Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

ANDRÉ GOMES MEDEIROS DE SOUZA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Campina Grande, Paraíba
Março de 2017

ANDRÉ GOMES MEDEIROS DE SOUZA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO REALIZADO NO LABORATÓRIO DE INTERFACE HOMEM MÁQUINA DA UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientadora:
Professora Maria de Fátima Queiroz Vieira, PhD.

Campina Grande, Paraíba
Março de 2017

ANDRÉ GOMES MEDEIROS DE SOUZA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professora Maria de Fátima Queiroz Vieira, PhD.
Universidade Federal de Campina Grande
Orientador, UFCG

SUMÁRIO

1 SUMÁRIO

1 APRESENTAÇÃO

Este relatório aborda as atividades do estágio supervisionado desenvolvido pelo aluno do curso de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG), André Gomes Medeiros de Souza, realizado no Laboratório de Interface Homem-Máquina (LIHM). O estágio foi realizado entre 08 de fevereiro de 2017 e 22 de março de 2017, totalizando uma carga horária de 184h.

1.1 LIHM

O Laboratório de Interfaces Homem-Máquina foi criado em 1986, no âmbito do Departamento de Engenharia Elétrica da UFCG, com o objetivo de apoiar pesquisas no desenvolvimento de interfaces para máquinas e sistemas. No ano de 2010, o LIHM lançou o selo de qualidade em Usabilidade, visando o aprimoramento de interfaces com o usuário. Atualmente o LIHM está envolvido no projeto do desenvolvimento de uma ferramenta de gerenciamento e monitoramento de descargas parciais, em cooperação com a Universidade de Huddersfield no Reino Unido.

O laboratório situa-se no bloco CI do Departamento de Engenharia Elétrica, sob coordenação da professora Dra. Maria de Fátima Queiroz, e ainda conta com a ajuda do professor Doutor Flávio Torres Filho. Para o acompanhamento das atividades realizadas durante o estágio, foram realizadas reuniões periódicas nas sextas, quando eram realizadas as prestações de conta.

O LIHM possui uma infraestrutura onde são realizados testes de usabilidade de Software de clientes, monitorando o comportamento do usuário durante o uso das ferramentas com propósito de identificar erros e possíveis melhorias na interface.

1.2 OBJETIVOS DAS ATIVIDADES

As atividades realizadas durante o estágio foram focadas no projeto de Monitoramento de Descargas Parciais, realizado em cooperação com a Universidade de Huddersfield no Reino Unido. No intuito de aprimorar a ferramenta, dando continuidade ao trabalho de conclusão de curso de Santos [1], foram realizadas as seguintes tarefas:

- Especificação de uma Base de Dados para o sistema Monitor de Descargas Parciais (MDP);
- Configuração de um Sistema de Gerenciamento de Banco de Dados (SGBD) para integração da base de dados e o sistema Monitor de Descargas Parciais;
- Integração da base de dados com o sistema Monitor de Descargas Parciais;
- Adaptação da Interface Homem-Máquina (IHM) para representar a estrutura de dados;
- Comunicação do sistema Monitor de Descargas Parciais com o *MatLab* via *OPC*;

2 REVISÃO BIBLIOGRÁFICA

Durante o período de estágio foram realizadas atividades que culminaram no desenvolvimento da base de dados para o projeto MDP, a integração da interface homem-máquina desenvolvida por Santos [1], e a configuração do servidor do LIHM para dar suporte ao acesso via Web.

2.1 SISTEMA MONITOR DE DESCARGAS PARCIAIS

O sistema Monitor de Descargas Parciais (MDP) é uma ferramenta desenvolvida em cooperação entre as Universidades de Huddersfield e de Strathclyde no Reino Unido (UK) – sob a coordenação do Professor Ian Glover, e a UFCG – sob a coordenação da professora Maria de Fátima Queiroz, a partir de um acordo de cooperação [1].

As descargas parciais são responsáveis por deterioração do material isolante de equipamentos de alta tensão, tais quais transformadores e isoladores, podendo levar ao mau funcionamento do equipamento e comprometimento de todo o sistema elétrico.

O sistema MDP é composto por uma rede de sensores, os radiômetros – sensores capazes de detectar a potência irradiada pelas descargas parciais. Enquanto a rede de sensores vem sendo desenvolvida no Reino Unido, no Brasil vem sendo desenvolvido o sistema de monitoramento com base no supervisório – InduSoft. Este sistema permite aos usuários monitorar o comportamento dos sensores, e estimar a localização da fonte de descargas parciais (PD), a partir da execução de algoritmos também desenvolvidos pela equipe do UK. Com a determinação estimada da localização é possível identificar em uma subestação um equipamento comprometido devido a PD [1].

Os dados produzidos pelos radiômetros são armazenados em um arquivo do tipo *xls* do *MicroSoft Excel*, e a localização da fonte de descargas parciais utiliza um algoritmo escrito por Umar

Khan da Universidade de Huddersfield, apoiado pelo artigo de Zangh [6], no formato *m* que corresponde a um script do MatLab.

2.2 EXTENSÃO DA FERRAMENTA

O projeto do sistema MDP previa a construção de um banco de dados para dar apoio a Interface Homem-Máquina. Por outro lado, com o desenvolvimento paralelo dos sensores foi necessário atualizar o projeto da IHM além de adequá-la a mudanças na estrutura de dados dos sensores.

Outra aspecto tratado neste estágio foi a configuração de uma conexão OPC para permitir a comunicação da ferramenta com o *MatLab*, visando executar o(s) algoritmo(s) de localização de descargas parciais.

A arquitetura da ferramenta é apresentada na figura 1, onde pode-se identificar seus diversos módulos e a comunicação entre eles.

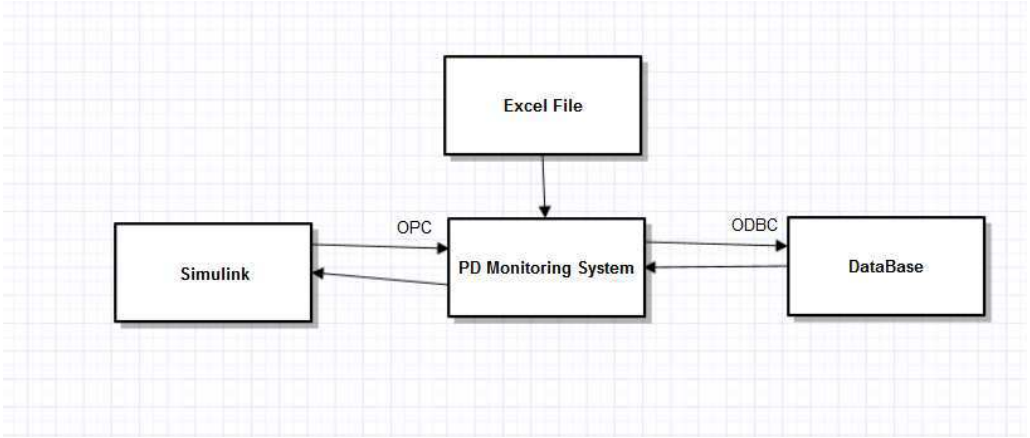


Figura 1: diagrama arquitetural do sistema monitor de descargas parciais. Fonte: O próprio autor.

As conexões entre os módulos podem ser realizadas via *InduSoft* [2], o qual disponibiliza os seguintes *drivers*: *ODBC, X e OPC* respectivamente. A seguir, cada um desses protocolos será descrito resumidamente.

2.2.1 OPC – *OLE FOR PROCESS CONTROL*

O OPC (*OLE for Process Control)* é um padrão industrial internacional utilizado para intercomunicação de sistemas. Esses padrões são mantidos pela *OPC Foundation* – fundação internacional especializada no desenvolvimento de aplicações de interoperabilidade industriais, especificando padrões de comunicação para o acesso a arquiteturas de dados.

Seu funcionamento é baseado na tecnologia *OLE (Object Linking and Embedding)* de componentes orientados a objeto, através das especificações COM e DCOM da *Microsoft,* permitindo o acesso a dados por um ou mais computadores que utilizem a arquitetura Cliente/Servidor.

A grande vantagem da especificação OPC se dá pela padronização de objetos, interfaces e métodos, mantendo uma estrutura coerente para diversas aplicações Cliente/Servidor – isso implica que as diversas aplicações podem operar com formatações de dados independentes em sua estrutura interna, mas os dados advindos das trocas de informações entre aplicações através do OPC possuem uma estrutura e formatação padrão, especificada pelo próprio OPC. Em outras palavras, duas aplicações que se comunicam através do protocolo OPC não precisam conhecer a origem da fonte de dados, visto que a especificação OPC padroniza o que cada aplicação recebe/envia durante a comunicação.

A arquitetura apresentada na figura 2 define a comunicação entre dois sistemas utilizando o protocolo OPC.

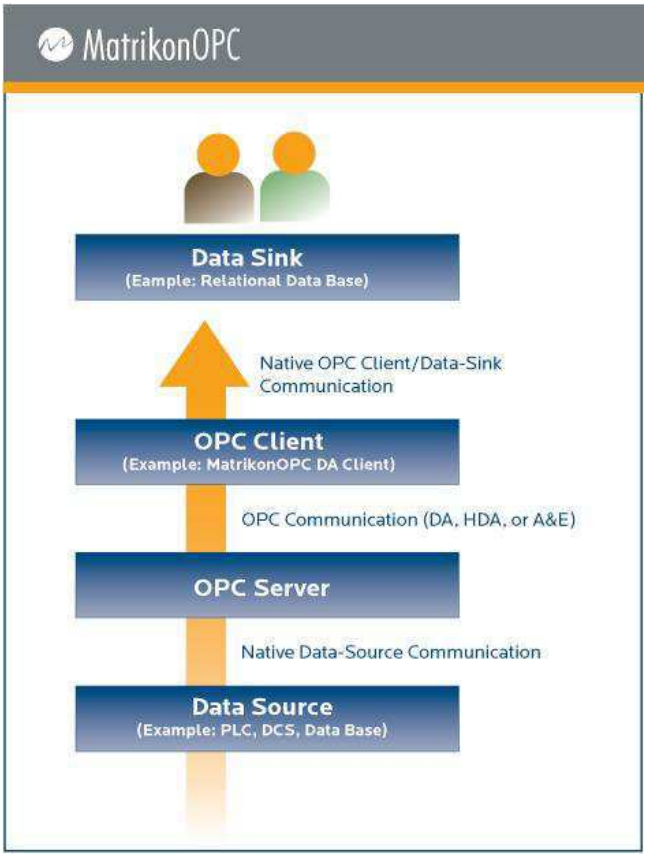


Figura 2: Diagrama arquitetural do protocolo OPC. Fonte: [3].

2.2.2 ODBC – *OPEN DATABASE CONNECTIVITY*

O padrão ODBC foi desenvolvido pela *Microsoft* com o objetivo de fornecer uma interface de comunicação entre bancos relacionais e aplicações – baseado no nível de especificação padrão SQL, com alta interoperabilidade – é possível uma única aplicação se comunicar com mais de uma fonte ou base de dados, utilizando o *driver* ODBC.

A arquitetura padrão ODBC é apresentada na figura 3.

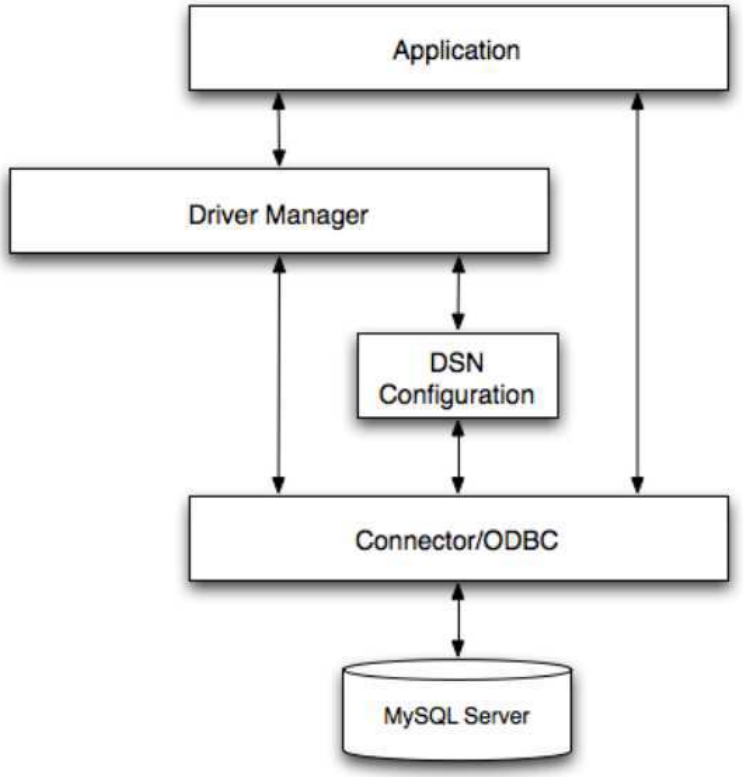


Figura 3: Diagrama arquitetural do driver ODBC. Fonte: [5].

A aplicação solicita as operações de banco, obedecendo aos padrões definidos pelo *Driver Manager*, que em grande parte dos casos apresenta funções padronizadas em SQL, diminuindo o esforço do programador em construir toda a expressão em SQL. Além disso, o *Driver Manager* é responsável por lidar com as exceções e regras de acesso ao SGBD.

2.2.3 VISÃO GERAL DA ARQUITETURA DO SISTEMA MONITOR DE DESCARGAS PARCIAIS

A integração entre estes *drivers* de comunicação é ilustrada na Figura 4, onde é apresentado um diagrama UMLde caso de uso detalhando o comportamento do sistema a partir da interação com o usuário.

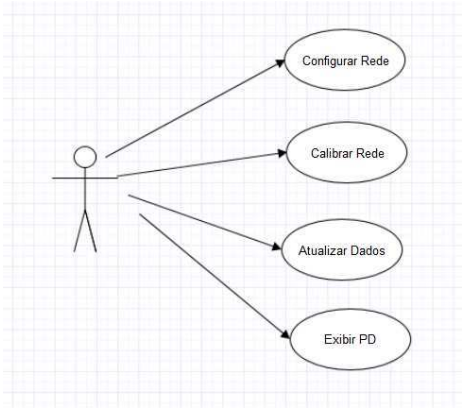


Figura 4: Caso de uso da ferramenta MDP. Fonte: O próprio autor.

Ao partir do diagrama de caso de uso foi construído um pseudocódigo que descreve a comunicação dos módulos da arquitetura, nas operações de leitura e escrita. O pseudocódigo (ver figura 5) define as tarefas que serão realizadas pelo *InduSoft*.

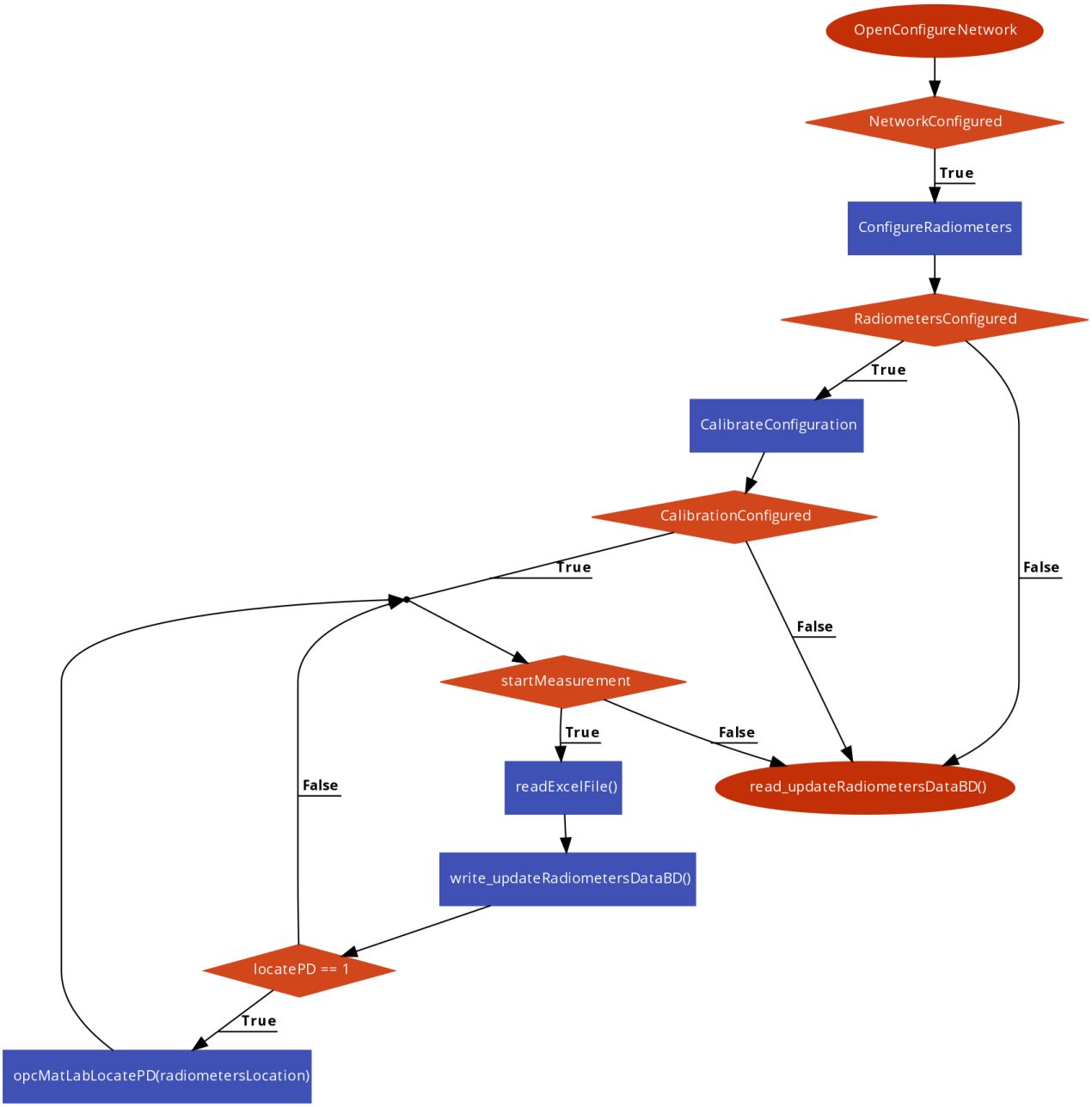


Figura 5: Pseudocódigo do sistema MDP. Fonte: O próprio autor.

A partir do pseudocódigo foram identificadas as funções que posteriormente foram codificadas na expansão da ferramenta MDP.

