Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

10 de dezembro de 2021

Conexão SCADA - DNP3 - Religador

João Figueiredo Lopes
Victor de Barros Barreto Pamplona Côrtes

PUC RIO

Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Conexão SCADA - DNP3 - Religador

Alunos: **João Figueiredo Lopes**
Victor de Barros Barreto
Pamplona Côrtes

Orientador: **Marco Antonio Grivet Mattoso Maia**

Trabalho apresentado como requisito parcial para conclusão do curso de Engenharia Elétrica da Pontifícia Universidade Estadual do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Agradecimento aos nossos pais, amigos e professores por sempre nos apoiarem ao longo de nossa jornada.

Resumo

Este Trabalho de Conclusão de Curso consiste no desenvolvimento de uma conexão e envio de informações de um servidor SCADA até um religador, elemento fundamental para uma rede elétrica inteligente. A seguir serão introduzidos tanto o funcionamento e conceito do sistema, quanto os elementos presentes neste processo. Além disso, utilizou-se o protocolo de comunicação DNP3 para concretizar a conexão dos nossos servidores SCADA com o religador. Esse elemento também será explicado mais à frente considerando todas as camadas de comunicação existentes ao longo do procedimento. O Trabalho também consiste na simulação deste sistema descrito. Utilizando softwares adquiridos online, foi possível operar remotamente um simulador de religador, operando, até mesmo, na capacidade de desligamento ou religamento do mesmo. Ademais, visualizou-se a comunicação do religador com o servidor SCADA fazendo com que fosse visto na prática o que será descrito posteriormente na teoria. Vale ressaltar que, devido à dificuldade de manter o software do religador (adquirimos apenas a versão temporária), desenvolvemos um sistema SCADA-Arduino para que fosse possível visualizar com mais clareza o funcionamento da operação do SCADA. Mais adiante, este projeto será detalhado.

Palavras-chave: Redes Elétricas Inteligentes, SCADA, DNP3, Religador, Modbus, Arduino

Abstract

SCADA - DNP3 - Recloser Connection

This Course Conclusion Thesis consists of making and maintaining a connection in which information is sent from a SCADA server to a recloser, a fundamental element for the smart grid. After that, both the general workings and concept of the system in question, will be introduced. To accomplish the task of transferring information from a SCADA server to a recloser, the DNP3 communication protocol was used. This element will also be explained further as all the layers of communication that exist throughout the procedure will be tackled. This project also consists of simulating this described system. Utilising software acquired online, it was possible to remotely operate a recloser simulator. Furthermore, this simulation of the communication between the recloser and the SCADA server was both observed and documented, facilitating the link between the theory described in this thesis what is to be expected in real world applications. It is noteworthy that, due to the difficulty of maintaining the recloser simulator software (only the temporary version was acquired), a SCADA-Arduino system was implemented, utilizing the Modbus Serial communication, so that it would be possible to grasp in a deeper way the full behaviour of a SCADA operation.

Key-words: Smart Grid, SCADA, DNP3, Recloser , Modbus, Arduino

Sumário

Lista de Figuras

1 Introdução

a Conceito de Smart-grid

Com o intuito de fortalecer a certificação de fornecimento de energia às residências, comércios ou indústrias, vemos atualmente um grande movimento para tornar as redes de transmissão elétrica cada vez mais inteligentes, exatamente para que se possa haver um monitoramento e maior confiabilidade e segurança deste setor. Considerando as mudanças que podem ser analisadas, temos, principalmente, a automação de subestações, um avanço no sistema de medição e a automação dos religadores, que, por sua vez, serão o foco deste projeto.

Vemos hoje em dia o desenvolvimento das Smart-grids tendo que balancear a demanda considerando que a transmissão de energia seja feita da maneira mais eficiente possível, aceitando e contornando as flutuações da rede e, além disso, deve-se evitar com que surtos elétricos afetem alguma região. Tendo isso em mente, é possível observar o desenvolvimento de locais com geração distribuída. Este sistema permitiu com que fosse criado o conceito de "prossumidor". Este é, como o nome já indica, o produtor que também é consumidor. Podemos observar este ocorrido quando o cliente final possui, por exemplo, um sistema com painéis fotovoltaicos e, assim, gera a energia que é fornecida para sua residência e o excedente é injetado na rede da concessionária.

Com esta ideia, percebe-se que a rede está em um processo de transição considerando todas as melhorias aplicadas à mesma. É necessário que as empresas responsáveis por este fluxo de informação e aplicações estejam cientes que esta mudança de ambiente é variável, ou seja, o desenvolvimento deve estar diretamente ligado às possibilidades de mudança e, mesmo assim, demonstrando confiabilidade, eficiência e qualidade do sistema de transmissão.

Trazendo à tona os religadores, podemos considerá-los um dos equipamentos mais importantes quando estamos tratando de redes elétricas inteligentes. Basicamente, este elemento permite com que as empresas gerenciadoras reconheçam e interrompam correntes elétricas em caso de falhas. Isto faz com que elas possuam as ferramentas para reconfigurar automaticamente a rede, isolando-a da falta e restaurando com maior facilidade o fornecimento de energia. Com isso, torna-se mais viável manter o serviço ao seu nível ótimo minimizando, assim, as interrupções.

No entanto, é necessário um sistema de altíssimo grau de qualidade para que as análises sejam feitas minuciosamente. Para este projeto utilizaremos o sistema SCADA que, de forma geral, providencia um conjunto de dados desde a área de atuação até o servidor através de um RTUs(Remote Terminal Units - Unidades de Terminais Remotos) ou IEDs(Intelligent Electrical Device - Equipamentos Eletrônicos Inteligentes) e, com o auxílio de sensores, é possível transmitir as informações obtidas para a rede de internet conforme o usuário mestre desejar.

A utilização desses religadores em linhas de transmissão estão tornando os dados muito mais acessíveis e confiáveis como jamais antes visto. Com isso, as equipes que operam nas redes elétricas têm a capacidade de tomar decisões muito mais precisas e seguras tendo em vista a vasta quantidade de informação que este equipamento nos provê. Abaixo podemos analisar alguns benefícios obtidos neste processo:

- Facilitação da análise de medidores de energia elétrica e das próprias redes;
- Interrupções de transmissão menores e em curtos períodos de tempo;
- Otimização da geração de eletricidade e distribuição da mesma, diminuindo a necessidade de efeitos poluidores para geração de energia;
- Diminuição significativa no custo de manutenção do sistema de transmissão.

Neste projeto focaremos no religador como equipamento a ser visualizado no software utilizado.

b Idealização do sistema completo SCADA-Religador

Antes de iniciarmos a conexão, foi necessário determinar quem seria o mestre, o operador que envia comandos requisitados, e quem seria o slave, o elemento que efetivamente produz as tarefas determinadas pelo mestre. Com isso, percebe-se que o religador deveria operar como slave, por ser o equipamento que gera informações a quem requisita e opera da maneira que o mesmo exige. Tendo isto em mente, temos que o servidor SCADA será tido como mestre pois, através dele, faremos os pedidos ao religador enviando comandos já conhecidos por nós, os agentes do sistema.

Inicialmente utilizou-se o software "DNP3 simulator" para se conectar ao simulador de religador, denominado "Test Harness", da Triangle MicroWorks. Para isso, foi necessário incitar o religador como sendo

o escravo da conexão e, com isso, o simulador de DNP3 trabalhou como mestre para que fosse possível visualizar os requisitos ao Test Harness. Nesta situação, por ser apenas um simulador do protocolo de comunicação, não foi possível operar o religador.

Posteriormente, o software SPA1 foi utilizado para efetivamente operar o simulador de religador no Test Harness. Nessa situação, a relação mestre-escravo se deu conforme a descrita anteriormente.

c Idealização do sistema completo SCADA-Arduíno

Como já descrito, nos aprofundamos na ideia de operar com mais versatilidade um sistema SCADA desenvolvendo um projeto com a utilização de um Arduino. Isso se deu, principalmente, devido à necessidade de ter mais base e conhecimento neste tipo de modelo de supervisão de equipamentos.

Para isso, utilizou-se o equipamento Arduíno Uno que se encaixou nos nossos requisitos para operar como equipamento escravo, fornecendo informações aos servidores SCADA.

