite(pinoE4, mb.Coil(DIGITAL4));
mb.Ireg(TENSAO, Vrms);
mb.Ireg(CORRENTE,Irms);
mb.Ireg(ENERGIA,whTotal);
mb.Ireg(FPOT,FatorPot);
//============================================================
emon1.calcVI(20,2000);
PotReal = emon1.realPower;
PotAparente = emon1.apparentPower;
FatorPot = emon1.powerFactor;
Vrms = emon1.Vrms;
Irms = emon1.Irms;
//============================================================
ltmillis = tmillis;
```



```
tmillis = millis();
timems = tmillis - ltmillis;
whTotal = whTotal + ((PotReal) * 1.0/3600.0 * (timems/1000.0));
}
//=========================================================
```

In the supervisory system, logics were developed to control loads during peak hours. Since the energy information value in kWh may be very small, it has been defined for the Arduino to provide the information in Wh, and the conversion is performed in the supervisory through the script indicated in Fig. 4.

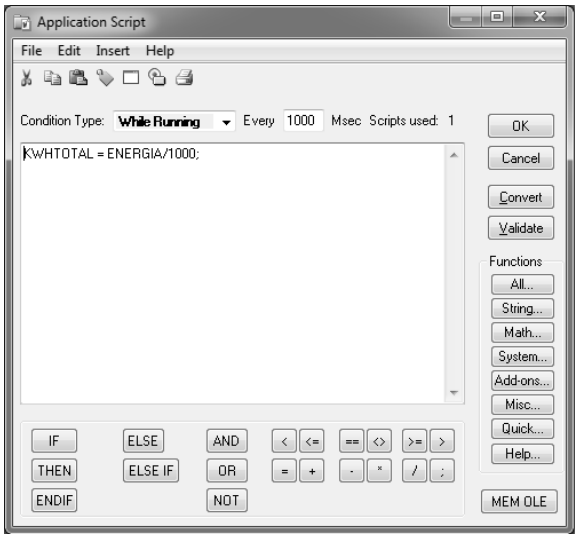

Figure 4 - Energy Unit Conversion Logic.

To identify the peak time, a script was created (see Fig. 5) that changes the status of an internal memory to "1" when the time is between 6:00 p.m. to 9:00 p.m. (peak time in Brazil), otherwise keep the condition at "0".

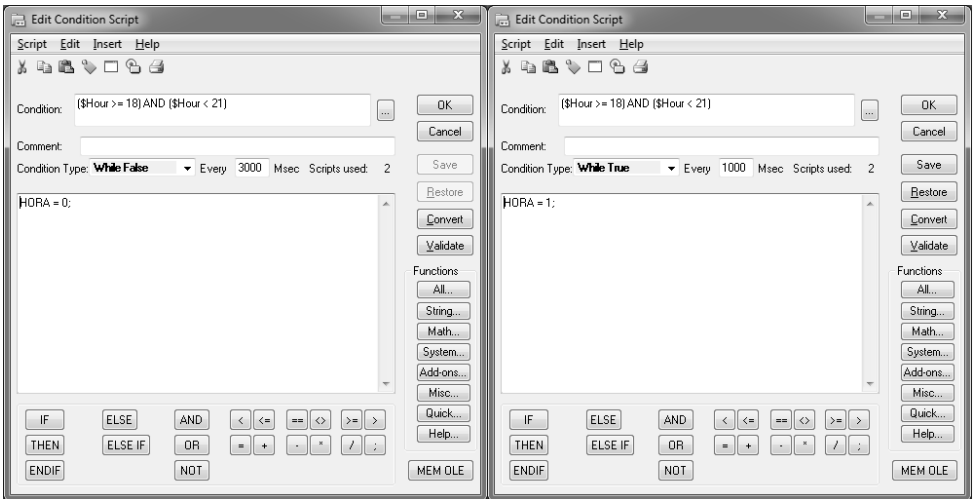

Figure5- Logic for peak time identification.

With the identification of the peak time through the internal memory "HORA", the logic for cutting the loads was developed, when this memory is one, there will be a cut of the loads, changing the value of the registers referring to the relays, as indicated in the Fig. 6.