3 DESCRIÇÃO DAS ATIVIDADES

Nesta sessão discutiremos as atividades desenvolvidas no laboratório durante o estágio supervisionado. Inicialmente, discutiremos a especificação da base de dados. Após a construção do banco de dados e da configuração do SGBD, falaremos sobre a integração do banco de dados com o *InduSoft*. Em seguida, serão apresentadas as telas da IHM que que foram modificadas para atender as necessidades atuais do projeto: a representação dos dados dos radiômetros na forma de *snapshot* e a construção dos gráficos de tendência. Por fim será descrito como foram configurados o *InduSoft* e o *MatLab* para execução do algoritmo de localização de descargas parciais.

3.1 ESPECIFICAÇÃO DO BANCO DE DADOS

A especificação do Modelo de Dados teve como base a IHM desenvolvida por Santos [1], na qual figuram todos os elementos que devem ser armazenados no banco de dados. A seguir são apresentadas as telas da IHM do sistema. Destaca-se o caráter bilíngue do projeto o que nos levou a construir a IHM em Inglês dado que os usuários utilizarão o sistema em língua Inglesa.

3.1.1 TELA DE CONFIGURAÇÃO DE REDE

A figura 6 apresenta a tela de configuração da rede de sensores.

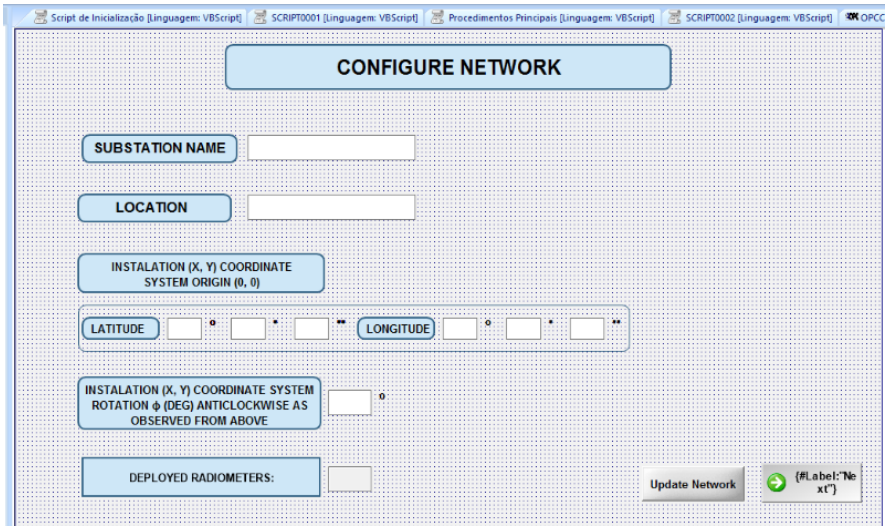


Figura 6: Visualização da tela de configuração de rede. Fonte: o próprio autor.

Observando os elementos gráficos da TELA DE CONFIGURAÇÃO DE REDE, são identificados 5 objetos que devem ser armazenados no banco de dados:

- *Substation Name* (SubEstação);
- *Location* (Localização);
- *System Origins Reference* (Refrência Geográfica para a origem do plano de coordenadas (x,y) em termos de latitude e longitude);
- *System Angle Rotatio*n (Rotação angular do sistema);
- *Deployed Radiometers* (Radiômetros lançados na rede).

No BD foram definidas duas entidades: *Network Configuration File* e *Station File,* as quais contemplam todos os objetos listados acima os quais por sua vez armazenam as informações referentes à subestação. Outra entidade que está representada é a localização geográfica:

- ID para especificar a localização (latitude ou longitude);
- Graus, minutos e segundos.

3.1.2 TELA DE CONFIGURAÇÃO DE RADIÔMETROS

Na figura 7 está ilustrada a tela de configuração dos radiômetros, onde o usuário pode editar a configuração dos radiômetros instalados em uma subestação.

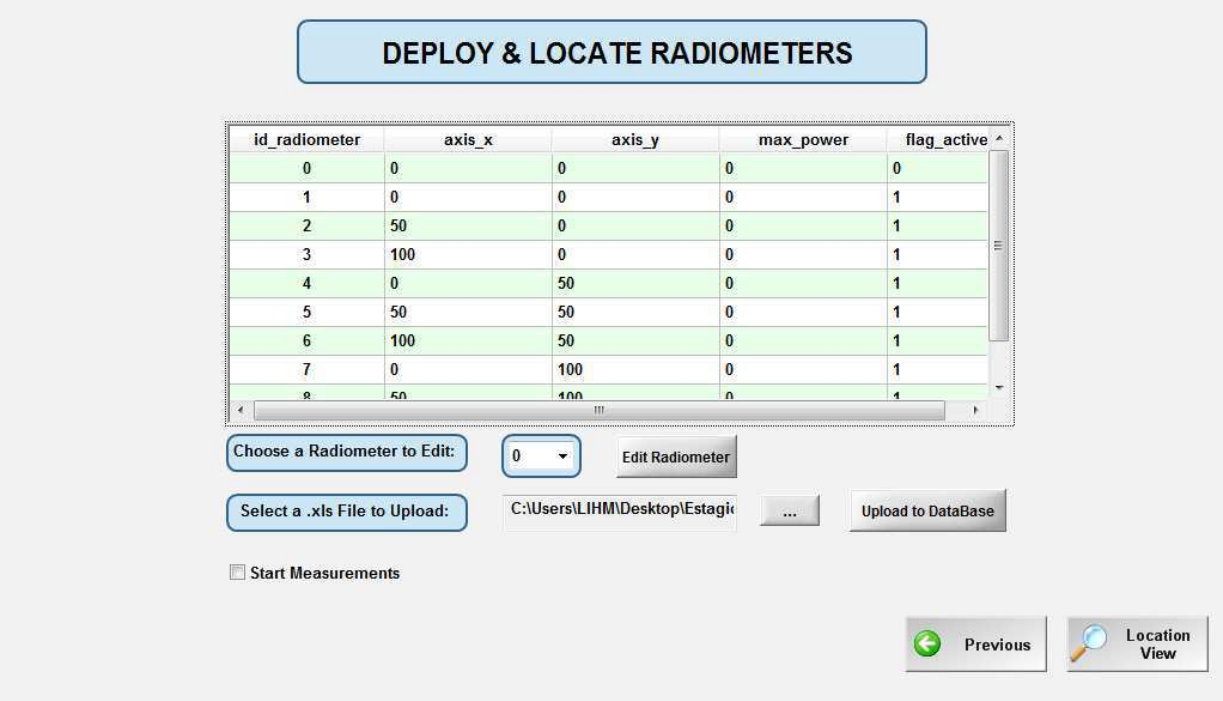


Figura 7: Tela de Configuração de Radiômetros. Fonte: O próprio autor.

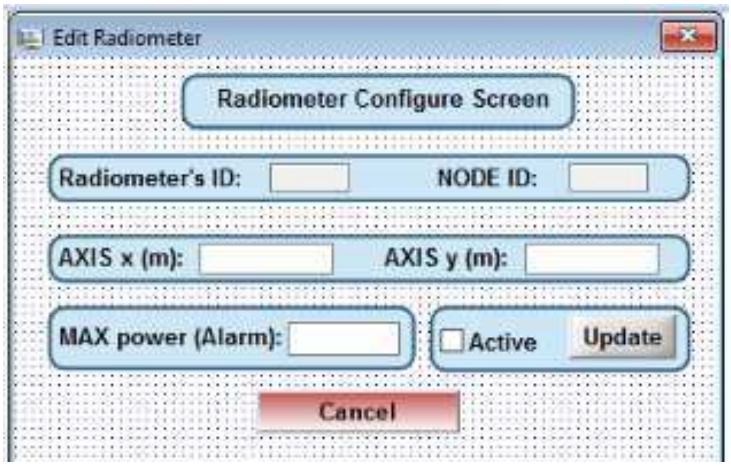


Figura 8: Tela de configuração de um radiômetro. Fonte: o próprio autor.

A entidade *Radiometers File* representa os dados de localização (latitude e longitude) dos radiômetros a partir do conjunto de dados listados a seguir:

- *Node ID* (ID de identificação do radiômetro);
- *Axis_x e Axis_y* (Ponto de coordenadas cartesianas);
- *Max. Power* (Potência Máxima);
- Localização (Latitude e longitude);

3.1.3 TELAS DE CALIBRAÇÃO

Nas telas de calibração (figuras 9 e 10) são informados os dados relativos ao tempo de atualização das informações no banco de dados, proveniente dos sensores no campo, assim como informações de calibração dos sensores.

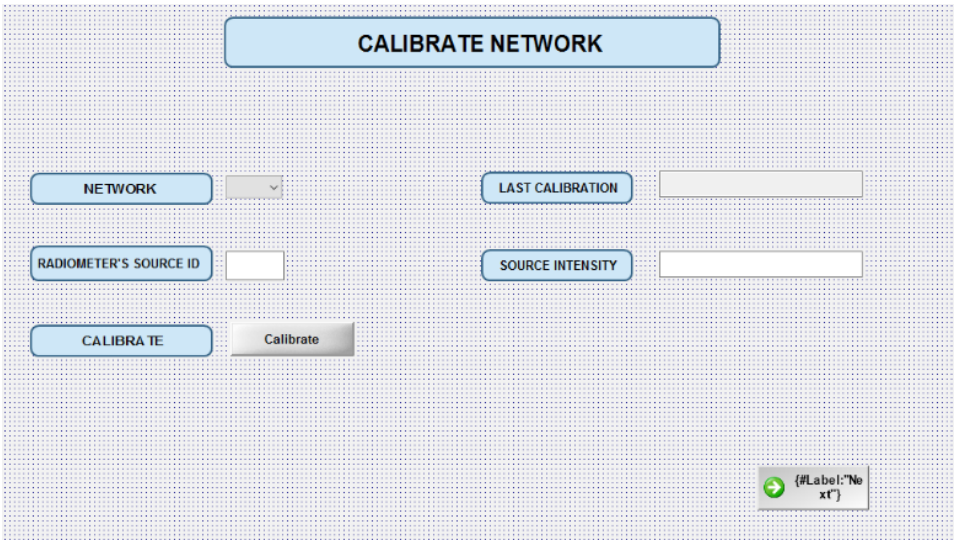


Figura 9: Tela de Configuração dos parâmetros de calibração. Fonte: O próprio autor.

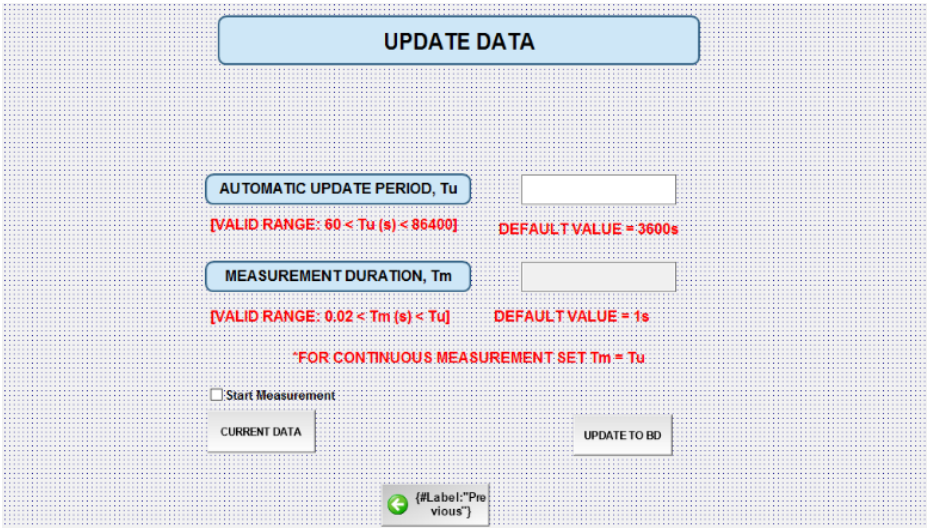


Figura 10: Tela de configuração da atualização dos dados para o Banco de Dados. Fonte: O próprio autor.

Para armazenar estas informações, foi modelada a entidade *Calibration Configuration File,* que armazena as seguintes informações:

- Id da rede que está sendo configurada;
- O radiômetro de referência para calibração;

- O valor da intensidade em potência (mW) do sinal recebido pelo radiômetro;
- Tempo de atualização automático;
- Tempo médio de duração de uma medição.

3.1.4 TELA DE CONFIGURAÇÃO DOS ALGORITMOS DE LOCALIZAÇÃO DE PD

Nas figuras 11 e 12 são ilustradas as telas nas quais o usuário seleciona um ou mais algoritmos para serem executados no cálculo da localização das descargas parciais.

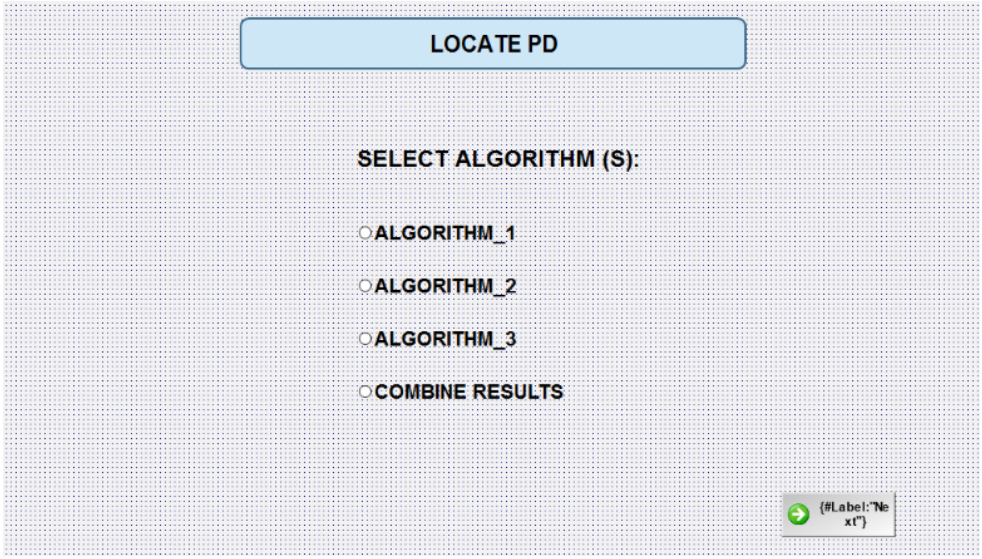


Figura 11: Tela de seleção de algoritmos. Fonte: O próprio autor.

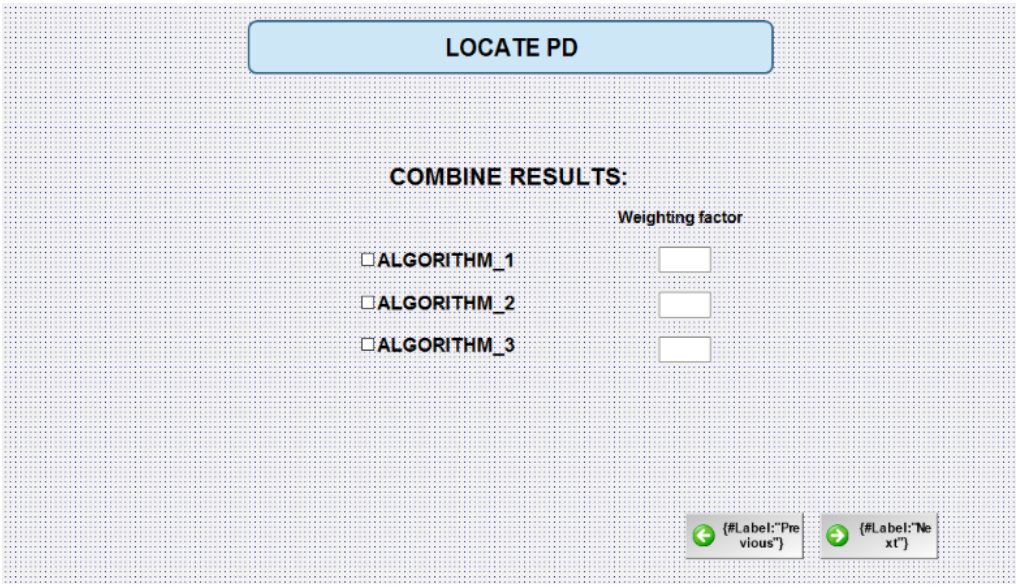


Figura 12: Tela de seleção do peso dos algoritmos no cálculo da localização. Fonte: O próprio autor.

Assim, a entidade *Algorithm Selection* armazenará as seguintes informações:

- Algoritmo Selecionado (1, 2, 3, ou uma combinação dos 3);

- Peso de cada algoritmo (1, 2 e 3);

3.1.5 ESTRUTURA DOS DADOS DO SENSOR – ARQUIVO MODELO DO EXCEL

As leituras dos sensores também são armazenadas no banco de dados. A partir do estudo de uma especificação cedida pelo professor Ian Glover, foi definida a entidade, *Data Acquired,* que armazena as informações contidas na estrutura de dados representada no arquivo *.xls*

Tabela 1: Trecho do arquivo *.xls* contendo os dados de leitura dos radiômetros. Fonte: O próprio autor.

Date & Time	*Node ID*	*Round*	*Number of Pulses*	*Number of Pulses Averaged*	*Average step size (mV)*	*Received Power (dBm)*
01/06/2017 14:54	1	1	101	95	86,73	-12,49
01/06/2017 14:54	2	1	100	94	80,00	-10,25
01/06/2017 14:54	3	1	99	93	69,30	-12,69

Assim, a entidade *Data Acquired* armazena os seguintes objetos:

- *TimeStamp* da leitura;
- O ID do radiômetro que executou a leitura;
- O *round* – rodada – da leitura do radiômetro;
- Número de pulsos;
- Média do número de pulsos;
- Média do passo de tensão;
- Potência recebida;

Aqui é importante ressaltar que a equipe do Reino Unido solicitou que o banco armazenasse 3 canais para a leitura de cada radiômetro. Assim, para cada radiômetro é armazenado:

- *Média de tensão do Channel 1;*
- *Média de tensão do Channel 2;*
- *Média de tensão do Channel 3;*

3.1.6 EXEMPLO DAS CONSULTAS EM SQL E DEFINIÇÃO DAS TUPLAS

As consultas a seguir, formuladas na linguagem SQL foram formuladas para validar o banco de dados através da IHM. Na tabela 2, ilustrada a seguir, são apresentadas as consultas SQL elaboradas para a IHM da ferramenta.

Tabela 2: Lista das principais consultas para validação do BD do sistema. Fonte: O próprio autor.