Após operar o software SPA1 com o sistema de religador, vimos que seria necessário atuar com algum sistema SCADA mais complexo e diversificado devido à simplicidade e limite de funcionalidades do SPA1. Com isso, após pesquisas, definimos o SCADABR como o servidor apropriado ao nosso projeto. Este trabalha como um servidor local SCADA que nos permite, sem a utilização de internet, visualizar equipamentos e operá-los conforme desejamos.

Para efetuar a comunicação SCADA-Arduíno, o protocolo de comunicação utilizado neste caso é, diferentemente do projeto SCADA-religador, o Modbus, aprofundado mais à frente no relatório.

2 Sistema SCADA

SCADA (sistema de controle de supervisão e aquisição de dados) refere-se à combinação entre controle e aquisição de dados. SCADA é essencialmente a coleta de informações por meio de uma UTR (unidade terminal remota), transferindo-a de volta para uma central de controle, onde são realizadas análises, essas informações são exibidas para operador para que assim seja realizado o controle/supervisão exercida pelo mesmo, que manuseia vários botões de controle, enviado comandos necessários para o bom funcionamento do sistema. SCADA é amplamente utilizado para fazer o controle de supervisão e aquisição de dados em plantas, fábricas e energia e instalações de geração.

Hardware: é essencialmente uma série de "unidades terminais remotas" (ou UTRs/RTUs) que coletam dados de processos em um sistema e enviam esses dados para o "mestre", através do sistema de comunicações (no caso estudado DNP3). O mestre então exibe os dados recebidos e permite que um operador/supervisor execute um controle remoto.

Em um sistema SCADA mais complexo, existem essencialmente cinco níveis:

1. Instrumentação em nível de campo e dispositivos de controle;
2. Terminais de Marshalling e RTUs(usados para empacotar sinais em aplicações de automação de uma maneira; claramente organizada. A fiação de um grande número de condutores, muitas vezes é organizada/diferenciada pelo uso de níveis codificados por cores. Junto com a fiação frontal, o que ajuda a evitar erros de conexão);
3. Sistema de comunicação;
4. A (s) estação (ões) mestre (s);
5. O departamento de tecnologia da informação comercial (TI) ou processamento de dados;
6. sistema de computador.

A RTU está ligada sensores analógicos e digitais de campo. O sistema de comunicação fornece o caminho para a comunicação entre os estação mestre e os sites remotos. Este sistema de comunicação pode ser de fio, fibra óptica, rádio, linha telefônica, microondas e possivelmente até satélite. A(s) estação (ções) mestre(s) ou possivelmente submestres reúne(m) dados das várias UTRs e, em geral, fornecer uma interface de operação para exibição de informações e controle dos locais remotos.

Software: pode ser dividido em dois tipos, software proprietário ou software aberto. Empresas desenvolvem software proprietário para a comunicação com seu hardware específico, por meio de drivers. O principal problema com esses sistemas é a dependência do fornecedor do sistema. Os sistemas de software aberto, se tornaram populares devido a grande gama de compatibilidade e assim interoperabilidade que eles trazem para o sistema. Interoperabilidade é a capacidade de misturar diferentes equipamentos dos fabricantes no mesmo sistema.

Citect e WonderWare são dois exemplos de software aberto disponíveis no para sistemas SCADA.

1. Interfaces de usuário;
2. Visores gráficos;
3. Alarmes(opcional);
4. Tendências(opcional);
5. Interface RTU (e PLC);
6. Acesso aos dados armazenados;
7. Base de dados;
8. Tolerância a falhas e redundância;
9. Processamento distribuído cliente / servidor.

As principais vantagens do SCADA são:

- Computadores podem registrar e armazenar uma grande quantidade de dados;
- Os dados podem ser exibidos de qualquer maneira que o usuário requeira;
- Milhares de sensores em uma ampla área podem ser conectados ao sistema;
- O operador pode incorporar simulações de dados reais ao sistema;

- Muitos tipos de dados podem ser coletados das UTRs;
- Os dados podem ser visualizados de qualquer lugar, não apenas no local;
- Na maioria das vezes é fácil e barato adicionar um dispositivo como um interruptor ou um sensor.

As principais desvantagens do SCADA são:

- O sistema SCADA é mais complicado do que o sistema sensor-painel;
- Diferentes habilidades operacionais em outros casos, nesse caso são necessárias, como analistas de sistema e programador;
- Com milhares de sensores, ainda há muitas conexões e fios para monitorar/organizar;
- O armazenamento de dados é mínimo e difícil de gerenciar caso não exista um banco de dados separadamente;
- O operador pode ver apenas até o PLC;
- A quantidade e o tipo de dados são pequenos e simples;
- A instalação de sensores adicionais torna-se progressivamente mais difícil à medida que o sistema cresce;
- A reconfiguração do sistema torna-se extremamente difícil.

Em 1988, a Comissão Eletrotécnica Internacional (IEC) começou a publicar uma norma intitulado 'IEC 870 Telecontrol equipment and systems'. Este foi desenvolvido de forma hierárquica para definir completamente um protocolo aberto para comunicações SCADA. O protocolo foi definido em termos de sistemas abertos modelo de interconexão (OSI) usando um subconjunto mínimo de camadas; o físico, data link e Application Layer. Isso incluiu a definição detalhada da estrutura da mensagem no nível data link e um conjunto de estruturas de dados do Application Layer para que os fabricantes possam usar o protocolo para criar sistemas que sejam capazes de interoperar.

O atual padrão para protocolos de transmissão é IEC 60870.5. embora esse protocolo inclua tipos de dados gerais que podem ser usados em qualquer aplicação SCADA, o uso do IEC 60870 é direcionado à indústria elétrica.

Ambos o IEC 870 protocolo DNP3 foram desenvolvidos no mesmo intervalo de tempo, com objetivos similares. DNP3 é um protocolo aberto desenvolvido pela Harris Controls Division, Distributed Automation. Embora o protocolo seja geralmente mencionado como DNP3 ou protocolo de rede distribuída Versão 3.0, é o padrão de telecomunicações que define as comunicações entre estações mestre, unidades remotas (UTRs/RTUs) e outros dispositivos eletrônicos inteligentes (IEDs). Foi desenvolvido para alcançar "interoperabilidade"(mencionada anteriormente na parte de software) entre sistemas na concessionária de energia elétrica, óleo e gás, água,etc. Desde sua criação para a indústria de distribuição elétrica na América, DNP3 ganhou aceitação significativa, DNP3 é suportado por um grande número de fornecedores e usuários em infraestrutura elétrica, hídrica e outras indústrias na América do Norte, América do Sul, África do Sul, Ásia, Austrália e Nova Zelândia. Na Europa O DNP3 concorre com o IEC 60870-5. No entanto, o IEC está atualmente restrito à indústria de distribuição de energia, enquanto DNP3 abrange as indústrias de petróleo e gás, água,etc.

Uma característica chave do protocolo DNP3 é que ele é um padrão de protocolo aberto e é foi adotado por um número significativo de fabricantes de equipamentos.Tanto DNP3 quanto IEC 60870-5 foram projetados especificamente para SCADA , assim possuem como uma das principais características a transmissão pacotes de dados pequenos e de forma confiável. Por este motivo, eles se diferenciam de protocolos de uso geral, como FTP, que faz parte do TCP / IP, os quais podem enviar arquivos grandes, mas de uma forma que geralmente não é tão adequada para o controle SCADA.

3 Modelo OSI

Em 1978, a ISO, diante da proliferação de sistemas fechados, definiu uma Referência de Modelo para comunicação entre sistemas abertos (ISO 7498), que se tornou conhecido como o modelo de interconexão de sistemas abertos ou simplesmente como o modelo OSI. OSI é essencialmente uma estrutura de gerenciamento de comunicações de dados, que quebra as comunicações de dados em uma hierarquia gerenciável de sete camadas. Cada camada tem um propósito definido e faz interface com as camadas acima e abaixo dela. Ao estabelecer padrões para cada camada, alguma flexibilidade é permitida para que os designers do sistema possam desenvolver protocolos para cada camada independente uma da outra. Em conformidade com os padrões OSI, um sistema é capaz de comunicar-se com qualquer outro sistema compatível, em qualquer lugar do mundo. Deve-se perceber desde o início que o modelo de referência OSI não é um protocolo mas sim conjunto de regras para padronizar e direcionar/estruturar o protocolo. A estrutura do modelo OSI especifica e define as funções ou serviços que devem ser fornecidos em cada uma das sete camadas. O modelo até agora mencionado foi desenvolvido, com o principal objetivo de fornecer uma estrutura para coordenação do desenvolvimento de futuros padrões de comunicação e permitir que os existentes padrões possam evoluir dentro de uma estrutura comum.