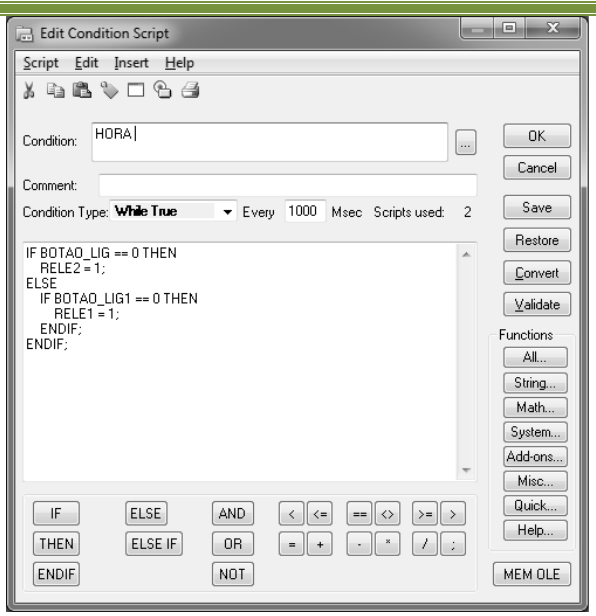

Figure6 - Cutting loads at peak times

The end user will be able to connect the loads that were cut during the peak times, so if he does this, the internal memories "BOTAO_LIG" and "BOTAO_LIG1" will switch to "1" according to the load that is connected, So these memories were included to prevent them from being switched off after the command, since within the peak times the loads tend to be turned off.

When the peak time runs out, the loads must be connected and therefore the logic of Fig. 7 has been developed.

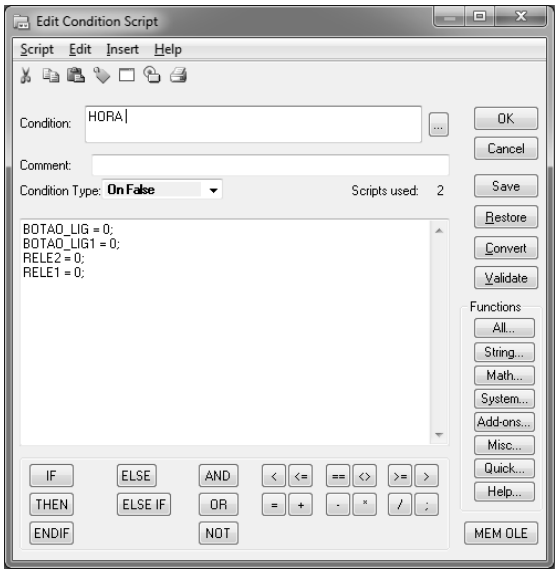

Figure7 - Button logic turns on at peak times

To turn on or off the loads just click on the animation of each one, when performing this action execute the script of Fig.8.

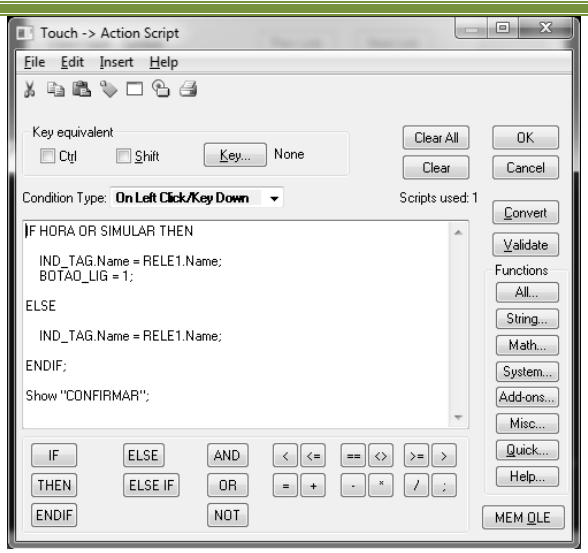

Figure8 - Logic to turn loads on or off

As indicated in Fig. 8, if it is in the peak time the variables "BOTAO_LIG" and "BOTAO_LIG1" are switched to "1" and preventing the load from being disconnected again, as explained above. Either in peak time or not the output to the relay will switch through the confirmation window.

In this window, the logic according to Fig. 9 has been developed for switching on and off.