Consulta SQL	**Expressão em SQL**
Qual a subestação que os sensores foram instalados?	SELECT *location_name* from network_configuration_file WHERE id_network=0;
Qual a localização referente ao ponto de origem que representa o sistema cartesiano?	SELECT *id_location* from network_configuration_file WHERE id_network=0;
Qual a rotação angular do sistema de sensores?	SELECT *angle_orientation* from newtork_configuration_file WHERE id_network=0;
Qual a data da última calibração?	SELECT *last_calibration* from calibration_configuration_file WHERE id_calibration = 0;
Qual o radiômetro de referência para calibração?	SELECT *id_radiometer* from calibration_configuration_file WHERE id_calibration = 0;
Qual a intensidade da fonte de referência?	SELECT *source_intensity* from calibration_configuration_file WHERE id_calibration = 0;
Qual o período de atualização?	SELECT *period_tu* from calibration_configuration_file WHERE id_calibration = 0;
Qual o período de medição?	SELECT *measurement_period_tm* from calibration_configuration_file WHERE id_calibration = 0;

A seguir foram definidas as tuplas que representam as entidades do BD. As tuplas representam os relacionamentos entre as entidades do BD, com destaque para as chaves primárias e estrangeiras:

Radiometer_File (id_radiometer, id_rad_latitude*, id_rad_longitude*, axis_x, axis_y, max_power, flag_active);

Station_File (id_local_station, station_name, id_latitude*, id_longitude*);

Algorithm_Selection (id_algorithm, selected_algorithm, combined_result, weight_factor_a1, weight_factor_a2, weight_factor_a3);

Network_Configuration_File (id_network, id_local_station*, location_name, id_origins_reference_lat*, id_origins_reference_long*, angle_orientation, radiometer_total);

Calibration_Configuration_File (id_calibration, id_network*, last_calibration, source_itensity, period_tu, measurement_duration_tm);

Local_Coordinates (id_local_coordinates, degrees, minutes, seconds);

Data_Acquired (id_data_acquired, measurement_date, number_of_pulses, number_of_averaged_pulses, averaged_step_size, channel1_voltage, channel2_voltage, channel3_voltage, received_power, id_radiometer*);

3.1.7 TABELAS DO BD

Seguem as tabelas 3 a 9, do banco de dados, com seus respectivos atributos. O modelo EER pode ser consultado no apêndice D.

Tabela 3: Entidade *Radiometer_File*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_radiometer	Inteiro
axis_x	Inteiro
axis_y	Inteiro
*id_rad__latitude**	Inteiro
*id_rad_longitude**	Inteiro
max_power	Float
flag_active	Boolean

Tabela 4: Entidade *Station_File*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_local_station	Inteiro
station_name	Char(45)
*id_latitude**	Inteiro
*id_longitude**	Inteiro

Tabela 5: Entidade *Algorithm_Selection*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_algorithm	Inteiro
selected_algorithm	Inteiro
combined_result	Boolean
weight_factor_a1	Float
weight_factor_a2	Float
weight_factor_a3	Float
*id_network**	Inteiro

Tabela 6: Entidade *Network_Configuration_File*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_network	Inteiro
*id_local_station**	Inteiro
location_name	Char(45)
*id_origins_reference_lat**	Inteiro
*id_origins_reference_long**	Inteiro
angle_orientation	Float
radiometer_total	Inteiro
flag_active	Boolean

Tabela 7: Entidade *Calibration_Configuration_File*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_calibration	Inteiro
*id_network**	Inteiro
last_calibration	Date
id_source_location	Inteiro
id_radiometer)	Inteiro
source_intensity	Float
period_tu	Float
measurement_duration_tm	Float

Tabela 8: Entidade *Local_Coordinates*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_local_coordinates	Inteiro
degrees	Float
minutes	Float
seconds	Float

Tabela 9: Entidade *Data_Acquired*, seus atributos e tipos. Fonte: O próprio autor.

Atributo	Tipo
id_data_acquired	Inteiro
measurement_date	Timestamp
number_of_pulses	Time
number_of_averaged_pulses	Inteiro
averaged_step_size	Float
channel1_voltage	Float
channel2_voltage	Float
channel3_voltage	Float
*id_radiometer**	Inteiro
Received_power	Float

3.2 *SGBD MySQL* – CONFIGURAÇÃO DO BANCO DE DADOS

Para construir o banco de dados, foi selecionado um Sistema Gerenciador de banco de dados (SGBD) o qual foi conectado com a ferramenta em desenvolvimento. O supervisório *InduSoft* suporta diversos tipos de SGBD, porém foi selecionado o SGBD *MySql* devido a familiaridade do autor com este software.

Os requisitos necessários para executar o banco de dados, são: em uma máquina com o SO Windows 7, ou mais recente, na versão 64 bits além de :

- *Microsoft* .NET framework;
- *Visual C++ Redistributable for Visual 2015*;
- *MySql Server* e demais componentes (driver ODBC, .NET, etc) – o link para download pode ser acessado através de [4];
- *MySql Workbench v. 6.3;*

Foi utilizada a versão *MySQL WorkBench 6.3.8 build 1228 CE,* a qual pode ser obtida para *download em* [4]. Durante a instalação do *MySql*, o componente de drive *ODBC* foi instalado automaticamente. Este padrão é reconhecido pelo *InduSoft*.

3.2.1 CRIANDO UMA CONEXÃO PARA O BANCO

A figura 13 ilustra a tela de criação de uma nova conexão de banco de dados do *MySql.*

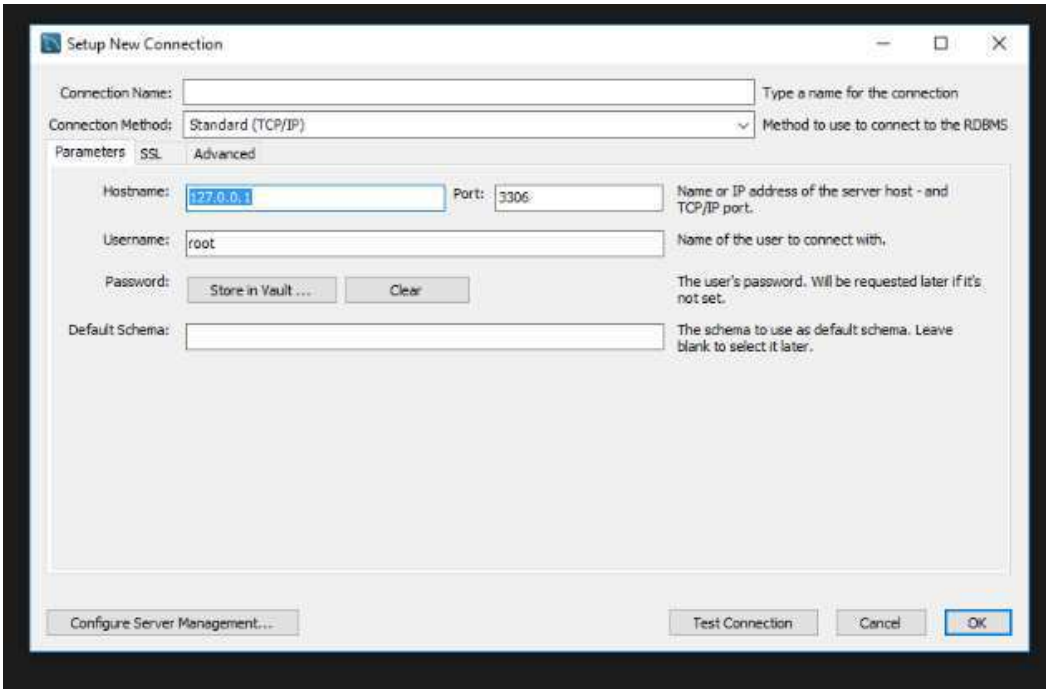


Figura 13: Configuração do SGBD *MySql*. Fonte: O Próprio autor.

No campo *HostName* deve-se especificar o endereço de IP da máquina que executará o SGBD. Caso o servidor esteja instalado na mesma máquina, basta preencher o campo com a string *localhost*. A porta padrão não deve ser alterada, visto que a porta de comunicação padrão reconhecida pelo *InduSoft* é a porta *3306*.

Em seguida, é definido um usuário e senha para controle de acesso. Após estes passos foi testada a conexão. Caso tudo esteja configurado corretamente, será exibida a mensagem ilustrada na figura 14.

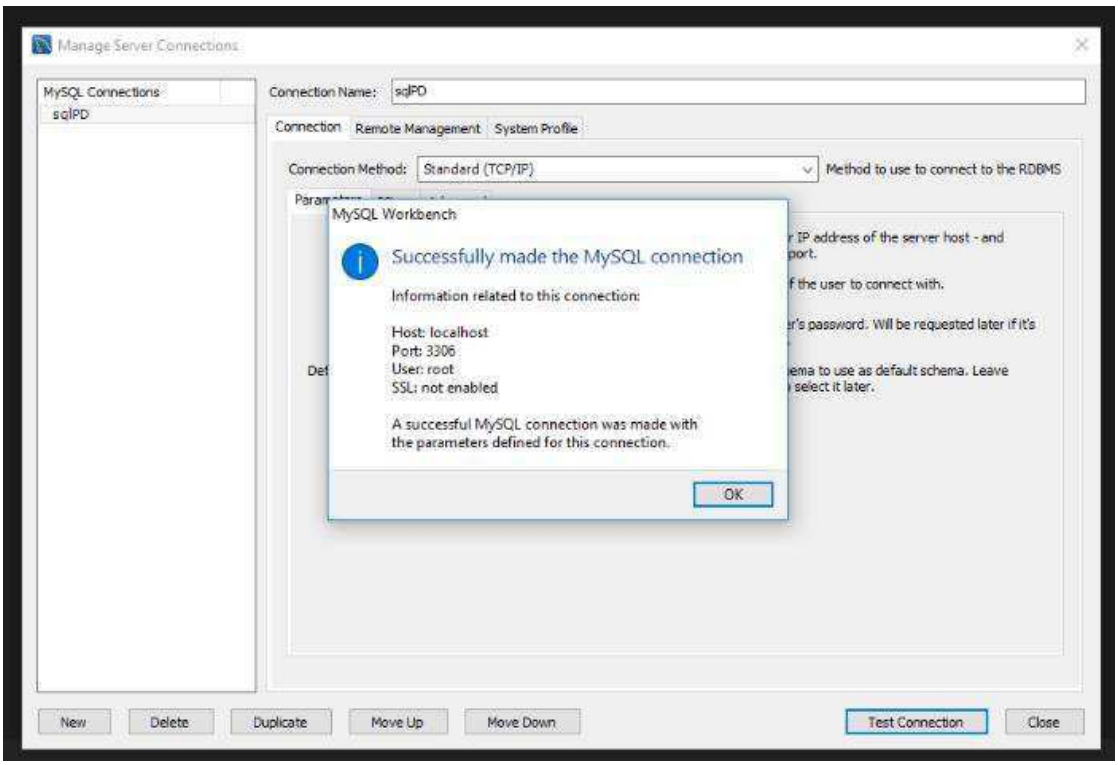


Figura 14: Mensagem de configuração Bem-sucedida do SGBD *MySql*. Fonte: O Própio autor.

Com a conexão configurada, deve-se gerar o script em SQL do modelo EER e, executá-lo pelo servidor criado, como apresentado na figura 15.

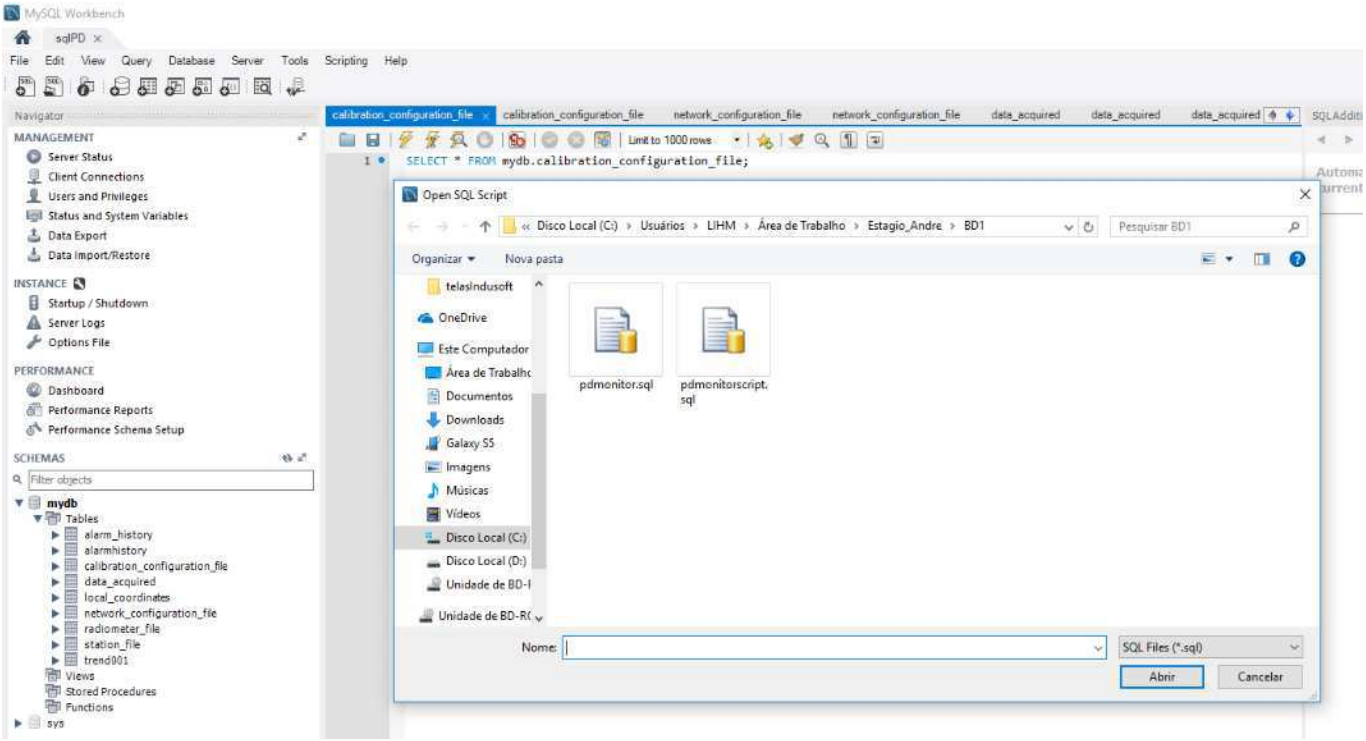


Figura 15: Tela de seleção de script para execução no *MySql* – criação da base de dados. Fonte: O próprio autor.

Após a execução do script, espera-se uma mensagem de confirmação. Após essa mensagem, o banco de dados está pronto para ser integrado à aplicação do *InduSoft.*

3.3 INTEGRAÇÃO: BANCO DE DADOS E *INDUSOFT*

Como discutido na sessão 2, o driver de comunicação ODBC fornece recursos para a conexão necessária entre o SGBD *MySql* e o *Indusoft*

Na tela de opções de projeto do *InduSoft* o usuário deverá selecionar a opção comunicação (ver figura 16) e acionar o botão *Configurar Banco de Dados Padrão*, quando será exibida a tela de opções de comunicação apresentada na figura 17.

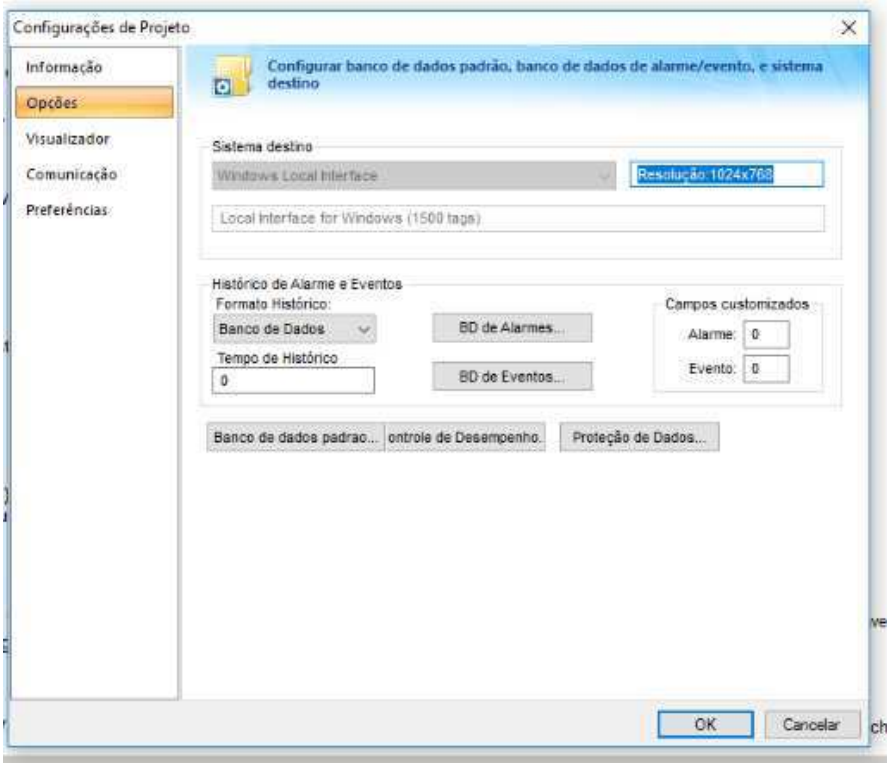


Figura 16: Tela de Opções de Comunicação do *Indusoft*. Fonte: O próprio autor.

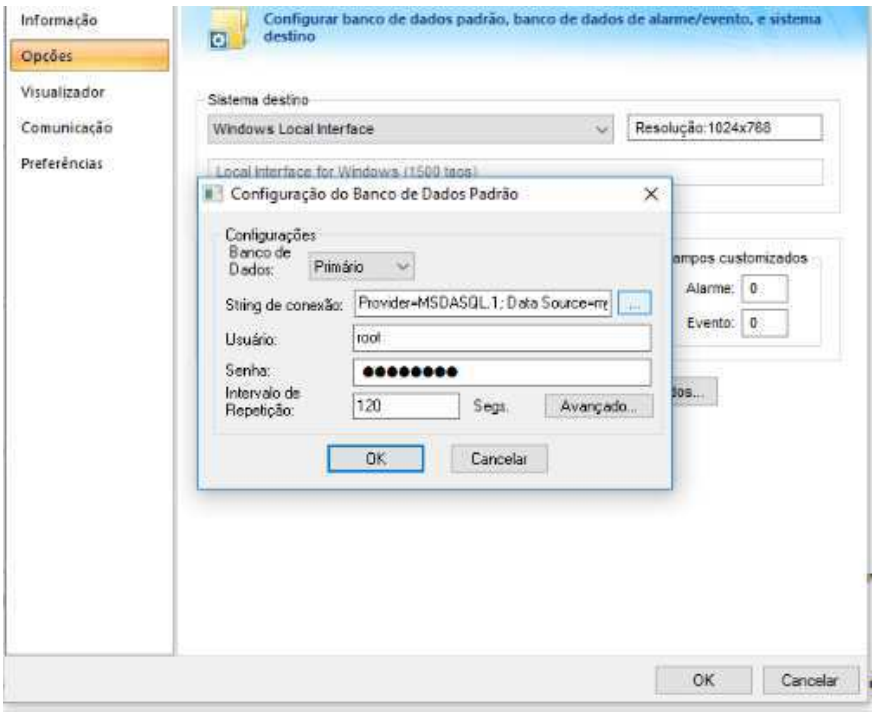


Figura 17: Tela de Configuração da base de dados primária do projeto. Fonte: O próprio autor.