Dentro desse modelo, a mensagem definida por um primeiro sistema (emissor) é dividida e encapsulada adequadamente a cada camada, da primeira até a última (camada física). Ao chegar na camada física, a mensagem é recebida(parte a parte, uma vez que foi segmentada pelas camadas) por um segundo sistema(receptor) e reconstruída, camada a camada no segundo sistema.

A seguir vamos explicitar e explicar todas as camadas presentes no modelo OSI, vale ressaltar que não serão extremamente detalhadas por não serem o foco principal do projeto mas focaremos nas camadas de Enlace, Transporte e Aplicação por estarem muito presentes na conexão SCADA-Religador.

A figura 1 auxilia na visualização e entendimento de todo o processo de transmisão de dados:

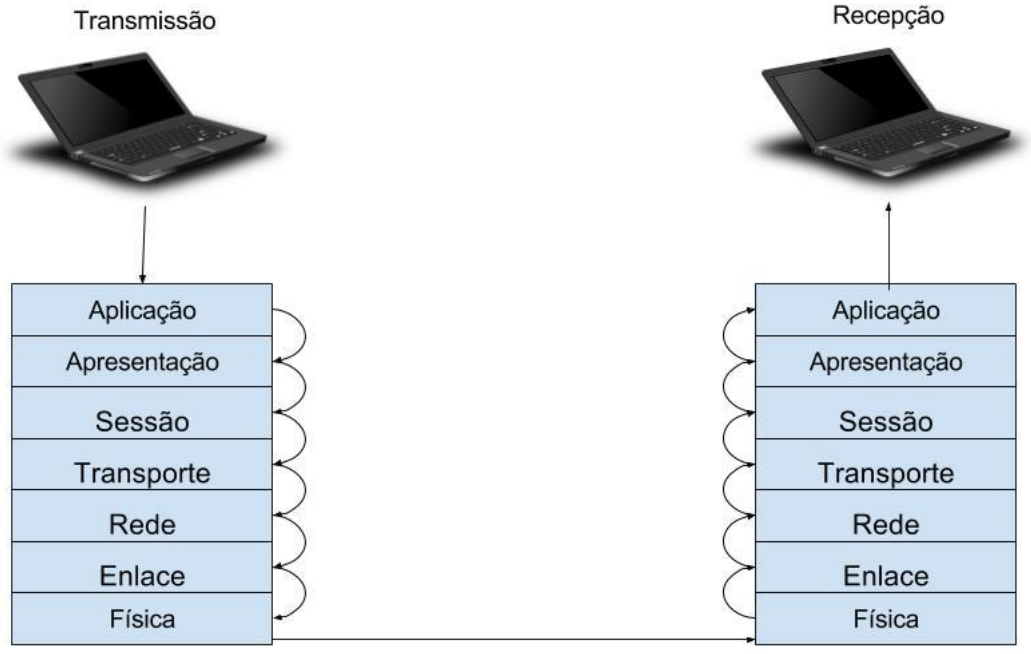

Figura 1: Caminho percorrido por um pacote de dados

1) Camada Física

Camada que define os sinais elétricos e conexões mecânicas no nível físico

Como o nome já indica, esta camada está traçada a enviar informações a nível físico, ou seja, através do fluxo bruto de bits de uma camada para outra. Isto pode ser feito através de meios guiados, como fibra ótica e fiação elétrica, ou por meios não guiados, sendo esses ondas de rádio e eletromagnéticas.

Podemos ressaltar algumas diferentes possibilidades para efetuar a interconexão entre a camada física do destinatário e da fonte da informação através de equipamentos eletrônicos. Serão esses, as pontes e os

roteadores.

Os roteadores são dispositivos que trabalham com a transmissão de pacotes do tipo ”armazena e reencaminha”. No entanto, eles requerem que o pacote originário passe por todas suas camadas, tendo em vista que ele somente possui 3, desde a Física, até a Camada de Rede. Com isso, podemos perceber que o roteador possui a capacidade de manter rotas de mensagem sem perda e de implementar algoritmos de roteamento, além da organização de mensagens da Camada de Enlace.

Outro equipamento que pode ser usado neste meio de transmissão físico é a ponte. Esta possui apenas a Camada Física e de Enlace e opera em quadros Ethernet organizando o cabeçalho (*header*) dos quadros e encaminhando diretamente para seu endereço de destino. Podemos descrever as pontes como sendo filtros de pacotes, exatamente pela organização descrita.

A imagem a seguir descreve a transmissão de um pacote de mensagem entre servidores passando por uma ponte e um roteador.

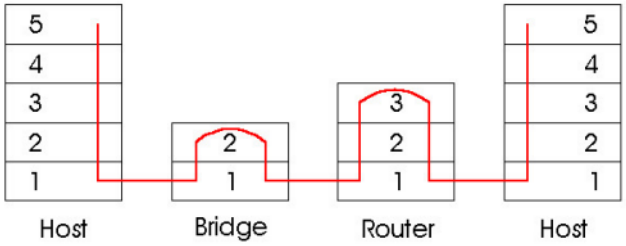

Figura 2: Aplicação de uma ponte e um roteador entre servidores

2) Camada de Enlace

Assim como a Camada de Transporte e a de Aplicação, a Camada de Enlace será mais aprofundada no seção de DNP3 deste relatório. No entanto, podemos ressaltar que está se trata, basicamente, de um ambiente em que se monta e envia quadro de dados de um sistema para outro oferecendo serviço para a Camada de Rede. Esta fornece um pacote de dados para o enlace que, por sua vez, adiciona informações de controle, gerando um novo pacote com (*header*), uma carga em forma de mensagem (*payload*) e o *trailer*, onde, mais comumente, se faz a checagem da informação. Este quadro contendo essas três vertentes é transmitido para a Camada Física para que seja enviada a mensagem.

3) Camada de Rede

Podemos descrever a cada de rede através de suas três principais funções:

- Endereçamento: Trata-se do direcionamento de fragmentos individuais de dados a um endereço específico que, por sua vez, é conhecido como IP.
- Encapsulamento e Desencapsulamento: Considerando o transmissor da mensagem, a Camada de Rede encapsula os segmentos em datagramas para que esses sejam organizados pela Camada de Enlace. No caso do receptor, ocorre o processo contrário em que ele examina o endereço de destino para certificar que a mensagem está sendo endereçada para o local correto. Caso contrário, a mensagem é perdida.
- Roteamento: Sendo este um processo intermediário entre o encapsulamento e desencapsulamento, o roteamento trabalha selecionando o caminho e direcionando os pacotes de dados aos seus devidos destinos.

4) Camada de Transporte

Camada responsável por gerenciar a comunicação entre os dois sistemas

Focando para o protocolo DNP3, esta camada tem a finalidade de segregar os fragmentos da originados Camada de Aplicação e formatá-los em quadros unitários (segmentos de transporte) para que sejam capazes de ser recebidos pela Camada de Enlace para que a mensagem completa possa ser compactada e enviada com maior precisão.

Define-se a mensagem principal da Camada de Transporte como sendo *Transport Header* (Cabeçalho de Transporte), nela estão presentes os campos FIN, FIR e Sequência que serão mais detalhados da seção de DNP3.

Agora, em meio mais genérico, vemos que a Camada de Transporte é utilizada para transmissão de dados independente da configuração de redes físicas entre duas máquinas distintas. Esta aplicação é fundamental pois possibilita a comunicação de diversas aplicações simuntâneas na rede em, até mesmo, um único dispositivo. Além disso, sabe-se que esta camada permite com que se use o mesmo canal para a transmissão de vários dados distintos de diferentes aplicações. Podemos usar como exemplo um computador que opera em diversos meios como email, páginas na web, serviço de streaming, dentre outros, ao mesmo tempo.