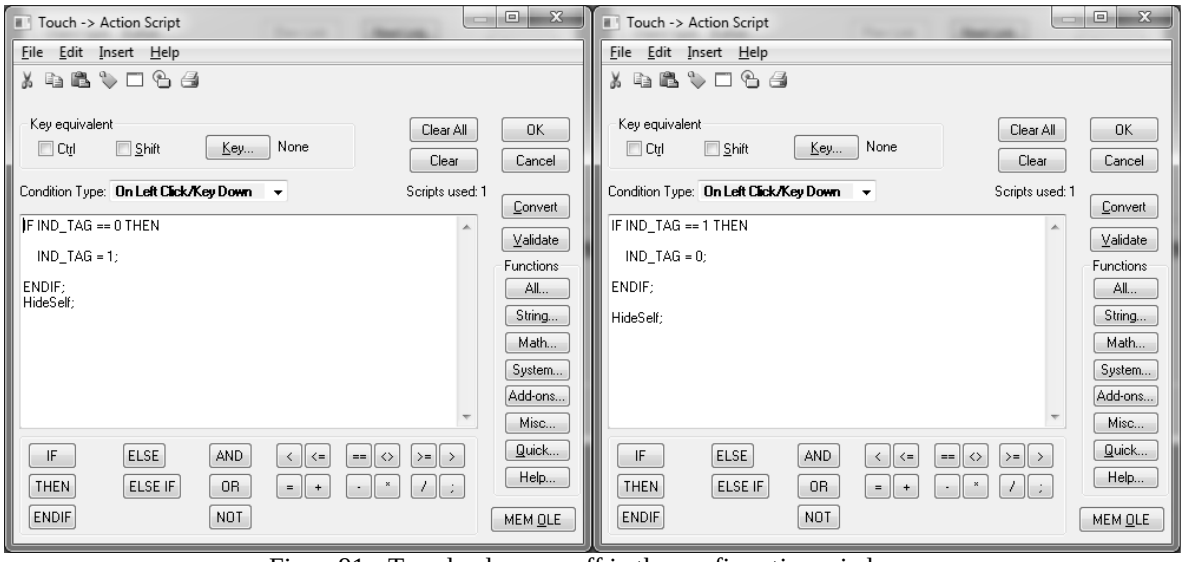

Figure91 - Turn loads on or off in the confirmation window.

To provide the information in the supervisor, the DAServer MBTCP communication driver was configured according to the parameters defined in the Arduino, as can be seen in Fig.10.

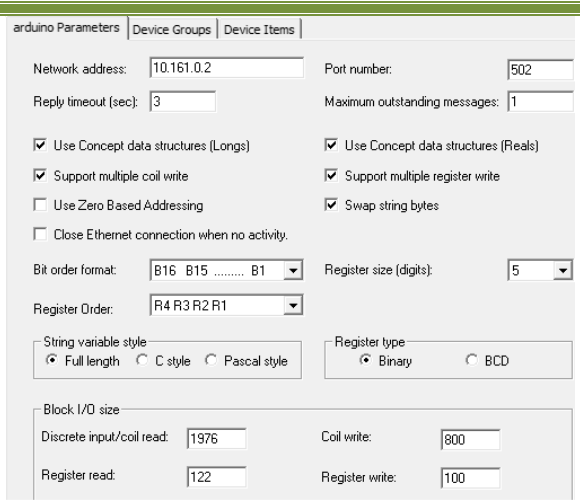

Figure 10 - Configuring the DAServer MBTCP Driver

The Intouch supervisory system requests data from the MBTCP DAServer communication driver, this in turn, records the variables made available by the Arduino.

With the variables available to the supervisor, the screens and notes of the TAGs were developed with the addresses provided by the communication driver.

For the creation of the TAGs the AccessName "MODBUS" was created, specifying the application where it should be read and the reference topic (see Fig. 11).

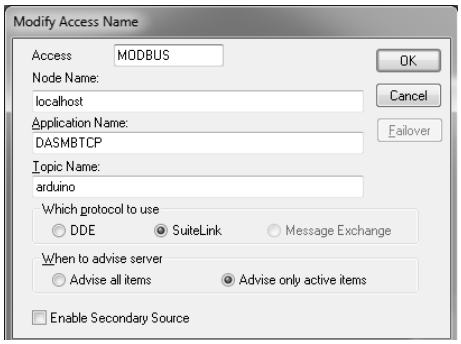

Figure21– AccessName's configuring.

Configure the AccessName, the TAGs were created, one for each variable made available by the Arduino, inserted as an address the read point of the MBTCP DAServer, linking the TAGs to the animations, the operation and monitoring screens were created (see Fig. Relays, current, voltage, energy, power factor, total kWh.