Na Tela de configuração (Figura 17), após clicar no botão "...", é exibida a tela de configuração da conexão, ilustrada na figura 18. Nela, seleciona-se o servidor de banco de dados, que no caso deste projeto é o *"mydb"*. Observar que o nome do serviço de nome é o mesmo utilizado para executar o script *SQL* do modelo EER. Ao selecionar o servidor de banco de dados, pode-se escolher também o catálogo referente ao modelo EER, que neste caso também é a string *"mydb"*.

Realizado o teste de conexão com sucesso, o banco de dados estará conectado ao *InduSoft.*

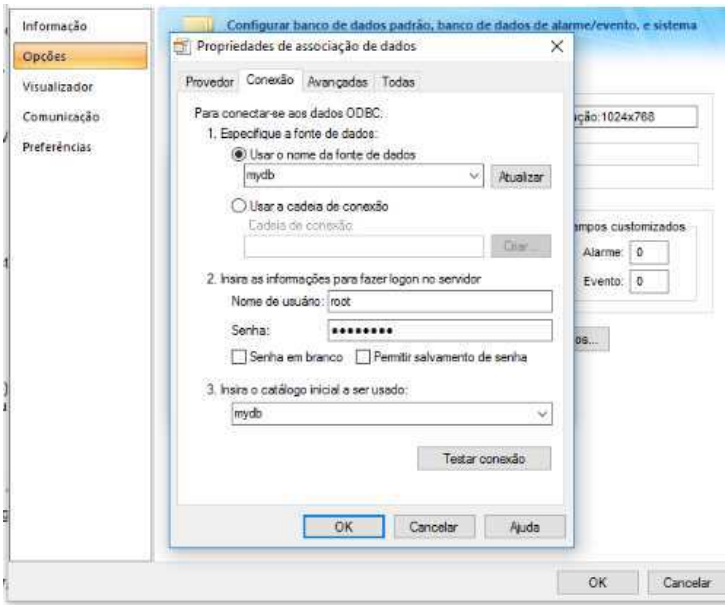


Figura 18: Opções avançadas de configuração da base de dados primária. Fonte: O próprio autor.

Os métodos de acesso ao banco de dados foram escritos na forma de um script, na linguagem padrão de scripts do *InduSoft* – o VBScript. Todos os métodos de acesso ao banco de dados, assim como os métodos de acesso ao arquivo *.xls* do Excel estão descritos no apêndice A – script de procedimento global.

3.4 INTEGRAÇÃO: *INDUSOFT E MATLAB*

Para realizar a comunicação entre *Indusoft* e *MatLab* é necessária a instalação de um *OPC Server*. Nesse projeto, a ferramenta *Indusoft* já possui um servidor OPC, o *Studio.Scada.OPC.3* [1]. Esse servidor padrão do *Indusoft* disponibiliza as *tags* do projeto para comunicação via OPC.

Os requisitos mínimos de configuração para instalação e uso do OPC são listados a seguir:

- Windows 7 (ou acima) 64 bits;
- *OPC foundation Files* – no *workspace* do *MatLab,* digite *opcregister*;
- *Indusoft Studio [2];*
- *MatLab* 2015 ou mais recente.

3.4.1 CONFIGURANDO CLIENTE OPC – *INDUSOFT*

Para configurar o cliente OPC da ferramenta, é necessário ir até a aba de comunicação do projeto, e inserir um serviço *OPC DA 2.5*, como ilustrado na figura 19.

Figura 19: Menu de comunicação do projeto. Fonte: O próprio autor.

Uma vez criada a comunicação, é possível configurar o servidor OPC que se deseja utilizar, assim como as *tags* do projeto que serão utilizadas pelo serviço OPC.

A figura 20 ilustra a configuração utilizada nesse projeto.

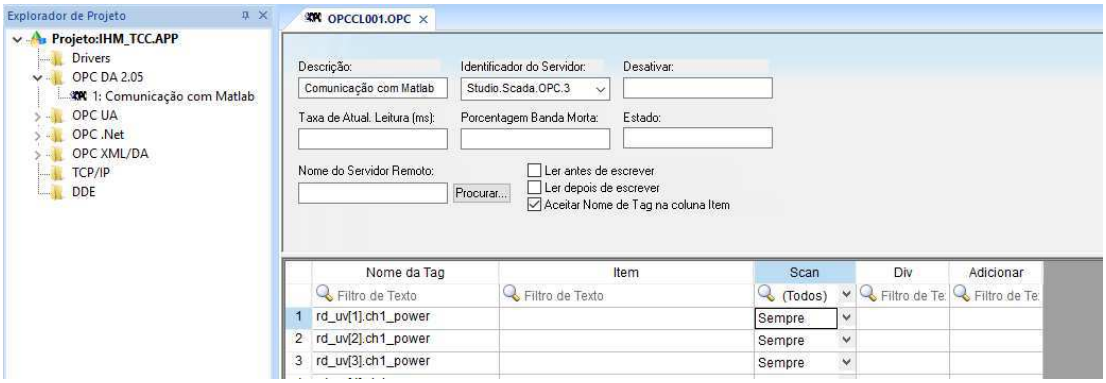


Figura 20: *Worksheet* da configuração da comunicação OPC. Fonte: O próprio autor.

Os parâmetros de configuração utilizados no projeto e, o seu significado, são descritos na tabela 10, a seguir:

Tabela 10: Itens de configuração da comunicação OPC. Fonte: O próprio autor.

Campo	Significado	Exemplo
Descrição	Descreve a conexão	Comunicação Com MatLab
Identificador do Servidor	Seleciona um servidor OPC da lista	Studio.Scada.OPC.3
Desativar	Permite a escolha de uma tag para habilitar/desabilitar a comunicação OPC	*enableOPC*
Nome da Tag	Tags do projeto que serão disponibilizadas pela comunicação OPC	Localização dos radiômetros
Item	Tag advinda da comunicação OPC	Localização da fonte de descarga Parcial

3.4.2 CONFIGURANDO O MATLAB E O SIMULINK

Depois de verificada a existência do *OPC Foundation Files*, deve ser executado o comando *opcregister* no *workspace* do *MatLab.* Em seguida pode-se iniciar o *Simulink* através do comando *simulink* no mesmo *workspace*.

Na tela *library browser* do *Simulink,*ilustrada na figura 21, após selecionar a biblioteca *OPC Toolbox*, seleciona-se, com o botão direito do mouse, o bloco *OPC Configuration* e adiciona-se este bloco a um novo modelo, como ilustrado na figura 22.

Bloco OPC Configuration

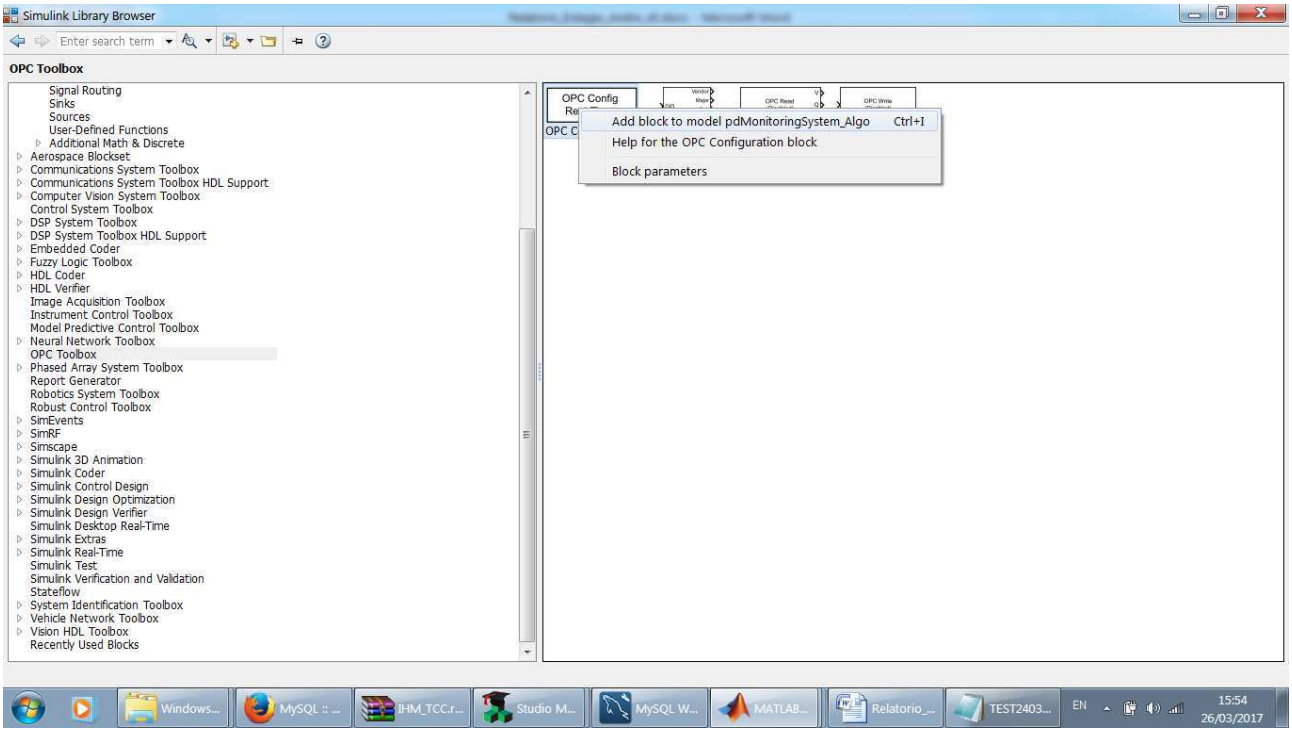


Figura 21: Blocos da biblioteca OPC Toolbox do *Simulink*. Fonte: O próprio autor.

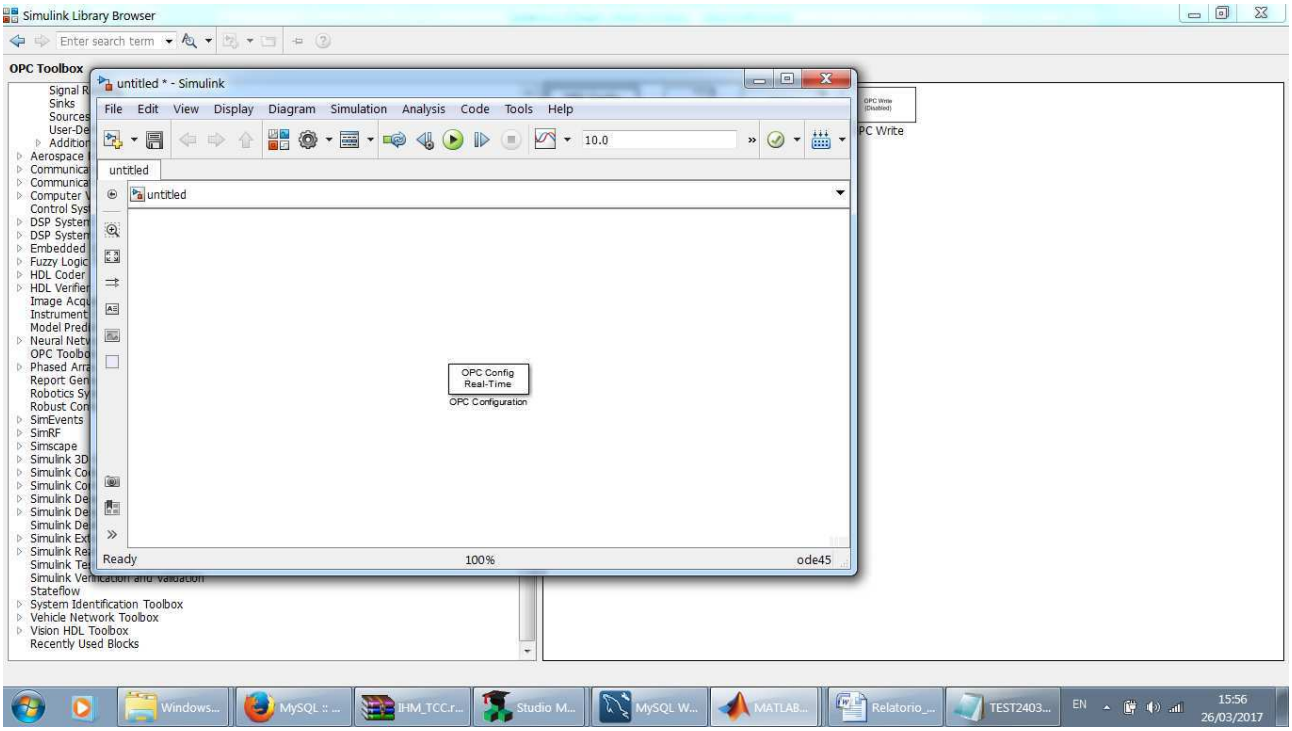


Figura 22: Novo modelo do *Simulink* contendo o bloco de configuração do serviço OPC. Fonte: O próprio autor.

Depois de inserido o bloco no modelo, deve-se configurá-lo. Isto é feito selecionando-se o bloco (clicando duas vezes sobre o bloco) e navegando-se até a opção *Open Block: OPC Conigurati*on. Nesta tela será configurado o servidor OPC para o Cliente *MatLab.* Para isto ativa-se o botão *Configure OPC Clients.*

A partir deste ponto adiciona-se um novo cliente utilizando o endereço de IP da máquina na qual está executando o *Indusoft* – neste caso, a máquina local (*localhost).* Depois destas etapas, são exibidos os resultados ilustrados nas figuras 23 e 24.

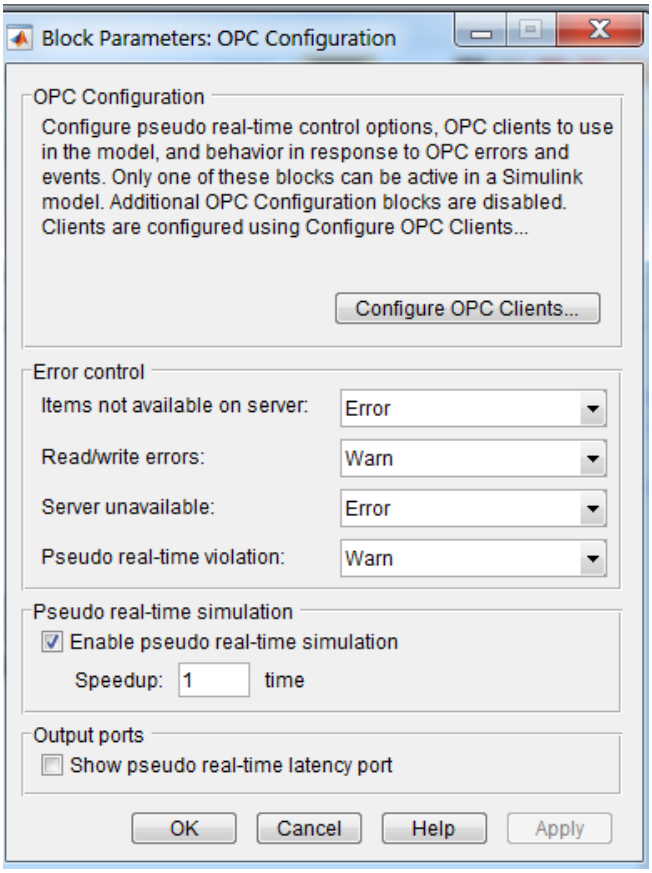


Figura 23: Tela de configuração do bloco *OPC Configuration*. Fonte: O próprio autor.

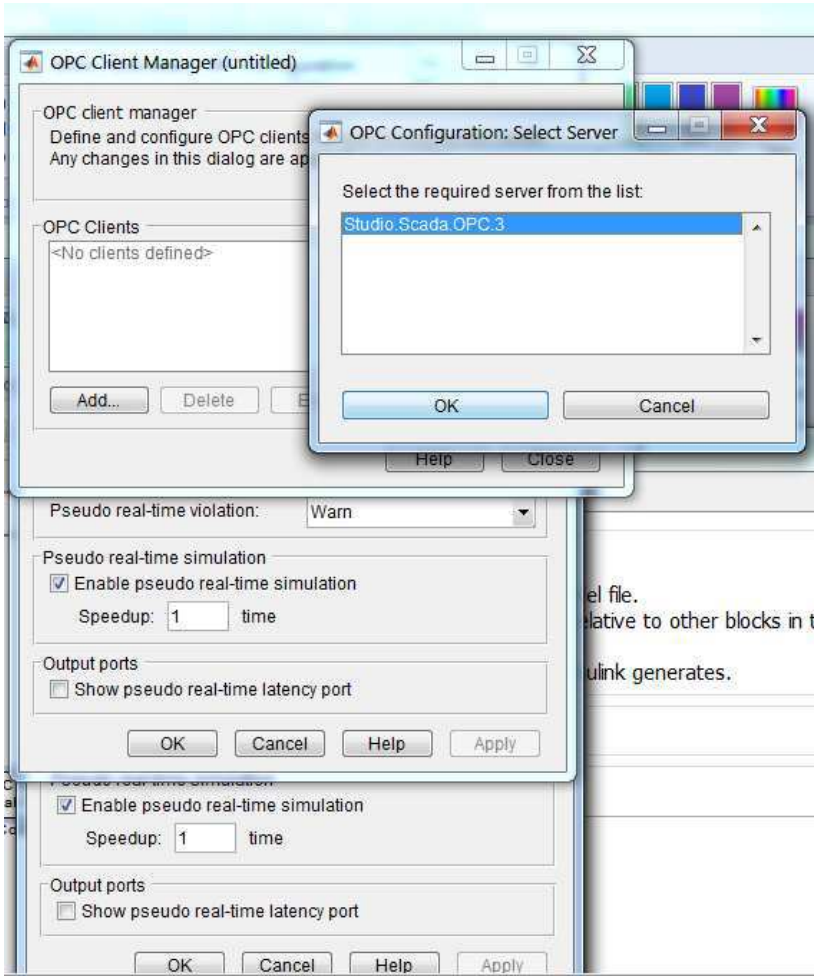


Figura 24: Tela de seleção de servidor OPC – o servidor selecionado é o servidor do *Indusoft*. Fonte: O próprio autor.