Nesses casos práticos do dia a dia, vemos importância da funcionalidade da Camada de Transporte. Em casos de transferência de arquivo, é possível observar a extrema necessidade que todos estes sejam enviados com as devidas informações organizadas. Já para casos menos significativos, como vídeos, a perda de pacotes não afeta a nossa visualização do serviço ao ponto de diminuir a qualidade do programa.

A figura abaixo resume o que foi descrito de forma resumida, porém, autoexplicativa:

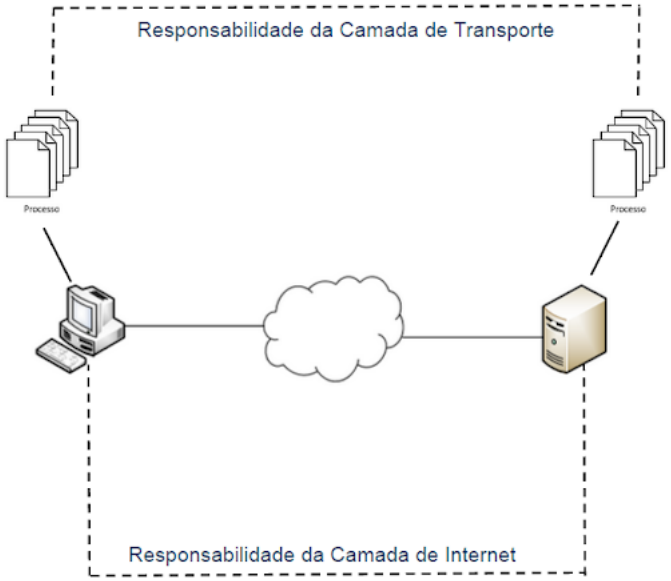

Figura 3: Demonstração básica da Camada de Transporte

5) Camada de Sessão

Camada responsável pelo controle das sessões entre os usuários

Tendo em vista a descrição da Camada de Transporte, a Camada de Sessão demonstra o objetivo de gerenciar este transporte de informações definido acima. Isso se define, principalmente, pela segurança da transmissão de dados, além de um maior sincronismo desses dados considerando uma conversa entre usuários, ou seja, um sistema fim a fim. Pode-se enaltecer os serviços relevantes desta camada da seguinte maneira:

- Intercâmbio de dados: consolidar a conexão entre dois usuários e estabelecer a conexão entre os mesmos;
- Sincronização: determinar certos pontos em uma interlocução para que estes sirvam como referência inibindo a perda de informações significativas;
- Relatório de erros: caso problemas sejam reconhecidos na comunicação entre usuários, os mesmos podem ser descritos aos operadores em questão.

6) Camada de Apresentação

É possível definir a Camada de Apresentação como a responsável pela conversão (através, também de criptografia) dos formatos de caracteres para que seja possível compreender as informações na Camada de Aplicação. Basicamente, trata-se de uma camada tradutora que aquisita dados codificados e os traduz para que nós, os usuários, possamos ler estas informações e utilizá-las da maneira que for mais viável.

7) Camada de Aplicação

De forma geral, podemos descrever a camada de aplicação como sendo a que fornece funções padronizadas, dados formatados e procedimentos para uma transmissão eficiênte de valores, atributos e comandos de controle para a aquisição de dados. Ou seja, esta é a camada que efetivamente está mais próxima do ser humano, onde ocorre a interação direta com a máquina.

Sendo esta a camada mais superficial dentre as sete do modelo OSI, podemos ter em vista diversas funcionalidades que ela nos proporcion, como a transferência de arquivos. Esta nos capacita carregar ou descarregar arquivos em equipamentos distinos. Isto é facilmente visualizado nos navegadores web atuais, considerando que, em máquinas distintas, é possível fazer o reconhecimento de permissões de acesso para que seja possível a troca de pacotes de mensagens.

Ainda nesta ideia de troca de mensagens, podemos ressaltar as mensagens de texto que, hoje em dia, podem ser enviadas e recebidas instantaneamente, essas estão inclusas, principalmente em e-mails, redes sociais e softwares direcionados para conversas particulares e grupais. No caso dos e-mails sabemos que é necessário que exista uma máquina para o endereço fonte e outra para o endereço de destino e que ambas se conectem para que a troca de mensagens seja efetuada. Caso não ocorra esta conexão, a mensagem é realocada em uma fila e a tentativa de envio será feita até que haja o recebimento da mensagem. No caso das redes socias e softwares de comunicação, o sistema é extremamente semelhante ao de emails, no entanto, possuem um processamento mais rápido resultando no imediatismo descrito.

Além disso, podemos citar a capacidade de acesso remoto como uma importante funcionalidade da camada de aplicação. Este processo requer um terminal virtual que, operando com teclado, mouse e tela presentes ao seu lado, é possível controlar máquinas que estão em outros ambientes a distâncias físicas indeterminadas, dependendo de diversos fatores programados para a seleção do terminal a ser utilizado. Para que este acesso remoto seja efetuado, foram desenvolvidos alguns serviços, assim podemos destacar o RPC(*Remote Procedure Call*) e RMI (*Remote Method Invocation*). Ambas tecnologias se destacam exatamente por permitirem que uma funcionalidade seja operada em uma máquina com endereçamento diferente da atual. No entanto, o modelo RMI apresenta um diferencial por poder baixar códigos de objetos mesmo se as classes destes não sejam definidas no equipamento do receptor. Ou seja, em outras palavras, a transmissão de dados atua de forma independente podendo atualizar este código de forma dinâmica.

Por fim, devemos nos atentar à segurança desta operação remota. Pode-se observar a presença dessa segurança em softwares atuais que nos pedem autentificações para que seja provado que nós, os principais operadores, somos nós mesmos. Este procedimento evita de forma considerada invasão alheia e exposição de dados particulares presentes em nossos computadores e celulares.

4 Protocolo DNP3

DNP3 (Protocolo de Rede Distribuída V.3) é um padrão de telecomunicações que define as comunicações entre estações mestres, unidades de telemetria remotas (RTUs) e outros dispositivos eletrônicos inteligentes (IEDs). Foi desenvolvido para alcançar interoperabilidade entre sistemas nas indústrias de utilidades elétricas, óleo e gás, água / águas residuais e segurança. O DNP3 foi inicialmente criado como um protocolo proprietário pela Harris Controls Division inicialmente para uso na indústria de utilidades elétricas. Em novembro de 1993, o protocolo foi disponibilizado para uso por terceiros, transferindo sua propriedade para o Grupo de Usuários DNP3. DNP3 foi projetado especificamente para SCADA (controle de supervisão e aquisição de dados). Isso envolve a aquisição de informações e o envio de comandos de controle entre dispositivos de computador fisicamente separados. Sendo assim, este padrão foi projetado para transmitir relativamente pequenos pacotes de dados de maneira confiável. A este respeito, é diferente de protocolos mais comumente utilizados e conhecidos, como FTP que faz parte do TCP / IP, que pode enviar arquivos muito grandes, mas de uma forma que geralmente não é adequada para controle SCADA.

O DNP3 é um padrão de protocolo aberto, amplamente implementado por fabricantes de equipamentos. Isso se mostra como uma grande vantagem, uma vez que ele fornece interoperabilidade entre equipamentos de diferentes fabricantes. Ou seja, um usuário pode comprar equipamento do sistema, como uma estação mestre de um fabricante, e adicionar um RTU proveniente de outro fabricante, de maneira simples e rápida. Assim estes equipamentos podem ser provenientes de diferentes fabricantes, tanto em uma instalação inicial, como progressiva conforme o sistema é desenvolvido ao longo do tempo.

É claro que os benefícios alcançáveis dependem de vários fatores. Alguns destes são mostrados abaixo. Percebendo os benefícios da interoperabilidade:

- Quantos fabricantes oferecem suporte ao protocolo?
- Eles estão aumentando ou diminuindo?
- Eles fornecem os produtos de que você precisa?
- Seus produtos são realmente interoperáveis?

Olhando para a lista de implementadores e usuários do DNP3 mostra que há um substancial nível de suporte para o protocolo em um número substancial de sistemas SCADA, RTUs, e muitos dispositivos eletrônicos inteligentes diferentes, como relés, instrumentos, conversores de protocolo e outros dispositivos. Uma lista de implementadores de DNP3 está disponível no Usuário DNP3 Site do grupo na Internet (em http://www.dnp.org/).