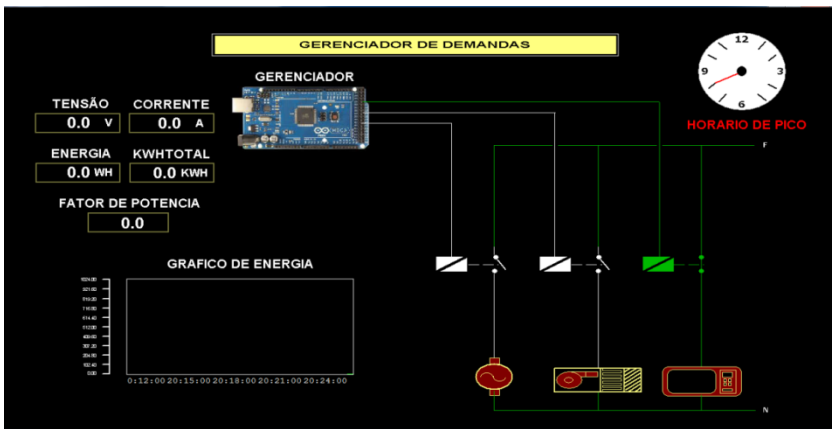

Figure12 - Supervisor Screen.

When you click on one of the loads, it opens the confirmation window to turn the load on or off, which is shown in Fig. 13.

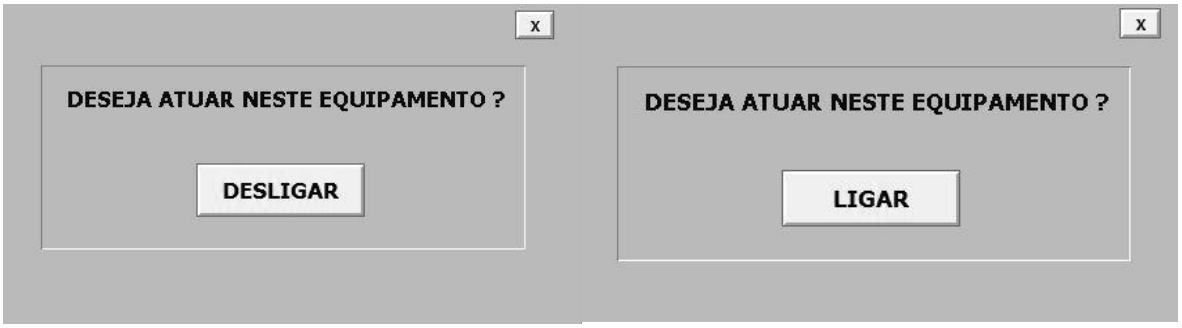

Figure 133 - Confirmation window.

4. RESULTS AND DISCUSSIONS

Using the Modbus TCP communication protocol, the communication between the Arduino controller and the supervisory controller was successful, allowing the visualization and control by the demand-side management through the peak cut.

With the calibration of the measurement circuits, the voltage and current variables were reliable and from these the active power and energy were calculated.

Therefore, we have succeeded in making the demand manager operational on the Arduino platform so that the energy of the final consumer is constantly monitored and presented in the supervisor, and when entering the peak hours, the manager performs the cut of the non-essential loads , Which are also indicated in the supervisor, and if some load is no longer essential in this period, the final consumer through the control of the supervisory, asks the manager that the load in question is re-energized.

5. CONCLUSION

Taking into account the aspects mentioned in this paper, it is concluded that with the growing concern with the rational and efficient use of energy and energy efficiency, they result in the best options to avoid demand and consequently lessening the need to expand the electric system.

In this paper, the demand manager in the Arduino platform, which uses the theory of demand side management, has been developed in this work by cutting the peak demand. This enables the visualization of information about energy, power factor, current and voltage and control the electric charges through the Arduino and SCADA system.

6. REFERENCES

[1] ISONI, M, Serviços Auxiliares Para Subestações. Conceitos Gerais e Principais Abordagens Técnicas, 2016.

[2] ANEEL. RESOLUÇÃO NORMATIVA Nº 414, OF SEPTEMBER 9, 2010, 9 September 2010. 69-70.

[3] INVENSYS SYSTEMS. FactorySuite Gateway User's Guide, Lake Forest, USA, 19 August 2014.

[4] JÚNIOR, J. P. D. S. COMBATE AO DESPERDÍCIO DE ENERGIA , JUIZ DE FORA, MG, Julho 2005.

[5] ALBUQUERQUE, C. F. D. Energia Ativa e Reativa, Itaí,December 2009. 1-2.