Blocos OPC Read e OPC Write

A próxima etapa a ser descrita consiste em inserir os dois blocos utilizados na troca de informações com o *Indusoft*.

Adicionado estes dois blocos ao modelo, acessamos suas configurações. Para o bloco de leitura (*OPC read)*, são definidas as tags que armazenam os valores referentes às potências lidas e à localização (posição no eixo de coordenadas) dos radiômetros e o cliente OPC ao qual ele será conectado. Isto é, o cliente configurado no bloco *OPC Configuration*. A figura 25. Ilustra o bloco de configuração e as respectivas *tags* selecionadas.

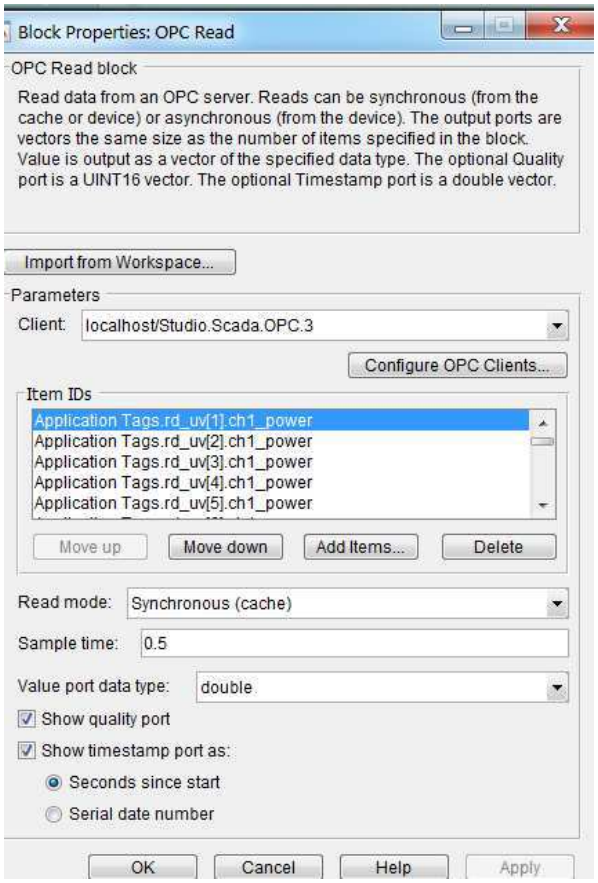


Figura 25: Configurações do bloco *OPC Read* e seleção das tags do *Indusoft* lidas pelo *Simulink.* Fonte: O próprio autor.

Para a configuração do bloco *OPC write*, seleciona-se o mesmo cliente configurado no bloco *OPC Configuration¸* e informam-se as *tags* de escrita – definidas no supervisório.

Concluídas as configurações com sucesso, será exibido um resultado similar ao ilustrado na figura 26.

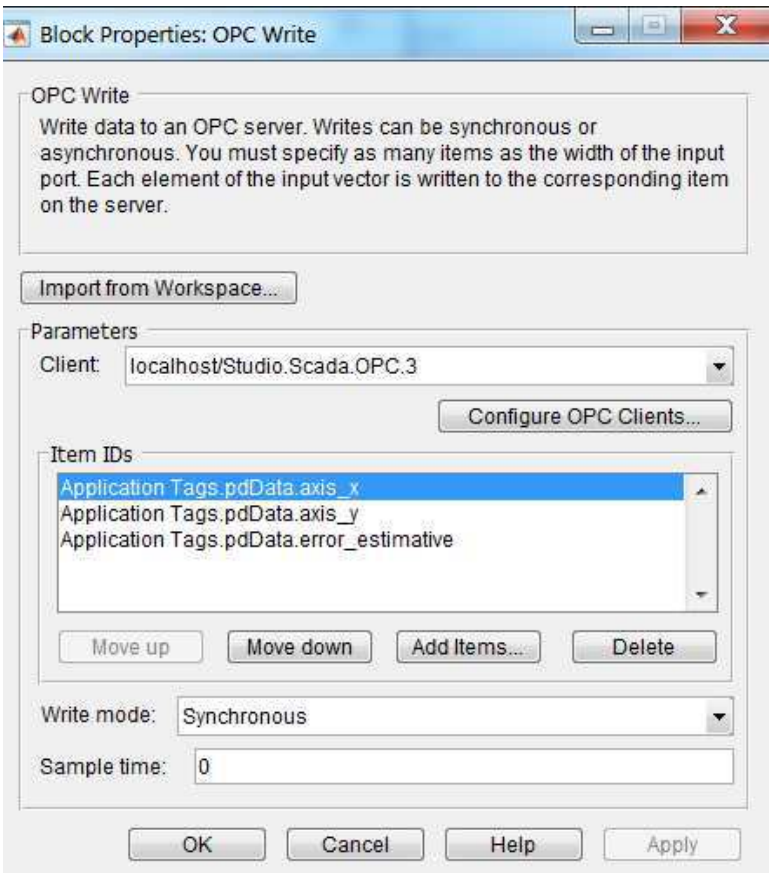


Figura 26: Configuração do bloco *OPC Write* do *Simulink* para escrever nas *tags* do *Indusoft* selecionadas no campo 'Item IDs' de configuração. Fonte: O próprio autor.

Bloco "Interpreted MatLab Function"

Este bloco é utilizado para executar uma função do *MatLab* que esteja disponível em um arquivo do tipo *m*, na biblioteca do *Simulink,* na categoria *User-Defined Functions* – ver figura 27. Este bloco será utilizado para:

- Receber um vetor de entrada ***u[]*** – o qual será a saída do bloco *OPC Read (potencias e localizações dos radiômetros)*;
- Processar o algoritmo de localização *[6]* o qual está listado no Anexo C;
- Produzir um vetor de saída ***y[]*** **–** que consistirá da posição (x,y) estimada da descarga parcial e o erro estimado associado.

A figura 28 apresenta a configuração do bloco – necessitando apenas identificar no campo *MatLab Function* a função que será chamada pelo bloco.

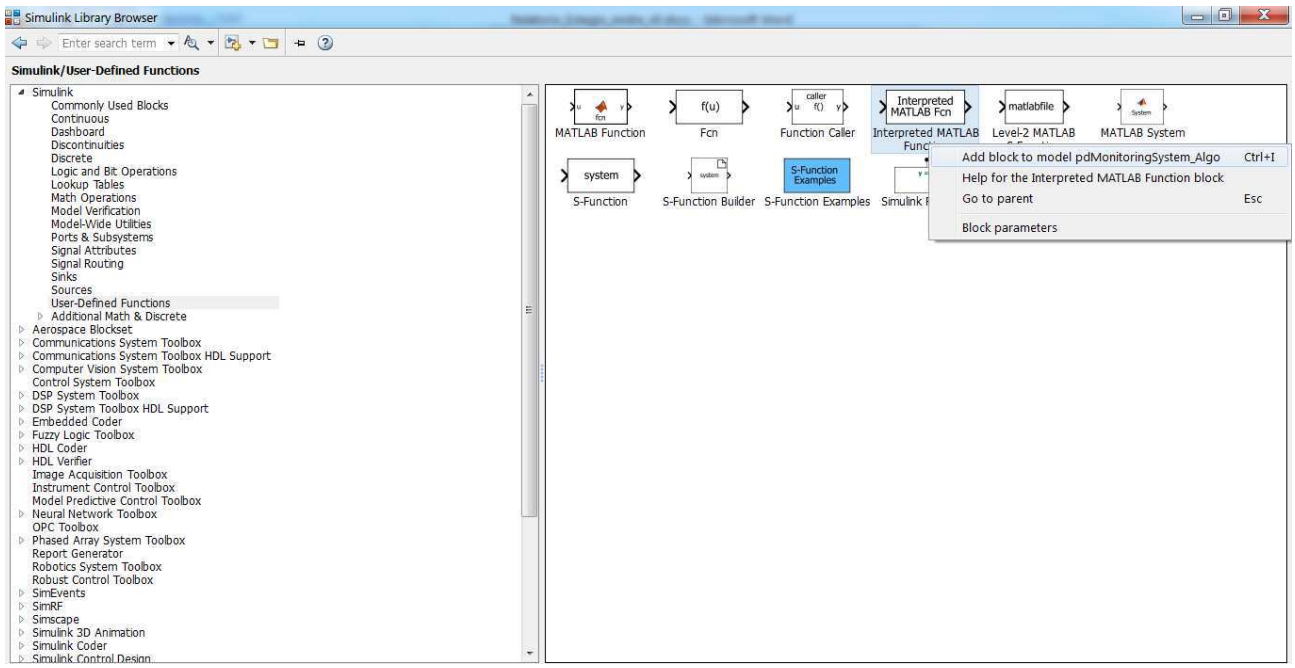


Figura 27: Biblioteca *User-Defined Functions* do *Simulink*, e seleção do bloco *Interpreted MatLab function*. Fonte: O próprio autor.

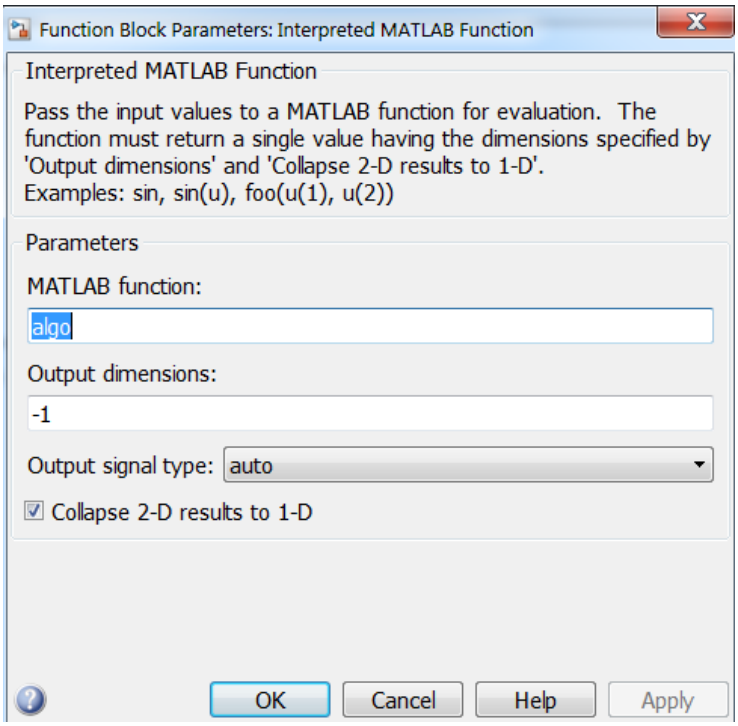


Figura 28: Configuração dos parâmetros do bloco *Interpreted MatLab function.* Fonte: O próprio autor.

Depois de configurar os blocos descritos pode ser iniciada sua conexão. Para isto coloca-se um bloco do tipo *display* para observar os resultados.

A figura 29 ilustra os blocos conectados e o resultado da simulação do algoritmo – já com a conexão com o *Indusoft*, ou seja, com os dados oriundos do *Indusoft*.

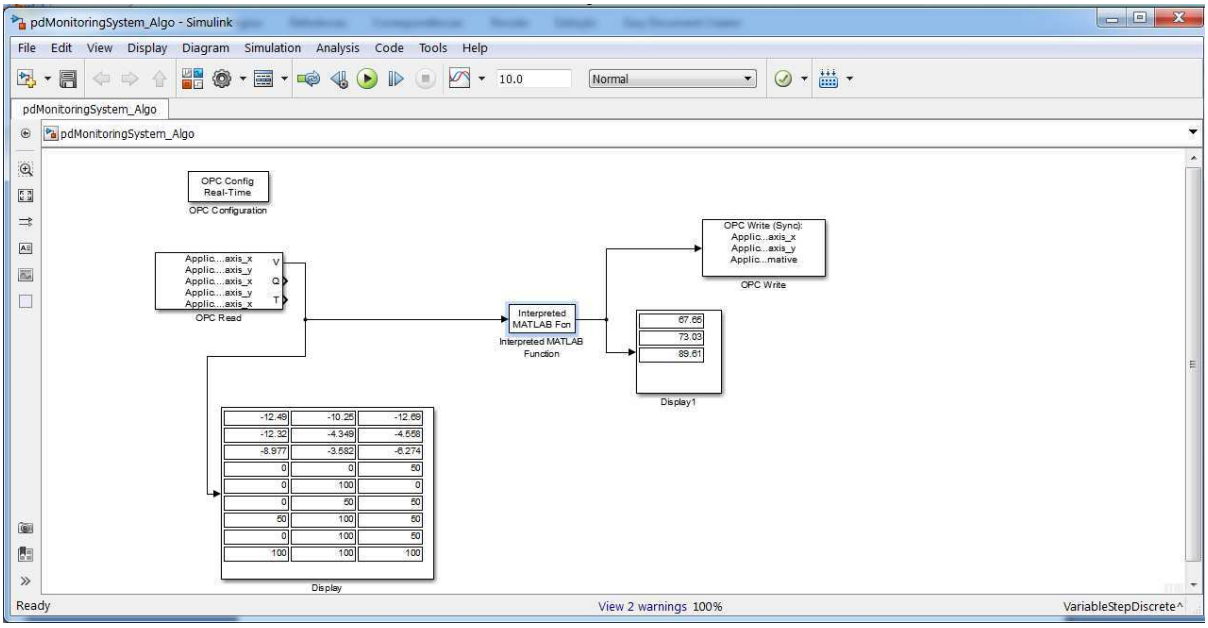


Figura 29: Resultado das ligações dos blocos do *Simulink* e da simulação da comunicação via OPC com o *Indusoft.*
Fonte: O próprio autor.

3.5 O PROJETO DE NOVAS TELAS

Como mencionado na sessão 2, algumas telas do sistema MDP tiveram que ser redesenhadas de modo a contemplar a estrutura de dados mais atual dos sensores, a qual foi discutida na sessão 3.1, enquanto outras telas foram construídas dado que haviam sido apenas especificadas no projeto anterior, Santos [1].

3.5.1 TELA *SNAPSHOT VIEW*

Essa tela representa, de forma gráfica, o comportamento dos radiômetros conectados à rede de sensores. Essa tela foi redesenhada para ampliar o número de radiômetros permitindo a inclusão de de novos radiômetros. Outra funcionalidade adicionada foi o posicionamento dos radiômetros no plano (x,y), em tempo de execução.

A figura 30 ilustra a tela proposta por Santos em [1], enquanto a figura 31 ilustra a nova representação dos radiômetros.

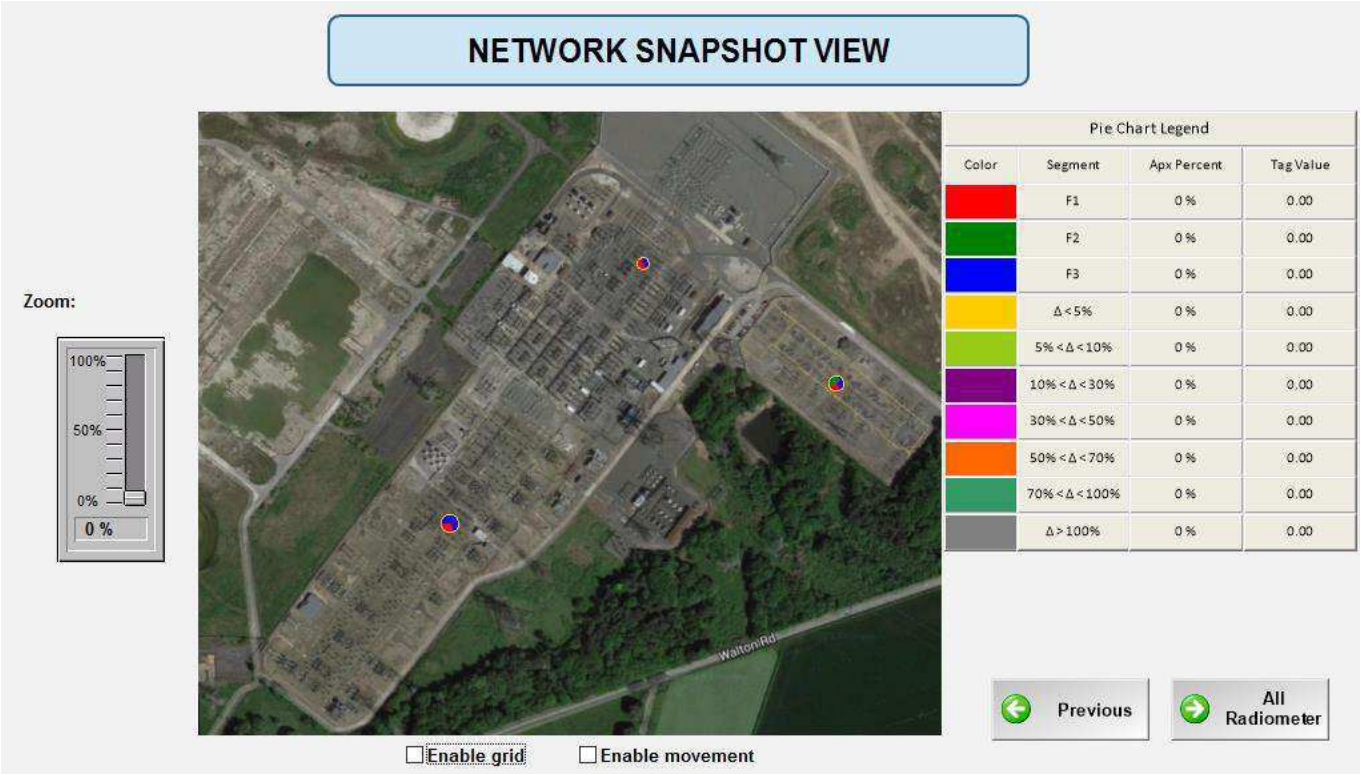


Figura 30: Tela *snapshot view* proposta por [1]. Fonte: Santos [1].

A partir da extração dos requisitos do cliente – a equipe parceira no projeto, a nova representação possibilita a visualização simultânea de até 9 radiômetros.

A primeira etapa no desenvolvimento da nova tela consistiu em redesenhar os objetos que representam os radiômetros que eram representados na forma de *piechart*, - um tipo de objeto *Indusoft – do tipo linked symbol*, que permite ao usuário definir um objeto gráfico que poderá simultaneamente exibir valores de até 10 atributos diferentes.

Como representaremos até 9 radiômetros, e cada radiômetro possui pelo menos 4 atributos diferentes (potência dos 3 canais mais a potência total), seriam necessários no mínimo três objetos do tipo *piechart* para representa-los, tornando inviável sua representação na tela.