DNP3 foi reconhecido como tendo um sistema de conformidade particularmente forte. No além de ter uma especificação abrangente de objetos de dados, DNP3 tem uma detalhada sistema de certificação de conformidade. Isso se baseia na definição de subconjuntos de implementação para os quais os dispositivos devem ser certificados. Isso fornece um meio para os fabricantes implementarem sistemas de função reduzida que ainda fornecem níveis definidos de funcionalidade.

Para obter interoperabilidade entre as versões de diferentes fabricantes do DNP3, foi necessário garantir que todas as implementações suportem os dados e funções, para que fosse mais viável a fabricação para comercialização de dispositivos capazes de se comunicarem pelo DNP3, foi criada uma definição de níveis de subconjuntos. Existem atualmente três níveis de subconjuntos definidos no DNP3. Estes são designados no formato DNP3-L1, L2 ou L3.

Os subconjuntos são definidos listando os objetos de dados específicos que devem ser tratados, e as operações a serem realizadas neles, para cada nível. Esses requisitos são definidos em uma tabela mostrando para cada tipo de objeto de dados os códigos de função e valores de campo do qualificador que deve ser suportado pelo mestre e escravo nesse nível. A tabela é chamada de tabela de implementação. Assim, cada dispositivo deve ter um documento de perfil, que especifica questões de nível de aplicação e nível de enlace de dados.

Level 1	Level 1 is the simplest level of DNP implementation. It is intended for use between a master station or intermediate device such as a data concentrator, and a small IED end device. This might be a relay, a meter, or a stand-alone controller of some type. Normally any inputs and outputs are local to the device. Examples of small end devices include meters, relays, auto-reclosers or capacitor bank controllers.
Level 2	Level 2 defines a larger subset of DNP features than Level 1. It is intended to be used between a master station or data concentrator, and an RTU or a large IED. Normally any inputs and outputs are local to the device.
Level 3	Level 3 defines the largest subset of DNP features. It does not require support of all the possible DNP features, but it does cover the majority of the most frequently required features. This is implemented typically between a master station and a larger or more advanced RTU. The inputs and outputs of Subset Level 3 devices would typically be remote as well as local.

Figura 4: Tabela de implementação [1]

Na área de comunicações SCADA e IED, havia a necessidade de uma simplificação do modelo OSI que omitisse algumas das funções de nível mais alto, sendo assim, foi criado um modelo pela Comissão Eletrotécnica Internacional (IEC) de 3 camadas, conhecido como modelo de arquitetura de desempenho aprimorado (EPA). As camadas usadas neste modelo são as duas camadas de hardware e a camada superior de software, a camada de aplicativo. É este modelo no qual DNP3 é baseado. Assim como no modelo OSI, cada camada do modelo transmite as informações passadas da camada superior adicionando informações relacionadas aos serviços executados pela camada em questão. A informação adicional é geralmente anexada como um cabeçalho, na frente da mensagem original. Assim, durante a composição da mensagem, ela cresce em tamanho com cada camada que passa para baixo. Uma vez que ela atravessa a camada física e chega em seu destino a mensagem é desmontada em unidades menores de dados e passa pelo processo inverso, onde as partes adicionadas a mensagem são removidas camada a camada e ela é organizada da mesma forma em que ela se originou. Para um melhor entendimento do protocolo em si, uma mensagem DNP3 será examinada ao longo deste capítulo.

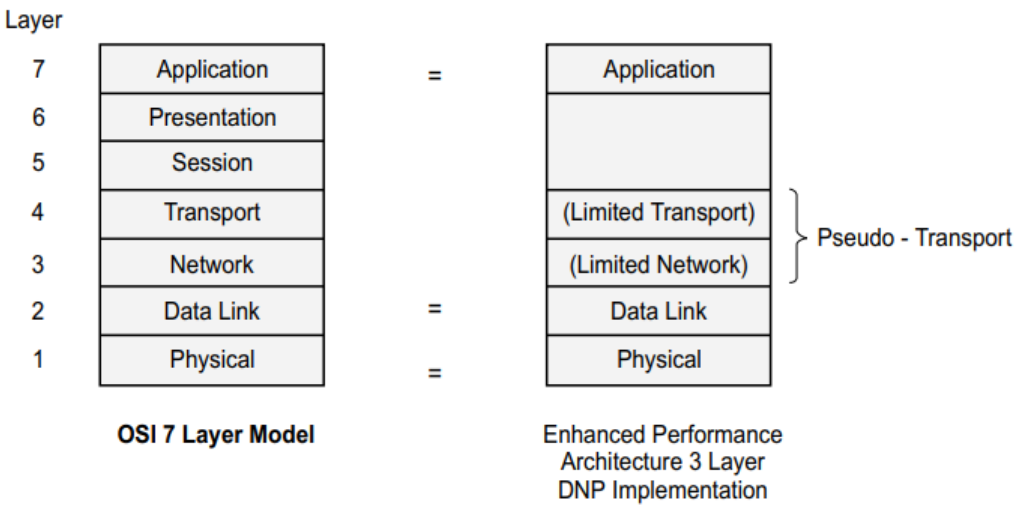

Figura 5: Equivalência entre OSI-DNP3 [1]

a Camada de aplicação

A camada de aplicação é o nível mais alto, onde os dados são gerados para envio. É a camada que repassa as informações referente a mensagem para os níveis inferiores para alcançar o resultado final da transmissão das informações necessárias. Assim, a camada de aplicação DNP3 fornece seus serviços para os programas de aplicativos do usuário, como um sistema HMI, um RTU ou outro sistema.

Os dados do usuário são os dados provenientes da aplicação que faz a interface com o usuário. O aplicativo do usuário pode ser visualizado como uma camada acima da camada do aplicativo, e pode, por

exemplo, ser um programa de interface humano-máquina (HMI). Um ponto chave é que os dados a serem repassados pela mensagem podem ser de qualquer tamanho, uma vez que o tamanho total dos dados não é limitado pelo protocolo. A camada de aplicação inicialmente forma os dados em blocos de tamanho mais propício a serem conduzidos. Estes blocos são chamados de ASDUs. A camada de aplicativo, então, cria os APDU da unidade de dados de protocolo de aplicativo combinando um cabeçalho com os dados ASDU. O cabeçalho do aplicativo é essencialmente a informação de controle do protocolo do aplicativo ou APCI. Isso tem tamanho de 2 bytes se a mensagem é uma solicitação ou 4 bytes se a mensagem é uma resposta. No caso de um comando ou outra solicitação do usuário, em que não há necessidade de dados adicionais, a mensagem se resume a um cabeçalho sem nenhum ASDU. Dependendo do tamanho total dos dados a serem transmitidos (o tamanho do ASDU), um ou mais APDUs são criados. No caso de serem necessários vários APDUs, estes são denominados fragmentos. Embora o número de fragmentos necessários para representar um ASDU não seja limitado, o tamanho de cada fragmento é limitado a 2.048 bytes. É importante ressaltar que o protocolo DNP3 definiu objetos de dados formando uma biblioteca de objetos, em que cada objeto de dados tem uma estrutura definida, diferentes tipos de dados podem ser usados para representar uma fonte de informação e cada definição de objeto de dados pode ter múltiplas variações.

No nível do aplicativo, existem dois tipos básicos de mensagens no DNP3. Estes são solicitações e respostas. Apenas uma estação mestre pode enviar uma solicitação, e apenas uma estação escrava pode enviar uma resposta. No entanto, existe uma classe especial de resposta chamada ”resposta não solicitada”(CON=0).

Informações de controle (AC) em cada mensagem que passa entre a estação mestre e a estação externa. As informações de AC são [Número FIR FIN CON SEQ] onde FIR é o primeiro fragmento da mensagem FIN indica o fragmento final da mensagem e CON é a confirmação(1 se necessária, 0 se não solicitada). Por último vem SEQ, a sequência do fragmento, em que bits entre 0-15 são solicitações de estação mestre e respostas de estação externa (solicitadas), entre 16-31 são respostas não solicitadas.

b Camada de pseudo-transporte

Função: Esta camada permite a transmissão de blocos maiores de dados do que poderia ser feito de outra forma. As funções de transporte fornecem entrega de ponta a ponta transparente à rede de mensagens inteiras, incluindo desmontagem e remontagem, assim como correção de erros detectados.