Assim, foi desenvolvido um novo objeto, do tipo *linked symbol*, capaz de exibir a variação percentual de potência dos três canais em relação à potência total e, possibilitando exibir todos os radiômetros, como é ilustrado na figura 31.

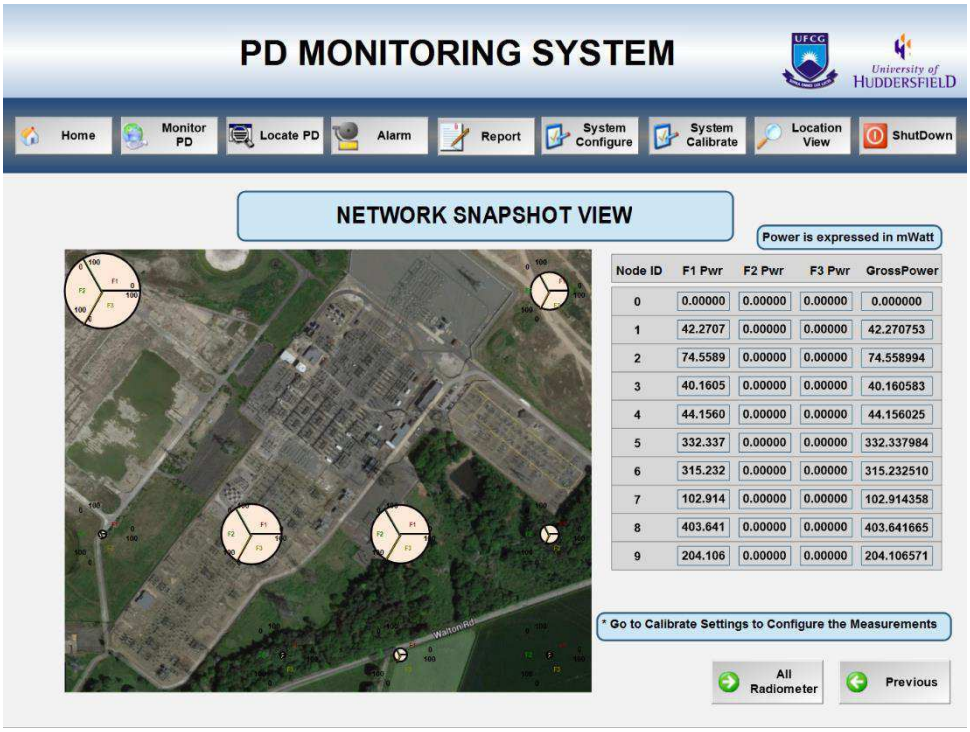


Figura 31: Representação dos radiômetros proposta pelo autor (tela *snapshot view)*. Fonte: O próprio autor.

3.5.2 TELA *HISTORIC DATA – GRÁFICO DE TENDÊNCIA*

No projeto desta tela foi configurado o objeto padrão do *Indusoft,* o *TrendView*, de forma a exibir em tempo real as tendências dos parâmetros de cada radiômetro.

Ao inserir um objeto do tipo *TrendView*, é necessário definir a origem dos dados que serão utilizados para gerar os gráficos de tendências. Ao inserir este objeto, suas configurações são editadas criando um *worksheet* na aba de tarefas do projeto, como é ilustrado na figura 32.

Figura 32: Inserindo um novo *worksheet* do objeto 'gráfico de tendência' do *Indusoft*. Fonte: O próprio autor.

Após inserir o *worksheet*, configura-se a comunicação do gráfico de tendências, como pode ilustrado na figura 33.

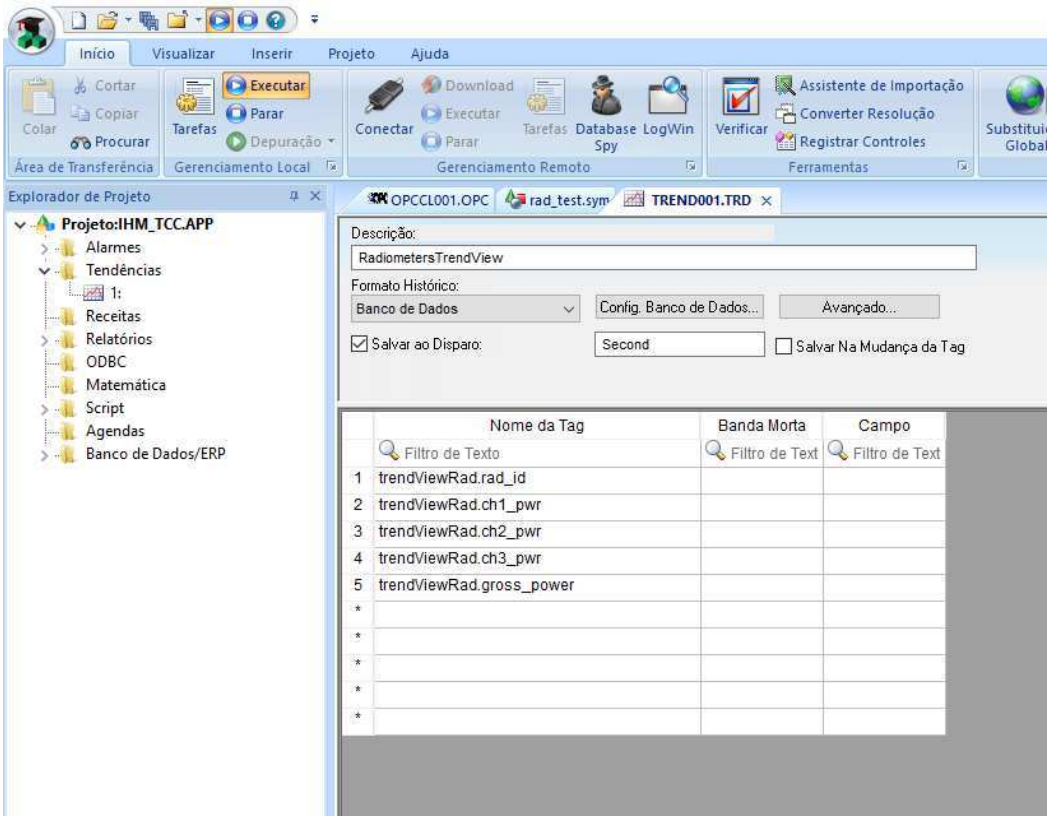


Figura 33: Edição do *worksheet* do *gráfico de tendência* do *Indusoft.* Fonte: O próprio autor.

A lista de *tags* que serão salvas no banco de dados é inserida no campo *Nome da Tag*. Sempre que o objeto for executado em tempo de execução, esses valores serão armazenados no banco de

dados do projeto, de modo que no futuro podem ser recuperados a partir do controle gráfico do objeto, o qual deve ser inserido em alguma tela do projeto.

A figura 34 ilustra o objeto configurado e executado durante a utilização da ferramenta:

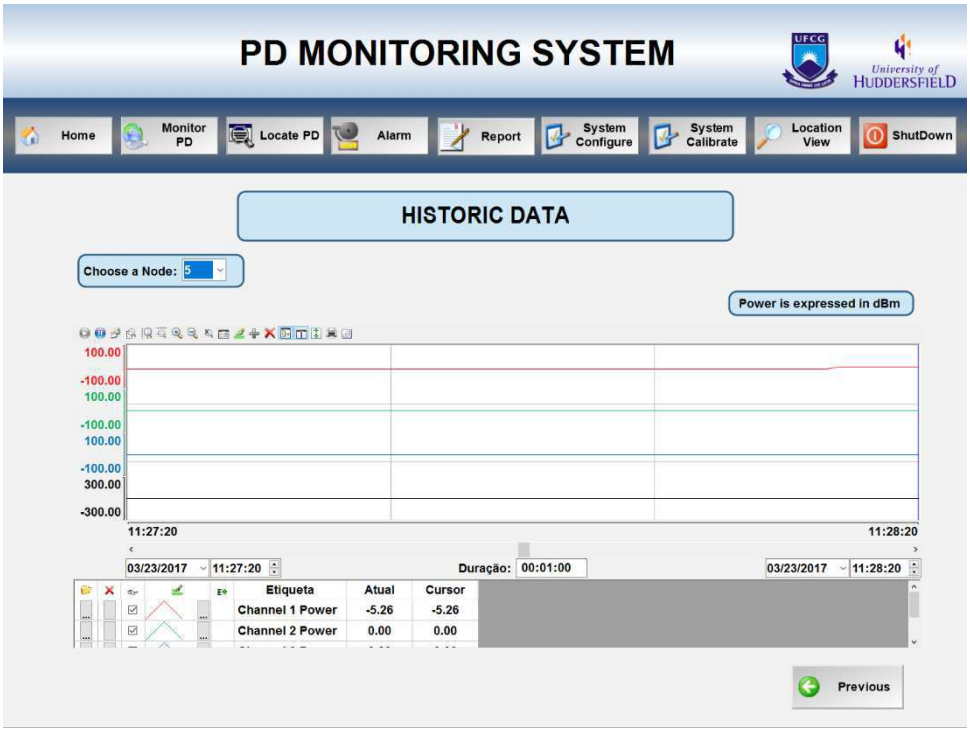


Figura 34: Tela *Historic Data*. Fonte: O próprio autor.

Na tela apresentada na figura 34 é possível selecionar um radiômetro, assim como o período de tempo correspondente às suas leituras, antes de solicitar a visualização do gráfico de tendência.

3.5.3 TELA *REPORT* – GERAÇÃO DE RELATÓRIOS

O relatório não tinha sido construído no projeto anterior. Após a realização de uma reunião com a equipe da universidade de Huddersfield, realizada no dia 17 de março de 2017, foi discutido o conteúdo do relatório:

- Subestação envolvida;
- Período do Relatório;
- Data da elaboração do relatório;
- Identificar a localização estimada da fonte de PD;
- Dados de potência referente aos radiômetros;

O *Indusoft* oferece suporte à geração automática de relatórios nos formatos *Txt, pdf ou html.* Visando simplificar, devido ao prazo do projeto, adotou-se o formato de texto (*txt*) para o relatório.

A tela do *Indusoft*, projetada para o relatório é apresentada na figura 35, enquanto a estrutura do relatório é apresentada na figura 37.

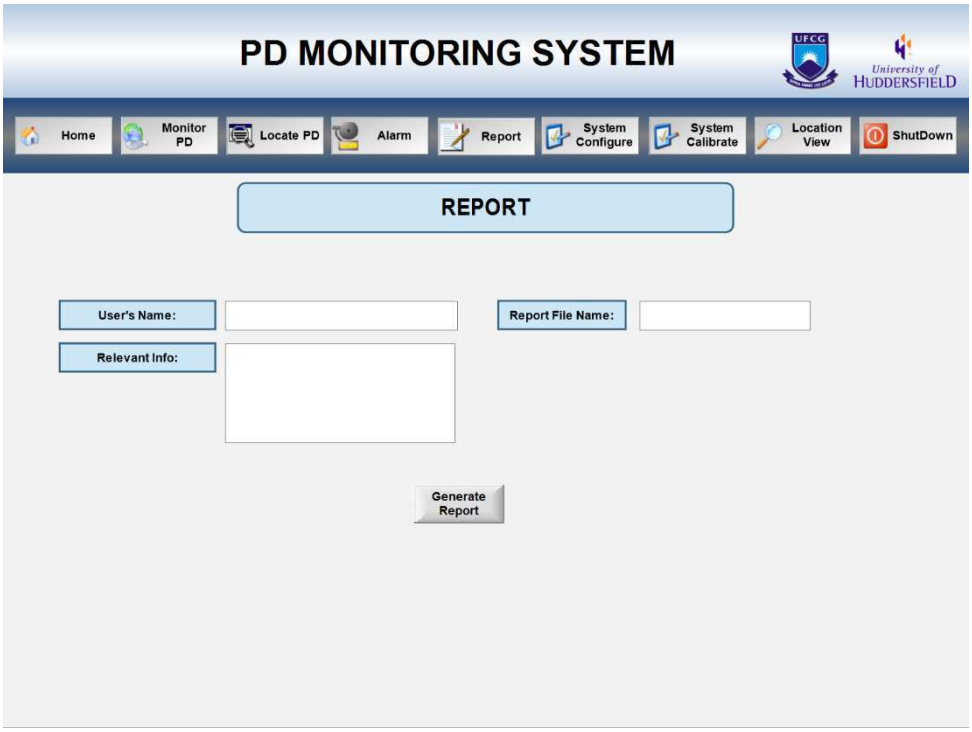


Figura 35: Tela de Geração de *Reports*. Fonte: O próprio autor.

Para criar um modelo de relatório navega-se até a aba de tarefas do projeto e insere-se um novo *worksheet* de relatório, como ilustrado na figura 36.

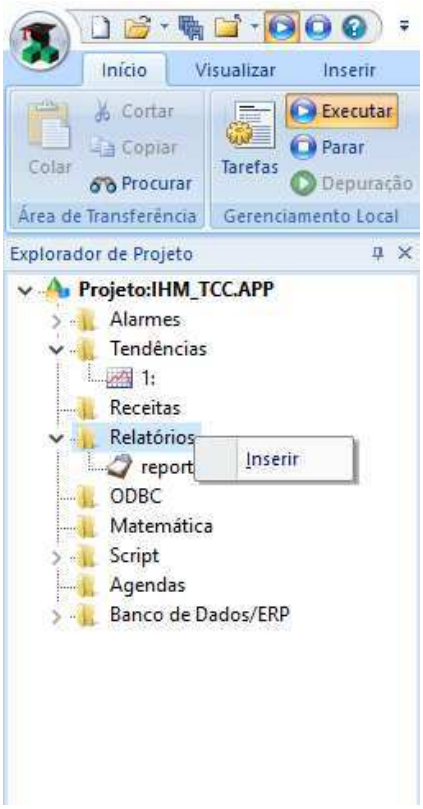


Figura 36: Menu de tarefas – inserindo um novo modelo de relatório. Fonte: O próprio autor.

Após inserir um novo *worksheet,* configura-se o modelo de relatório como ilustrado na figura 37.

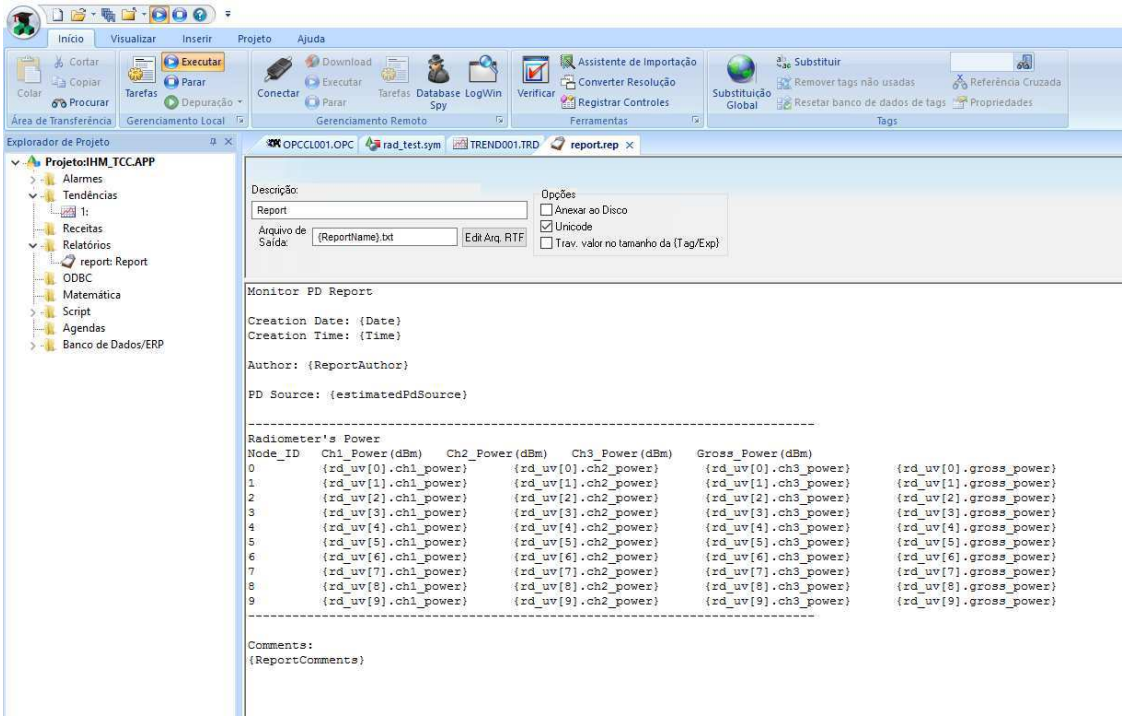


Figura 37: Tela de edição e configuração da geração de relatório. Fonte: O próprio autor.

As *strings* que aparecem entre “{}” representam as *tags* do sistema que serão utilizadas pelo gerador de relatórios do *InduSoft.*

No apêndice B encontra-se um relatório gerado para o sistema de monitoramento utilizando o supervisório.

4 CONSIDERAÇÕES FINAIS

O período de estágio no LIHM propiciou situações nas quais pude colocar em prática diversos conceitos aprendidos em algumas disciplinas específicas da graduação. Disciplinas tais como Informática Industrial – explorou os conceitos projeto de interface gráfica e banco de dados, Instrumentação Eletrônica – definiu conceitos importantes sobre projeto de instrumentos e sensores, Arquiteturas Avançadas de Computadores – constituiu uma base sólida a respeito da estrutura de redes, computadores e periféricos e, Automação Industrial – o contato com supervisório. Esta formação foi importante durante a execução das atividades.

Após concluir o plano de atividades do estágio, pude consolidar conceitos e metodologias de execução de projetos, e me sinto apto profissionalmente a atuar na área de controle e automação, com enfoque em projeto de interfaces homem-máquina, gestão de supervisórios e projeto de banco de dados.

Como sugestões de continuidade para ferramenta proponho que sejam adotadas técnicas mais eficazes na comunicação entre o *Indusoft* e o *algoritmo* de localização das descargas parciais, assim como a inclusão de funcionalidades para gerenciamento de usuários.

BIBLIOGRAFIA

[1] SANTOS, Rafael de Melo Silva. **Desenvolvimento da IHM de um sistema de monitoramento de descargas parciais no ambiente de uma subestação de sistemas elétricos.** Campina Grande, 2016

[2] INDUSOFT, Wondeware. **InduSoft Web Studio 8.0 Training Manual.** Rev. B. Janeiro, 2016.

[3] MATRIKON, Honeywell International INC. **OPC Server/OPC Client architecture.** Disponível em: <https://www.matrikonopc.com/opc-server/opc-client-server.aspx>. Acessado em março de 2017

[4] MYSQL, Oracle Corporation. **Download MySQL server community.** Disponível em: < https://dev.mysql.com/downloads/mysql/>. Acessado em janeiro de 2017.