A camada de transporte leva os dados do usuário, a unidade de dados do serviço de transporte e os quebra em uma ou mais unidades de dados seguindo o protocolo de transporte, e envia cada parte para os dados da camada de enlace. Estas menságens menores(chamadas TPDUs) tornam-se os LSDUs na camada de enlace de dados. Lembrando que as LSDUs podem ter no máximo 250 bytes de dados do usuário, isso define o tamanho máximo do LPDU, dentro da camada de transporte é adicionado um único byte de cabeçalho, deixando 249 bytes para o transporte de dados.

Em uma estação secundária, os TPDUs de entrada são remontados como TSDU e repassados para a camada de aplicação. Os bytes do cabeçalho de transporte são removidos e o TSDU é reformado a partir de vários TPDMs. A camada de transporte é responsável por garantir que o TSDU é remontado na sequência correta.

c Camada de enlace de dados

A camada de enlace de dados tem como principal objetivo fornecer transmissão confiável de dados(agrupados) através do meio físico, provendo também controle de fluxo e detecção de erro .

O estabelecimento do enlace se baseia em realizar a configuração lógica por trás da comunicação entre o remetente e o receptor da mensagem. A unidade de dados no nível da camada de enlace é chamada de quadro, ele tem um tamanho máximo de 292 bytes, incluindo códigos CRC, e carrega um total de 250 bytes de informação dos níveis superiores. O quadro inclui fonte de 16 bits e endereços de destino em seu cabeçalho. O intervalo de endereços entre 0xFFF0 e 0xFFFF é reservado para mensagens de transmissão processadas por todos os receptores. É possível que um dispositivo físico tenha mais de um endereço lógico, nestes casos, os endereços aparecem como dispositivos separados para a estação mestre. O cabeçalho do quadro também contém um código de função. As funções suportadas por este são aquelas necessárias para inicializar e testar a operação de cada enlace lógico estabelecido entre o remetente e o receptor. Como um recurso de segurança adicional, cada quadro transmitido pode solicitar uma confirmação de recebimento(confirmação na camada de enlace).

No DNP3 apenas uma estação mestre pode emitir um pedido, e apenas um escravo pode fornecer uma resposta. No caso de uma estação externa (uma estação escrava) tendo dados não solicitados para enviar ao mestre estação, ele emite uma resposta não solicitada ao mestre. No nível de aplicação, esse é um tipo diferente de mensagem para uma solicitação. No nível do link de dados, no entanto, o quadro da mensagem não parece diferente de qualquer outro quadro de mensagem, seja emitido por um mestre ou um escravo. Como no DNP3 as comunicações são balanceadas, os termos definidos como primário e secundário se relacionam com a estação que inicia uma transação no nível de enlace de dados e a estação/estações que recebem (não existe correlação se as estações estão mestre ou escravo), assim qualquer estação pode ser uma estação primária. No DNP3, as estações também são definidas como mestres ou escravos, Essas informações são usadas no nível do link para determinar a configuração de um bit de direção da mensagem, o "DIR" bit. O bit de direção é definido para mensagens de um mestre e limpo para mensagens de uma estação escrava.(DIR bit)

Uma definição final que é importante entender é o 'link de dados' ou apenas 'link'. A ligação refere-se à conexão lógica entre uma estação primária e uma estação secundária. Isto é portanto, um caminho de comunicação unilateral. Para estabelecer comunicações bidirecionais entre dois dispositivos, é necessário estabelecer os links em cada direção.

O DNP3 controla a transmissão no nível da camada de enlace usando procedimentos de transmissão definidos. Esses procedimentos possuem um byte de controle contido no "quadro" da mensagem para controlar a transmissão. Os procedimentos definem quais ações são realizadas em cada final, já o byte de controle fornece a coordenação entre os quadros, definindo o tipo de transmissão que está sendo realizada(o tipo de quadro e onde o quadro se encaixa no processo).

O formato de quadro foi baseado no FT3 especificado pela IEC 870-5-1. O formato especifica um cabeçalho de 10 bytes, seguido opcionalmente por até 16 blocos de dados. O tamanho geral da mensagem é limitado a 292 bytes, o que fornece uma capacidade máxima de dados de 250 bytes. Assim, um quadro totalmente empacotado compreenderá o cabeçalho mais 16 blocos de dados, com o último bloco contendo 10 bytes de dados.

Formato de quadro FT3:

Inicio:2 bytes: 0564 (hex)

Tamanho:Contagem de dados do usuário em bytes, mais 5, sem contar os bytes CRC(tamanho padrão: 0–255).

Byte de controle de quadro: Tem em seu início o bit de controle mencionado acima(DIR bit), em seguida, o bit primário/secundário, que define se o quadro está iniciando ou respondendo a mensagem, segue então os bits de contagem de quadros que são usados em mensagens primárias para detectar perdas ou duplicação de frames para uma estação secundária e em mensagens secundárias para cada transação "SENDCONFIRM" bem-sucedida entre as mesmas estações primária e secundária. O próximo bit é o bit de controle de fluxo de dados. Em seguida ficam os códigos de função de enlace de dados, os quais devem ser interpretados diferentemente caso seja uma transmissão primária ou secundária. Por último existe o controle de erros(CRC), ele é responsável por detectar quando um erro ocorreu em uma mensagem com um código de redundância cíclica de 16 bits.

Endereço de destino: 2 bytes (LSB, MSB)

Endereço de origem: 2 bytes (LSB, MSB)

CRC: Código de verificação de redundância cíclica de 2 bytes

Dados:Cada bloco possui 16 bytes de dados do usuário. O último bloco tem 1–16 conforme necessário. Em caso de um quadro completo, o último bloco terá 10 bytes de dados do usuário.

Exemplo do livro:

Primary to secondary (A to B)

<05 64 05 C0 01 00 0C 00 CF A0>

Link header	0564	
Length	05	
Control byte	C0	; DIR=1, PRI=1, FCV=0, FC=RESET
Destination address	0001	
Source address	000C	
CRC value	CFA0	

Secondary to primary (B to A)

<05 64 05 00 0C 00 01 00 FD E9>

Link header	0564	
Length	05	
Control byte	00	; DIR=0, PRI=0, FC=ACK
Destination address	000C	
Source address	0001	
CRC value	FDE9	

Figura 6: Menságem primária e resposta [1]

d A camada física

A camada física define as características da interface física em termos de especificações elétricas, tempo, pinagem e assim por diante. Isso inclui os detalhes necessários para estabelecer e manter o link físico. O elemento de dados neste nível é essencialmente o bit, ou seja, preocupa-se em como passar um bit de dados por vez. A camada física também inclui as funções de controle da mídia, como detalhes necessários para estabelecer e manter o link físico e para controlar o fluxo de dados.

Esta camada converte cada quadro FT3 em um fluxo de bits por um meio físico, em que uma camada física assíncrona bit-serial é usada com dados de 8 bits, 1 bit de partida, 1 bit de parada, sem paridade, com os níveis de tensão e sinais de controle seguindo o padrão RS-232C, já o protocolo de hardware é o CCIT V.24 para comunicações DTE / DCE. O Comitê Técnico do Grupo de Usuários DNP3, produziu um padrão para transmissão de DNP3 nas redes, fornecendo uma outra opção para essa situação.

O DNP3 oferece suporte a comunicações de Multidrop de um Mestre, ponto a ponto e múltiplos mestres. Ele suporta os modos operacionais em poll e quiescente. Operação quiescente é assim chamada porque sondagens para verificar se há alterações não são necessárias. Isso ocorre porque a estação mestre pode contar com a estação externa para enviar uma ”resposta não solicitada”quando houver uma alteração que precise ser relatada. Assim, na ausência de mudança, o sistema permanece quiescente ou em um estado silencioso, com nem pesquisas da estação mestre, nem respostas das estações externas. Este modo de operação permite um melhor uso da capacidade do sistema de comunicações. Em um sistema quiescente, geralmente uma pesquisa periódica de fundo ainda é usada, talvez em intervalos de hora em hora, para proteção contra falhas de comunicação não detectadas. Se isso não foi feito, a estação mestre não teria como detectar a falha de comunicação com a estação externa, caso ocorra. Seria apenas supor que nada mudou. Visualização do DNP3 71 A capacidade de suportar operação peer-peer e quiescente requer que as estações que não são designadas como estações mestras podem iniciar comunicações. Isso às vezes referido como comunicações ’balanceadas’, o que significa que qualquer estação pode atuar como uma estação primária (ou emissora) e como uma estação secundária (respondente) ao mesmo tempo. Apesar da capacidade das estações não mestre de iniciar comunicações dentro do DNP3, apenas estações mestres podem iniciar solicitações de dados ou emitir comandos para outras estações. Assim, embora o termo balanceado seja aplicado ao sistema de comunicações, a diferenciação entre estações mestras e escravas continua necessária. Às vezes, os termos mestre e outstation são usados para refletir mais apropriadamente as capacidades do sistema. As arquiteturas também podem envolver o uso de conversores de protocolo para fazer interface com um ou mais dispositivos usando um protocolo de comunicação diferente. Um conversor de protocolo pode ser usado em caso de uma topologia hierárquica, onde os dispositivos outstation usam apenas DNP3, e o mestre SCADA pode usar um sistema de comunicação diferente. No caso de dispositivos DNP3 com uma porta de rede, DNP3 é encapsulado em TCP / IP Pacotes Ethernet. Embora isso adicione a sobrecarga associada a esses pacotes, fornece um meio eficaz de usar redes locais ou de área ampla para atender SCADA comunicações. Em alguns casos, isso pode permitir a extensão eficiente de um sistema SCADA fazendo uso de uma rede corporativa existente.

e O mundo pós Smart-Grid

Os sistemas usados para aplicações SCADA frequentemente envolvem o uso de linhas PSTN e modems analógicos/links de rádio. As taxas de dados envolvidas estão normalmente na faixa de 2.400 a 9600 baud. Quando DNP3 é usado em uma situação de curta distância, uma configuração multiponto é realizada usando o RS-485, ele é limitado a 32 dispositivos. Atualmente existem tendências mais amplas que estão levando as comunicações em uma direção diferente. Onde os dispositivos finais são inteligentes, ou seja, eles têm processamento on-board, e podem ser conectados ao resto do sistema por meio de um concentrador de dados ou dispositivo de controle, como um RTU. Esses dispositivos inteligentes mudaram a natureza dos requisitos de comunicação. Onde em sistemas mais antigos, o sistema SCADA recebe informações de um RTU, este conectado aos dispositivos finais, atualmente é necessário um sistema um pouco mais complexo. No outro extremo da escala, os requisitos para comunicações também se tornaram mais extensos, onde antes as plantas operacionais individuais eram pouco conectadas com dentro de uma rede, isso não acontece mais. A rede elétrica de hoje mudou de forma que as empresas possam operar em áreas geográficas maiores, e os sistemas de comunicação mudaram paralelamente a isso. Uma das principais consequências disso é a adoção generalizada de redes locais e de longa distância (LANs e WANs) para fornecer conectividade de dados entre organizações/empresas. Essas tecnologias estão sendo cada vez mais utilizadas na indústria de utilidades elétricas, como em muitas outras.

Assim, coincidiu com o crescimento dos protocolos SCADA abertos de DNP3 e IEC60870-101, houve uma revolução na conectividade em toda a organização, e esta é com base no uso de LANs e WANs. Dentro de uma subestação ou planta operacional, uma LAN pode ser usada para fornecer comunicações confiáveis de alta velocidade entre os equipamentos locais também como por meio de roteadores para LANs próximas ou remotas, ou para uma WAN de nível empresarial. Assim surgiu a pressão para transportar DNP3 em uma rede WAN. O Comitê Técnico do Grupo de Usuários DNP3 definiu um método para transportar DNP3, envolvendo o uso do pacote de protocolos da Internet para as camadas de transporte e rede, e a camada física Ethernet.

O pacote de protocolo da Internet é o grupo de programas que fornecem os serviços do Internet. Estes incluem protocolo de Internet (IP), protocolo de controle de transmissão (TCP) e UDP.

O protocolo da Internet fornece um meio de envio de pacotes de dados, chamados datagramas, os endereços de origem e destino são transportados no cabeçalho do datagrama como 32 bits. O IP não oferece garantia de entrega, pois não há confirmações e a verificação de erros é limitada ao cabeçalho. A responsabilidade por estas funções são deixadas para níveis superiores. O IP também fornece fragmentação de dados em menores pacotes conforme exigido pela rede pela qual o datagrama está passando e para a remontagem dos fragmentos no datagrama original na extremidade de destino.Cada datagrama é independente, enviado por comunicação única. Os datagramas do protocolo Internet têm um cabeçalho de 20–24 bytes seguido por seus dados. Elas podem ter, até 65 K bytes, mas tem um tamanho máximo recomendado de 576 bytes.

O protocolo de controle de transmissão (TCP) fornece um meio confiável para a conexão de dados de um usuário para outro. Ele é projetado para fazer uso de IP para transmitir datagramas e assim fornecer uma transmissão livre de erros. O TCP estabelece links de comunicação entre 'sockets' em cada extremidade. Seguindo a inicialização do link, as comunicações podem ocorrer em ambas as direções entre cada fim. O TCP divide os dados em segmentos e os passa para o IP, que por sua vez pode fragmentá-los em datagramas menores. O TCP tem um cabeçalho de segmento de 20–24 bytes. O tamanho do segmento é determinado pelo menor 'Tamanho máximo do segmento' configurado em cada extremidade do link, ou em 536 bytes para um endereço de IP.

O protocolo de datagrama do usuário (UDP) se destina a fornecer um meio de acessar o datagrama/serviço de IP diretamente da camada de transporte. Um datagrama UDP traduz para um datagrama IP sendo enviado, com as informações adicionais de origem e destino portas, além de uma soma de verificação. O UDP não fornece um recurso de entrega confiável como o TCP, porém ele pode usar o endereçamento de transmissão de IP para enviar para vários destinos simultaneamente.

A ideia de transportar DNP3 em um ambiente de rede envolve o encapsulamento dos dados quadros da camada de enlace de dados DNP3 dentro dos quadros da camada de transporte do pacote de protocolo da Internet e permitindo que a pilha de protocolo entregue os quadros da camada de enlace de dados DNP3 para o local de destino no lugar da camada física DNP3 original.

- DNP3 deve usar o conjunto de protocolos da Internet para transportar mensagens de LAN / WAN;
- O link físico mais recomendado é Ethernet, sem inibir a utilização de outros;
- Todos os dispositivos devem ser capazes de operar TCP e UDP;

- TCP deve ser usado para WANs;
- TCP é altamente recomendado para LANs;
- UDP é necessário se as mensagens de transmissão não puderem ser perdidas(sejam de suma importância);
- A pilha de protocolo DNP3 deve ser mantida por completo;
- Na camada de enlace, as confirmações devem ser desabilitadas.

5 Protocolo Modbus Serial

a O Protocolo

Agora tratando do Modbus, temos que este protocolo foi desenvolvido para a comunicação de computadores presentes em uma mesma rede. Independentemente do tipo de rede, a ideia deste protocolo é definir uma estrutura de mensagem que possui bytes para que a maior quantidade de dispositivos possível possa se comunicar. Nesta seção do relatório, trataremos mais a fundo da comunicação por modo RTU pois foi a utilizada durante o processo de comunicação do SCADA com o Arduino. Mais a frente o mesmo será detalhado.

Sobre o meio de comunicação utilizando o protocolo Modbus, já a nível da mensagem, temos que este se utiliza do princípio mestre-escravo, mesmo caso a rede seja própria para *peer-to-peer* (arranjo de uma estrutura de rede onde cada computador opera tanto como cliente, quanto como servidor). Analisando o método mestre-escravo, este funciona da mesma maneira ao descrito ao longo do projeto, em que o mestre requisita um comando e o escravo trabalha para fornecer a operação pedida.