[5] MYSQL, Oracle Corporation. **Connector/ODBC Architecture.** Disponível em: <https://dev.mysql.com/doc/connector-odbc/en/connector-odbc-architecture.html>. Acessado em março de 2017.

[6] ZHANG, Peng; ZHOU, Jianguo; XU, Yaming. **RSS-Based Source Localization When Path-Loss Model Parameters are Unknown.** IEEE COMMUNICATIONS LETTERS, VOL.18, NO.6, June 2014.

APÊNDICE A – SCRIPT GLOBAL DO CÓDIGO

```
Option Explicit
'Manter a declaração Option Explicit na primeira linha desta interface.

'Procedimentos com escopo global podem ser implementados aqui.
'Variáveis globais NÃO são suportados nesta interface.

Function ReadExcelFile ()
	Dim excelObj, workbookObj, worksheetObj, rows, cols, x, y, dbConfirm, rDate, rTime
	Dim dataValues, sql, dataUpload, dataNotUploaded
	Dim ch1_rcvdPower, ch2_rcvdPower, ch3_rcvdPower, nodeId, numOfPulses, numAvgPulses,
averaged_step_size, timestamp, radRound
	Set excelObj = CreateObject("Excel.Application")
	excelObj.Visible = False
	excelObj.DisplayAlerts = False
	Set workbookObj = excelObj.workbooks.open(""&$xlsFilePath&"")
	Set worksheetObj = excelObj.sheets(1)
	rows = worksheetObj.UsedRange.Rows.count
	cols = worksheetObj.UsedRange.Columns.count
	For x=2 To rows Step 2
		For y=1 To cols Step 7
			timestamp = CStr(worksheetObj.cells(x,y).value)
			rDate = """&$Ncopy(timestamp, 6,4) &"-"& $NCopy(timestamp, 0, 2) &"-"&
$NCopy(timestamp, 3, 2)
			rTime = $NCopy(Timestamp, 11,5) &":00"'
			timestamp = rDate & " " &rTime
			nodeId = worksheetObj.cells(x,y+1).value
			radRound = worksheetObj.cells(x,y+2).value
			numOfPulses = worksheetObj.cells(x,y+3).value
			numAvgPulses = worksheetObj.cells(x,y+4).value
			averaged_step_size = Replace(worksheetObj.cells(x,y+5).value, ",", ".")
			ch1_rcvdPower = Replace(worksheetObj.cells(x,y+6).value, ",", ".")
			ch2_rcvdPower = 0
			ch3_rcvdPower = 0

			sql = "SELECT * FROM data_acquired WHERE  rad_id= " &nodeId &" AND
measurement_date =" &timestamp
			DataValues = timestamp& "," &nodeId& "," &radRound& "," &numOfPulses& ","
&numAvgPulses& "," &ch1_rcvdPower& "," &ch2_rcvdPower&"," &ch3_rcvdPower& ","
&averaged_step_size& ""
			dbConfirm = $DBCursorOpenSQL("mydb", sql, "", $dbLogError)
			If dbConfirm < 0 Then
				$DBCursorClose()

				dbConfirm = $DBInsert("mydb", "data_acquired", dataValues,
"measurement_date, rad_id, round, number_of_pulses, number_of_averaged_pulses, ch1_received_power,
ch2_received_power, ch3_received_power, averaged_step_size", $dbLogError)
			Else
				dataUpload = 1
				dataNotUploaded = " " & dataNotUploaded & " " &timestamp
			End If

		Next
		'$Trace(timestamp& "<- TimeStamp, ch2 power  " &ch2_avgStepSize &"")
	Next
	If dataUpload = 1 Then
		MsgBox("Following data not updated:" &dataNotUploaded& "")
	End If
	excelObj.quit
End Function

Function StartAllVariables ()
```

```
	Dim sqlExpression, cursorPos, rowCount
	$dataUpdateTrigger = 0
	$StartDataCompute = 0
	$xlsFilePath = ""&$GetAppPath() & "SensorData.xlsx"
	sqlExpression = "SELECT * FROM network_configuration_file WHERE id_network = 0"
	cursorPos = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError)
	rowCount = $DBCursorRowCount(cursorPos)
	If rowCount < 0 Then
		SaveStation()
		SaveRadiometerReference()
		SaveGenericRadiometers()
		SaveCalibrationConfig()
		SaveNetworkConfig()
	Else
		Call LoadNetworkConfigs()
		Call LoadRadiometers()
	End If
	$DBCursorClose()
	sqlExpression = "SELECT * FROM data_acquired"
	cursorPos = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError )
	rowCount = $DBCursorRowCount(cursorPos, $dbLogError)
	If rowCount < 1 Then
		'ReadExcelFile()
		End If

End Function

Function LoadNetworkConfigs()
	Dim sqlExpression, cursorPos, rowCount

	sqlExpression = "SELECT * FROM network_configuration_file WHERE id_network = 0"
	cursorPos = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError)
	rowCount = $DBCursorRowCount(cursorPos)

	If rowCount > 0 Then
		$NetworkConfig.idNetwork = $DBCursorGetValue(cursorPos, "id_network")
		$NetworkConfig.angle_orientation = $DBCursorGetValue(cursorPos, "angle_orientation")
		$NetworkConfig.id_local_station = $DBCursorGetValue(cursorPos, "id_local_station")
		$NetworkConfig.id_origins_lat = $DBCursorGetValue(cursorPos, "id_origins_lat")
		$NetworkConfig.id_origins_long = $DBCursorGetValue(cursorPos, "id_origins_long")
		$NetworkConfig.location_name = $DBCursorGetValue(cursorPos, "location_name")
		$NetworkConfig.radiometer_total = $DBCursorGetValue(cursorPos, "radiometer_total")
		$NetworkConfig.id_calibration = $DBCursorGetValue(cursorPos, "id_calibration")
		$DBCursorClose(cursorPos)
		Call LoadLocation($NetworkConfig.id_origins_lat, $NetworkConfig.id_origins_long, 1)
		Call LoadCalibrationConfigs($NetworkConfig.id_calibration)
		Call LoadStation($NetworkConfig.id_local_station)
	Else
		MsgBox("Atenção, ocorreu o seguinte erro: " &$dbLogError)
	End If
End Function

Function LoadRadiometers()
	Dim sqlExpression, posCursor, rowCount, x, y
	sqlExpression = "SELECT * FROM radiometer_file "
	posCursor = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError)
	rowCount = $DBCursorRowCount(posCursor)

	If rowCount > 0 Then
		For x=0 To rowCount Step 1
			$RadiometersFile[x].axis_x = $DBCursorGetValue(posCursor, "axis_x",
$dbLogError)
```

```
                        $RadiometersFile[x].axis_y = $DBCursorGetValue(posCursor, "axis_y",
$dbLogError)
                        $RadiometersFile[x].id_lat = $DBCursorGetValue(posCursor, "id_rad_lat",
$dbLogError)
                        $RadiometersFile[x].id_long = $DBCursorGetValue(posCursor, "id_rad_long",
$dbLogError)
                        $RadiometersFile[x].idMoto  = $DBCursorGetValue(posCursor, "id_moto",
$dbLogError)
                        $RadiometersFile[x].idRadiometer = $DBCursorGetValue(posCursor,
"id_radiometer", $dbLogError)
                        $RadiometersFile[x].max_power = $DBCursorGetValue(posCursor, "max_power",
$dbLogError)
                        $RadiometersFile[x].last_power = $DBCursorGetValue(posCursor, "last_power",
$dbLogError)
                        If  $DBCursorGetValue(posCursor, "flag_active", $dbLogError) > 0 Then
                                $RadiometersFile[x].active_flag = $DBCursorGetValue(posCursor,
"flag_active", $dbLogError)
                        End If
                        $DBCursorNext(posCursor)
                Next
                For y=0 To 9
                        Call LoadLocation($RadiometersFile[y].id_lat, $RadiometersFile[y].id_long, 2+y)
                Next
                $DBCursorClose(posCursor)
        End If

End Function

Function LoadStation(id_station)
        Dim sqlExpression, posCursor, rowCount
        sqlExpression = "SELECT * FROM station_file WHERE id_local_station = " &id_station& " "
        posCursor = $DBCursorOpenSQL("mydb", sqlExpression)
        rowCount = $DBCursorRowCount(posCursor)
        If rowCount >0 Then
                $Station.Name = $DBCursorGetValue(posCursor, "station_name", $dbLogError)
                $Station.id_station = $DBCursorGetValue(posCursor, "id_local_station", $dbLogError)
                $Station.id_lat = $DBCursorGetValue(posCursor, "id_st_latitude", $dbLogError)
                $Station.id_long = $DBCursorGetValue(posCursor, "id_st_longitude", $dbLogError)
                $DBCursorClose(posCursor)
                Call LoadLocation ($Station.id_lat, $Station.id_long, 0)
        End If

End Function

Function LoadCalibrationConfigs(id_calibration)
        Dim sqlExpression, posCursor, rowCount
        sqlExpression = "SELECT * FROM calibration_configuration_file WHERE id_calibration = "
&id_calibration& " "
        posCursor = $DBCursorOpenSQL("mydb", sqlExpression)
        rowCount = $DBCursorRowCount(posCursor)
        If rowCount >0 Then
                $CalibrationConfig.id_calibration = $DBCursorGetValue(posCursor, "id_calibration")
                $CalibrationConfig.last_calibration = $DBCursorGetValue(posCursor, "last_calibration")
                $CalibrationConfig.measurement_duration_tm = $DBCursorGetValue(posCursor,
"measurement_duration_tm")
                $CalibrationConfig.period_tu = $DBCursorGetValue(posCursor, "period_tu")
                $CalibrationConfig.radiometer_id = $DBCursorGetValue(posCursor, "radiometer_id")
                $CalibrationConfig.source_intensity = $DBCursorGetValue(posCursor, "source_intensity")
                $DBCursorClose(posCursor)
        End If
End Function

Function LoadLocation (location_id_lat, location_id_long, destinationId)
```

```
	Dim sqlExpression, cursorPos, rowCount, lat_h, lat_m, lat_s, long_h, long_m, long_s
	sqlExpression = "SELECT * from local_coordinates WHERE id_local_coordinates = "
&location_id_lat& " "
	cursorPos = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError)
	rowCount = $DBCursorRowCount(cursorPos)
	If rowCount > 0 Then
		$LocationFile[destinationId].lat_d = $DBCursorGetValue(cursorPos, "degrees")
		$LocationFile[destinationId].lat_m = $DBCursorGetValue(cursorPos, "minutes")
		$LocationFile[destinationId].lat_s = $DBCursorGetValue(cursorPos, "seconds")'
	End If
	$DBCursorClose(cursorPos)
	sqlExpression = "SELECT * from local_coordinates WHERE id_local_coordinates = "
&location_id_long& " "
	cursorPos = $DBCursorOpenSQL("mydb", sqlExpression, "", $dbLogError)
	rowCount = $DBCursorRowCount(cursorPos)
	If rowCount > 0 Then
		$LocationFile[destinationId].long_d = $DBCursorGetValue(cursorPos, "degrees")
		$LocationFile[destinationId].long_m = $DBCursorGetValue(cursorPos, "minutes")
		$LocationFile[destinationId].long_s = $DBCursorGetValue(cursorPos, "seconds")
	End If
	$DBCursorClose(cursorPos)
End Function

Function SaveNetworkConfig()
	Dim dbConfirm, DataValues

	$NetworkConfig.idNetwork = 0
	$NetworkConfig.angle_orientation = 0.0
	$NetworkConfig.location_name = "Station Name"
	$NetworkConfig.radiometer_total = 10
	$NetworkConfig.id_origins_lat = 2
	$NetworkConfig.id_origins_long = 3
	$LocationFile[1].id_lat =2
	$LocationFile[1].id_long = 3
	$LocationFile[1].locationSourceName = "NetworkConfig"
	$LocationFile[1].locationIDSource = 0
	$LocationFile[1].lat_d = 33.333
	$LocationFile[1].lat_m = 22.222
	$LocationFile[1].lat_s = 11.111
	$LocationFile[1].long_d = 11.111
	$LocationFile[1].long_m = 33.333
	$LocationFile[1].long_s = 22.222
	Call SaveLocation($LocationFile[1].lat_d, $LocationFile[1].lat_m, $LocationFile[1].lat_s,
$LocationFile[1].id_lat)
	Call SaveLocation($LocationFile[1].long_d, $LocationFile[1].long_m, $LocationFile[1].long_s,
$LocationFile[1].id_long)
	DataValues = $NetworkConfig.idNetwork & ",'" &$NetworkConfig.location_name& "'," 
&$NetworkConfig.angle_orientation& "," &$NetworkConfig.radiometer_total& ","
&$NetworkConfig.id_local_station& "," &$NetworkConfig.id_origins_lat& ","
&$NetworkConfig.id_origins_long& "," &$CalibrationConfig.id_calibration& " "
	dbConfirm = $DBInsert ("mydb", "network_configuration_file", DataValues, "id_network,
location_name, angle_orientation, radiometer_total, id_local_station, id_origins_reference_lat,
id_origins_reference_long, id_calibration", $dbLogError)
End Function

Function SaveCalibrationConfig()

	Dim dbConfirm, DataValues, uDate
	$CalibrationConfig.id_calibration = 0
	$CalibrationConfig.measurement_duration_tm = 1.0
	$CalibrationConfig.period_tu = 3600
	$CalibrationConfig.radiometer_id = 0
	uDate = $ClockGetDate($GetClock()     )
```

```
	$CalibrationConfig.date_last_calibration = """&$Ncopy(uDate, 6,4) &"-"& $NCopy(uDate, 0, 2) &"-"&
$NCopy(uDate, 3, 2) & """
	$CalibrationConfig.selected_source = 1
	DataValues = $CalibrationConfig.id_calibration& "," &$CalibrationConfig.date_last_calibration& ","
&$CalibrationConfig.measurement_duration_tm& "," &$CalibrationConfig.period_tu& ","
&$CalibrationConfig.radiometer_id& "," &$CalibrationConfig.source_intensity & ","
&$CalibrationConfig.selected_source& " "
	dbConfirm = $DBInsert("mydb", "calibration_configuration_file", DataValues, "id_calibration,
last_calibration, measurement_duration_tm, period_tu, radiometer_id, source_intensity, source_selection",
$dbLogError)

End Function

Function SaveRadiometerReference()
	Dim dbConfirm, DataValues
	$RadiometersFile[0].axis_x = 0.0
	$RadiometersFile[0].axis_y = 0.0
	$RadiometersFile[0].idMoto = 0
	$RadiometersFile[0].idRadiometer =0
	$RadiometersFile[0].id_lat = 4
	$RadiometersFile[0].id_long = 5
	$RadiometersFile[0].max_power = 0.0
	$RadiometersFile[0].active_flag = 1

	$LocationFile[2].id_lat = 4
	$LocationFile[2].id_long = 5
	$LocationFile[2].lat_d = 10.10
	$LocationFile[2].lat_m = 10.10
	$LocationFile[2].lat_s = 10.10
	$LocationFile[2].long_d = 10.10
	$LocationFile[2].long_m = 10.10
	$LocationFile[2].long_s = 10.10

	$LocationFile[2].locationIDSource = 0
	$LocationFile[2].locationSourceName = "RadiometersFile"

	Call SaveLocation(10.10, 10.10, 10.10, $RadiometersFile[0].id_lat)
	Call SaveLocation(10.10, 10.10, 10.10, $RadiometersFile[0].id_long)
	DataValues = $RadiometersFile[0].idRadiometer& "," &$RadiometersFile[0].idMoto& ","
&$RadiometersFile[0].id_lat& "," &$RadiometersFile[0].id_long& "," &$RadiometersFile[0].axis_x& ","
&$RadiometersFile[0].axis_y& "," &$RadiometersFile[0].max_power& ", " &$RadiometersFile[0].active_flag&
",0"
	dbConfirm = $DBInsert("mydb", "radiometer_file", DataValues, "id_radiometer, id_moto,
id_rad_latitude, id_rad_longitude, axis_x, axis_y, max_power, flag_active, last_calculated_power",
$dbLogError)
End Function

Function SaveGenericRadiometers()
	Dim dbConfirm, DataValues, x
	Dim location_count
	location_count = CInt(6)
		For x=1 To 9 Step 1
		$RadiometersFile[x].axis_x = 0.0
		$RadiometersFile[x].axis_y = 0.0
		$RadiometersFile[x].idMoto = x
		$RadiometersFile[x].idRadiometer = x
		$RadiometersFile[x].id_lat = CInt(location_count)
		$RadiometersFile[x].active_flag = 0
		location_count =  CInt(location_count+1)
		$RadiometersFile[x].id_long =  CInt(location_count)
		location_count =  CInt(location_count+1)
		$RadiometersFile[x].max_power = 0.0
```

```
                $LocationFile[2+x].id_lat = 5 +x
                $LocationFile[2+x].id_long = 6 + x
                $LocationFile[2+x].lat_d = 10.10
                $LocationFile[2+x].lat_m = 10.10
                $LocationFile[2+x].lat_s = 10.10
                $LocationFile[2+x].long_d = 10.10
                $LocationFile[2+x].long_m = 10.10
                $LocationFile[2+x].long_s = 10.10

                $LocationFile[2+x].locationIDSource = x
                $LocationFile[2+x].locationSourceName = "RadiometersFile"


                Call SaveLocation(10.10, 10.10, 10.10, $RadiometersFile[x].id_lat)
                Call SaveLocation(10.10, 10.10, 10.10, $RadiometersFile[x].id_long)
                DataValues = $RadiometersFile[x].idRadiometer& "," &$RadiometersFile[x].idMoto& ","
&$RadiometersFile[x].id_lat& "," &$RadiometersFile[x].id_long& "," &$RadiometersFile[x].axis_x& ","
&$RadiometersFile[x].axis_y& "," &$RadiometersFile[x].max_power& ", " &$RadiometersFile[x].active_flag&
",0"
                dbConfirm = $DBInsert("mydb", "radiometer_file", DataValues, "id_radiometer, id_moto,
id_rad_latitude, id_rad_longitude, axis_x, axis_y, max_power, flag_active, last_calculated_power",
$dbLogError)
        Next
End Function

'Standard: id for latitude and longitude to Station are 0 and 1
'Standard: id for latitude and longitude to Origins_Reference are 2 and 3
'Standard: id for latitude and longitude to Radiometers starts at 4 and 5
Function SaveLocation (dg, mn, sc, id)
        Dim loc_dg, loc_mn, loc_sc, loc_id, sql, dbConfirm

        loc_dg = $StrFromReal (dg, 2, "f")
        loc_mn = $StrFromReal(mn, 2, "f")
        loc_sc = $StrFromReal(sc, 2, "f")
        loc_id = id
        dbConfirm = $DBInsert("mydb", "local_coordinates", " "& loc_id & "," & loc_dg & "," & loc_mn & ","  &
loc_sc & "", "id_local_coordinates, degrees, minutes, seconds", $dbLogError)

End Function

'Standard id is 0
Function SaveStation()

        Dim dbConfirm, sql
        Dim DataValues
        $Station.id_station = 0
        $Station.Name = "Station"
        $Station.id_lat = 0
        $Station.id_long = 1
        $LocationFile[0].id_lat =0
        $LocationFile[0].id_long = 1
        $LocationFile[0].locationSourceName = "Station"
        $LocationFile[0].locationIDSource = 0
        $LocationFile[0].lat_d = 0.333
        $LocationFile[0].lat_m = 22.222
        $LocationFile[0].lat_s = 11.111
        $LocationFile[0].long_d = 11.111
        $LocationFile[0].long_m = 33.333
        $LocationFile[0].long_s = 22.222
        DataValues = $Station.id_station & ",' " &$Station.Name & " ', " &$Station.id_lat& ", "
&$Station.id_long & " "
        Call SaveLocation($LocationFile[0].lat_d, $LocationFile[0].lat_m, $LocationFile[0].lat_s,
$LocationFile[0].id_lat)
```

```
        Call SaveLocation($LocationFile[0].long_d, $LocationFile[0].long_m, $LocationFile[0].long_s,
$LocationFile[0].id_long)
   dbConfirm = $DBInsert( "mydb", "station_file", DataValues, "id_local_station, station_name, id_st_latitude,
id_st_longitude", $dbLogError)
        $Trace($dbLogError & dbConfirm)
End Function

'The group variable identifies the object which was selected to update location: 0 -> Station, 1 ->
NetworkConifg, 2 (Equal or greater than) -> RadiometerFile
Function UpdateLocation (dg, mn, sc, id_local, group)
        Dim loc_dg, loc_mn, loc_sc, dbConfirm, dataValues
        If id_local Mod 2 = 0 Then
                $LocationFile[group].lat_d = dg
                $LocationFile[group].lat_m = mn
                $LocationFile[group].lat_s = sc
        Else
                $LocationFile[group].long_d = dg
                $LocationFile[group].long_m = mn
                $LocationFile[group].long_s = sc
        End If
        loc_dg = $StrFromReal (dg, 2, "f")
        loc_mn = $StrFromReal(mn, 2, "f")
        loc_sc = $StrFromReal(sc, 2, "f")
        dataValues =  loc_dg & "," & loc_mn & ","  & loc_sc & ""

        dbConfirm = $DBUpdate("mydb", "local_coordinates", dataValues, "degrees, minutes, seconds",
"id_local_coordinates =" &id_local& "", $dbLogError)
        If dbConfirm < 0 Then
                Call MsgBox ("Unable to update BD data (LOCAL COORDINATES). Error Message: "
&$dbLogError& "", vbOKOnly+vbExclamation, "Error while updating to DataBase.")
        End If
        $Trace($dbLogError)
End Function

Function UpdateRadiometer()
        Dim dataValues, dbConfirm, locationUpdateConfirm
        dataValues = $RadiometerEdit.idMoto& "," &$RadiometerEdit.axis_x& "," &$RadiometerEdit.axis_y&
"," &$RadiometerEdit.max_power& "," &$RadiometerEdit.active_flag& ""
        locationUpdateConfirm = UpdateLocation($Rad_Location_Config_Screen.lat_d,
$Rad_Location_Config_Screen.lat_m, $Rad_Location_Config_Screen.lat_s,
$Rad_Location_Config_Screen.id_lat, 2+$RadiometerEdit.idRadiometer)
        locationUpdateConfirm = UpdateLocation($Rad_Location_Config_Screen.long_d,
$Rad_Location_Config_Screen.long_m, $Rad_Location_Config_Screen.long_s,
$Rad_Location_Config_Screen.id_long,2+$RadiometerEdit.idRadiometer)
        dbConfirm = $DBUpdate("mydb", "radiometer_file", dataValues, "id_moto, axis_x, axis_y,
max_power, flag_active", "id_radiometer =" &$RadiometerEdit.idRadiometer & "",  $dbLogError)
        If dbConfirm < 0 Then
                Call MsgBox ("Unable to update BD data. Error Message (LOCAL COORDINATES): "
&$dbLogError& "", vbOKOnly+vbExclamation, "Error while updating to DataBase.")
        End If
        $Trace($dbLogError)
        $RadiometersFile[$RadiometerEdit.idRadiometer].active_flag = $RadiometerEdit.active_flag
        $RadiometersFile[$RadiometerEdit.idRadiometer].axis_x = $RadiometerEdit.axis_x
        $RadiometersFile[$RadiometerEdit.idRadiometer].axis_y = $RadiometerEdit.axis_y
        $RadiometersFile[$RadiometerEdit.idRadiometer].max_power = $RadiometerEdit.max_power
        $Trace($dbLogError)
        UpdateRadiometer = dbConfirm
End Function

Function UpdateNetworkConfigs ()
        Dim DataValues, dbConfirm
```

```
		DataValues = $NetworkConfig.angle_orientation& ",'" &$NetworkConfig.location_name& "'"
		dbConfirm = $DBUpdate("mydb", "network_configuration_file", DataValues, "angle_orientation,
location_name", "id_network =" &$NetworkConfig.idNetwork& "", $dbLogError)
		If dbConfirm > 0 Then
			Call UpdateLocation($LocationFile[1].lat_d, $LocationFile[1].lat_m, $LocationFile[1].lat_s,
$LocationFile[1].id_lat, 1)
			Call UpdateLocation($LocationFile[1].long_d, $LocationFile[1].long_m,
$LocationFile[1].long_s, $LocationFile[1].id_long, 1)
			DataValues = "'"&$Station.Name&"'"
			dbConfirm = $DBUpdate("mydb", "station_file", DataValues, "station_name",
"id_local_station="&$Station.id_station&"","")
		Else
			Call MsgBox("Unable to update data (Network Settings) ", vbOKOnly+vbExclamation,
"Network Settings: Error.")
		End If
		If dbConfirm > 0 Then
			Call UpdateLocation($LocationFile[0].lat_d, $LocationFile[0].lat_m, $LocationFile[0].lat_s,
$LocationFile[0].id_lat, 0)
			Call UpdateLocation($LocationFile[0].long_d, $LocationFile[0].long_m,
$LocationFile[0].long_s, $LocationFile[0].id_long, 0)
			Call MsgBox("Data saved successfully!", vbOKOnly+vbExclamation, "Network Settings:
Success.")
		Else
			Call MsgBox("Unable to update data (Network Settings) ", vbOKOnly+vbExclamation,
"Station Settings: Error.")
		End If
End Function

Function UpdateCalibrationConfigs()
		Dim dataValues, dbConfirm, locationUpdateConfirm, sqlDate, uDate
		uDate = $ClockGetDate($GetClock()     )
		$CalibrationConfig.date_last_calibration = "'"&$Ncopy(uDate, 6,4) &"-"& $NCopy(uDate, 0, 2) &"-"&
$NCopy(uDate, 3, 2) & "'"
		$Trace($CalibrationConfig.date_last_calibration)
		dataValues = $CalibrationConfig.measurement_duration_tm & "," &$CalibrationConfig.period_tu& ","
&$CalibrationConfig.source_intensity& "," &$CalibrationConfig.radiometer_id & ","
&$CalibrationConfig.date_last_calibration&"," &$CalibrationConfig.selected_source& ""
		dbConfirm = $DBUpdate("mydb", "calibration_configuration_file", dataValues,
"measurement_duration_tm, period_tu, source_intensity, radiometer_id, last_calibration, source_selection",
"id_calibration =" &$CalibrationConfig.id_calibration&"", $dbLogError)
		If dbConfirm >0 Then
			Call MsgBox("Data saved successfully!", vbOKOnly+vbExclamation, "Calibration Settings:
Success.")
		Else
			Call MsgBox("Unable to update data (Calibration Settings) " &$dbLogError,
vbOKOnly+vbExclamation, "Calibration Settings: Error.")
			$Trace(""&$dbLogError)
		End If
End Function

Function GetFileName( myDir, myFilter )
  ' Written by Rob van der Woude
  ' http://www.robvanderwoude.com

  ' Standard housekeeping
  Dim objDialog

  ' Create a dialog object
  Set objDialog = CreateObject( "UserAccounts.CommonDialog" )

  ' Check arguments and use defaults when necessary
  If myDir = "" Then
    ' Default initial folder is "My Documents"
```

```
    objDialog.InitialDir = CreateObject( "WScript.Shell" ).SpecialFolders( "MyDocuments" )
  Else
    ' Use the specified initial folder
    objDialog.InitialDir = myDir
  End If
  If myFilter = "" Then
    ' Default file filter is "All files"
    objDialog.Filter = "All files|*.*"
  Else
    ' Use the specified file filter
    objDialog.Filter = myFilter
  End If

  ' Open the dialog and return the selected file name
  If objDialog.ShowOpen Then
    GetFileName = objDialog.FileName
  Else
    GetFileName = ""
  End If
End Function
```

APÊNDICE B – RELATÓRIO GERADO PELO SISTEMA MDP

Monitor PD Report

Creation Date: 03/26/2017
Creation Time: 15:33:47

Author: Andre

PD Source: P(x,y) (67.648945,73.031461)
Estimated Error: 89.608501

Radiometer's Power

Node_ID	Ch1_Power(dBm)	Ch2_Power(dBm)	Ch3_Power(dBm)	Gross_Power(dBm)
0	0.000000	0.000000	0.000000	0.000000
1	-12.490500	0.000000	0.000000	-12.490500
2	-10.250000	0.000000	0.000000	-10.250000
3	-12.692700	0.000000	0.000000	-12.692700
4	-12.318300	0.000000	0.000000	-12.318300
5	-4.349270	0.000000	0.000000	-4.349270
6	-4.557900	0.000000	0.000000	-4.557900
7	-8.977490	0.000000	0.000000	-8.977490
8	-3.581850	0.000000	0.000000	-3.581850
9	-6.274030	0.000000	0.000000	-6.274030

Comments:

Testing the main functions of Pd Monitoring System

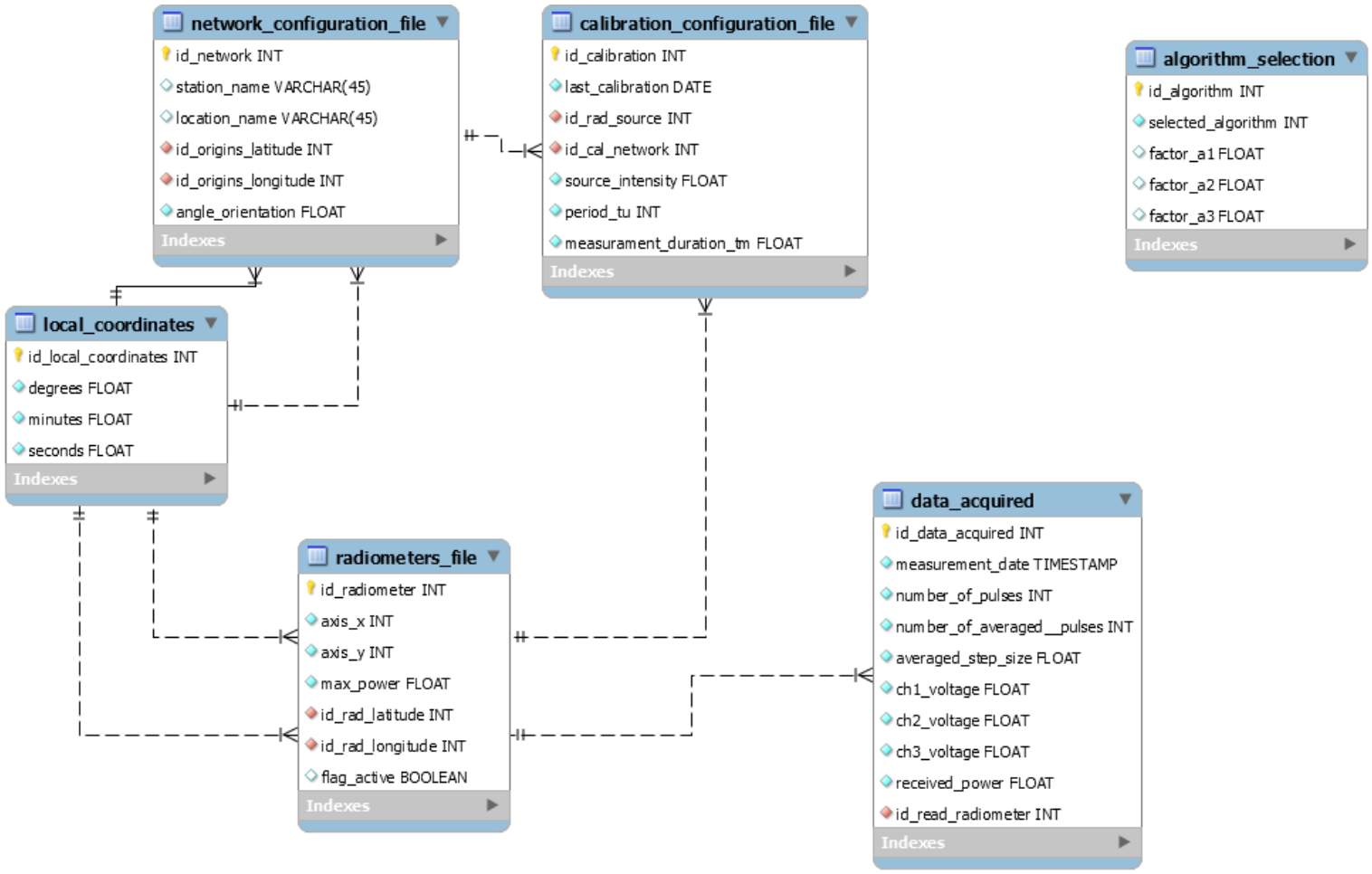
APÊNDICE C– ALGORITMO DE LOCALIZAÇÃO DE PD (ZHANG [6])

```
function y = algo(u)
clc; close all;
gridSize = 20;
transLoc = [6 8];
d0 = 1;
r0 = -60;
min = 1;
max = 5;
p1=u(1); p4=u(4); p7=u(7);
p2=u(2); p5=u(5); p8=u(8);
p3=u(3); p6=u(6); p9=u(9);
x1 = u(10); y1 = u(11); x2 = u(12); y2 = u(13); x3 = u(14); y3 = u(15);
x4 = u(16); y4 = u(17); x5 = u(18); y5 = u(19); x6 = u(20); y6 = u(21);
x7 = u(22); y7 = u(23); x8 = u(24); y8 = u(25); x9 = u(26); y9 = u(27);
step = 0.05;
X = [x1,y1; x2,y2; x3,y3; x4,y4; x5,y5; x6,y6; x7,y7; x8,y8; x9,y9];
%X = [0,18; 18,18;0,9;9,9;18,9; 0,0;9,0;18,0]; % Receivers Locations
n = size(X,1);
R= [p1, p2, p3, p4, p5, p6, p7, p8, p9];
%R = [-9.41,  -13.18,-3.07,  1.28,   -9.82,  -9.75,  -9.01,  -14.69];
transLocEst = RatioMethod(X,R,n,min,max,step);
transLocEst
plot(X(:,1),X(:,2),'ko','MarkerSize',12,'lineWidth',2);
grid on
hold on
plot(transLoc(1,1)   ,   transLoc(1,2),'b^','MarkerSize',12,'lineWidth',2);
plot(transLocEst(1,1),transLocEst(1,2),'rv','MarkerSize',12,'lineWidth',2);
legend('Reciever locations','Transmitter true location','Transmitter estimated location',...
      'Location','Best')
```

```
% Compute the Root Mean Squred Error
estError = sqrt(sum((transLocEst - transLoc).^2));
title(['Estimation error is ',num2str(round((estError),2)),' metre'])
y = transLocEst;
y(3) = estError;
function estLoc = RatioMethod(X,R,n,min,max,step)
index=min;
P=zeros(n,1);
dis=zeros(n,1);
A=zeros(n-1,3);
b=zeros(n-1,1);
LSE=+inf;
optX=zeros(3,1);
for i= 1:n
   P(i,1) = 10^(R(i)/10);
end
while index < max
   for i = 1:n-1
      A(i,1) = 2 * ( P(i)^(2/index)*X(i,1) - P(n)^(2/index)*X(n,1) );
      A(i,2) = 2 * ( P(i)^(2/index)*X(i,2) - P(n)^(2/index)*X(n,2) );
      A(i,3) = P(n)^(2/index) - P(i)^(2/index);
      b(i,1) = P(i)^(2/index)*( X(i,1)^2 + X(i,2)^2 ) -  P(n)^(2/index)*( X(n,1)^2 + X(n,2)^2
);
   end
   Y = A\b;
   for i = 1:n
      dis(i)= sqrt( (X(i,1)-Y(1,1))^2 + (X(i,2)-Y(2,1))^2 );
   end
   sum=0;
   for i = 2:n
      tmp = ((dis(1)/dis(i))^index - P(i)/P(1))^2;
      sum = sum + tmp;
   end
   if sum < LSE
      LSE = sum
```

```
        optX = Y
    end
    index = index + step
end
estLoc = [optX(1,1),optX(2,1)];
```

APÊNDICE D – MODELO EER DO BD


network_configuration_file
id_network INT
station_name VARCHAR(45)
location_name VARCHAR(45)
id_origins_latitude INT
id_origins_longitude INT
angle_orientation FLOAT
Indexes
calibration_configuration_file
id_calibration INT
last_calibration DATE
id_rad_source INT
id_cal_network INT
source_intensity FLOAT
period_tu INT
measurament_duration_tm FLOAT
Indexes
algorithm_selection
id_algorithm INT
selected_algorithm INT
factor_a1 FLOAT
factor_a2 FLOAT
factor_a3 FLOAT
Indexes
local_coordinates
id_local_coordinates INT
degrees FLOAT
minutes FLOAT
seconds FLOAT
Indexes
radiometers_file
id_radiometer INT
axis_x INT
axis_y INT
max_power FLOAT
id_rad_latitude INT
id_rad_longitude INT
flag_active BOOLEAN
Indexes
data_acquired
id_data_acquired INT
measurement_date TIMESTAMP
number_of_pulses INT
number_of_averaged__pulses INT
averaged_step_size FLOAT
ch1_voltage FLOAT
ch2_voltage FLOAT
ch3_voltage FLOAT
received_power FLOAT
id_read_radiometer INT
Indexes