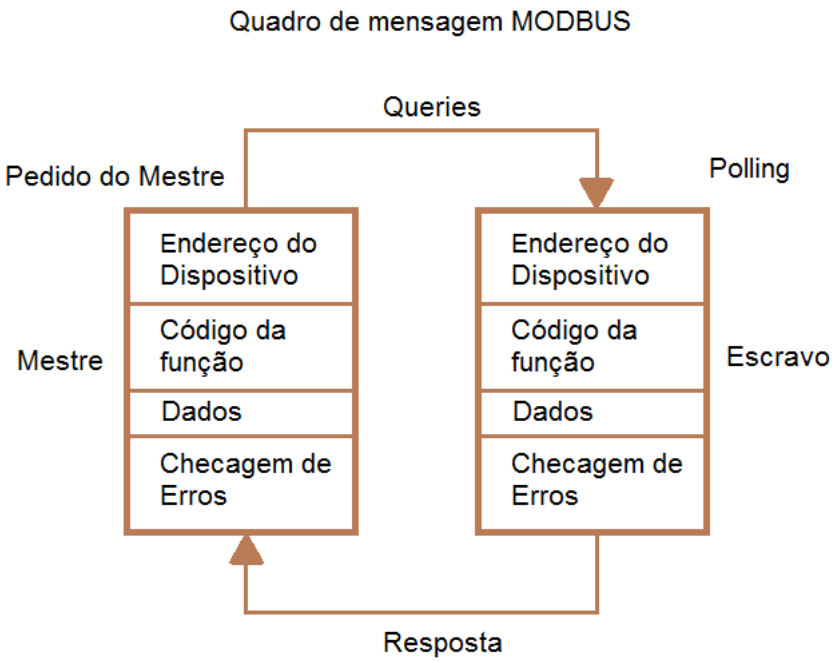

Figura 7: Processo de envio e recebimento de mensagem com Modbus

b Comunicação Modbus

Nos aprofundando no meio de comunicação Modbus, como podemos observar na imagem acima, percebe-se que a mensagem é iniciada pelo mestre através de um requerimento (queries) e esta será respondida pelo escravo. O envio de informação pode ser feito, tanto individualmente, quanto por um enlace de mensagens em forma de cadeia, em que podemos denominar como *broadcast*. No entanto, é importante ressaltar que a resposta do escravo só será efetuada caso ocorra um endereçamento específico para este em questão.

Já pelo lado do escravo, a mensagem precisa ser correspondente aos moldes estabelecidos pelo mestre, e isso definirá a função a ser realizada pela mensagem, os dados presententes na informação que serão relevantes na resposta, além do campo de checksum (ciclo de checagens do Modbus) para a checagem da informação enviada. No caso da resposta do escravo ao mestre não ser viável, considerando algum problema no meio do processo, uma mensagem será endereçada ao mestre contendo uma justificativa a não correspondência.

A seguir serão destacados os modos de transmissão de mensagens utilizados no protocolo Modbus. Os dois disponíveis são: ASCII e RTU, tendo em vista que nosso foco se dará para o RTU pois foi o meio

utilizado na conexão SCADA-Arduino detalhada mais a frente no relatório. Estes dois modos se diferem, principalmente, no empacotamento da informação.

O ASCII, ao ser selecionado, sua mensagem se configura no envio de dois caracteres no padrão ASCII para cada palavra de dados. Este modo permite que existema grandes intervalos entre o envio dos dados de uma mesma mensagem.

Já para o RTU, diferentemente do ASCII, sabe-se que para cada palavra contida na informação, apenas um caracter é enviado na forma de um hexadecimal. Com isso em mente, é possível perceber que este modo possui uma maior densidade de caracteres presentes em um pacote de mensagens, aumentando a eficiência da comunicação. Este fator foi fundamental para a escolha deste modo em nosso processo de conexão com o servidor SCADA exatamente para termos uma comunicação ininterrupta, imediatista e muito eficiênte.

Considerando os parâmetros de comunicação, devemos ressalatar que a quantidade de bits por cada palavra da mensagem será igual a 11. O arranjo se dá por 1 (um) bit de início da mensagem, 8 (oito) bits de dados e:

- Caso não exista paridade, 2 (dois) bits de fim da informação;
- 1 (um) bit para paridade par e 1 (um) para fim da informação;
- 1 (um) bit para paridade impar e 1 (um) para fim da informação.

Após este processo, temos a checagem da informação através do checksum que, por sua vez, opera de maneira similar ao CRC presente na seção DNP3.

c Pacote de Mensagem

Trazendo a tona o empacotamento da mensagem no protocolo Modbus, podemos ressaltar os indicadores de início e fim da mensagem. Estes possuem a função de determinar o dispositivo a qual está sendo enviada a informação e ler o conteúdo presente neste pacote. Isso se dá através da detecção da mensagem e leitura do campo de endereço. A diferenciação entre os modos RTU e ASCII se dá exatamente pela distinção entre seus indicadores.

Nos aprofundando no empacotamento no modo RTU, podemos ressaltar que indica seus campos de início e fim por meio do não envio de mensagem por um certo período de tempo, sendo este ,aproximadamente, 3,5 vezes o período utilizado para o envio de uma palavra de dados. Neste meio tempo, podem ser enviadas quaisquer mensagens com hexadecimais sem a interrupção da informação.

Como exemplo de cálculo para início e fim da mensagem, podemos considerar a utilizada em nosso projeto de 115200 bps. Tendo em vista que uma palavra de dados de 11 bits pode ser representada por $1/115200$, com isso teremos $(1/115200) * 11$ para o envio completo e, assim, o tempo total será de 95,4us. Ou seja, para fecharmos a transmissão e iniciarmos uma nova, o invervalo entre mensagens deve ser de 334,2ms.

Considerando o envio de dados do Arduino, percebe-se que o, por ser contínuo e muito eficiênte, o sistema SCADA sempre irá reconhecer a comunicação sem perda de pacotes. No entanto, caso o intervalo seja 1,5 vezes maior que o período necessário para a transmissão de uma palavra de dados, o escravo irá descartar as informações recebidas da mensagem atuale se utilizará do próximo caracter para o endereçamento de uma nova mensagem. Abaixo podemos analisar a estrutura de um pacote de dados em Modbus RTU:

Início do Pacote	Endereço do Escravo	Função Modbus	Dados para o Escravo	Checksum		Fim do Pacote
T Início	1 char	1 char	N char	CRC-	CRC+	Tfim

Figura 8: Estrutura do pacote Modbus no modo RTU
[2]

Uma diferenciação relevante do caso do RTU em relação ao ASCII se dá nos campos de Endereço do Escravo e Função Modbus, onde ambos possuem 1 *byte* ao invés de 2. Além disso, no Checksum, o início deste se dá pelo byte menos significativo e este campo é finalizado com o byte mais significativo.

Ainda tratando sobre os dois campos citados acima, é válido ressaltar para o Endereço do Escravo que os bits representando o endereço do escravo que receberá o requerimento do mestre estarão presentes no escopo da informação. Além disso, ao responder, o mesmo endereço é incitado na mensagem para o

mestre para que este saiba qual escravo está enviando o pacote. Já para o campo Função Modbus, como o nome já indica, a mensagem presente neste *byte* indicará a ação a ser feita pelo escravo. Na resposta, este campo será utilizado para descrever se o requerimento foi bem ou mal sucedido sabendo-se que, caso tenha sido um sucesso, a mensagem conterá o mesmo código enviado e, caso contrário, o bit mais significativo enviado será enviado pelo escravo em nível 1 (um).

d Status de Transmissão de Informação

A seguir serão destacados alguns status que são apresentados em uma comunicação mestre-escravo no protocolo Modbus:

- Função inválida

  Ocorre quando o escravo reconhece a mensagem do mestre, porém diferentemente do caso acima, ele identifica que a função ModBus solicitada não está implementada no dispositivo, assim ele envia uma mensagem de exceção. Com isso, o mestre finaliza o processo de recepção.

- Time-out

  Ocorre quando o terminal escravo não envia uma resposta até …. Então o mestre executa novas tentativas(retries sequences)(intervalo mínimo de 100 ms), transmitindo novamente a mensagem enviada, isso ocorre até o escravo responder a mensagem enviada pelo mestre ou o número de tentativas programadas pelo mestre chegue ao limite, em ambos os casos, o mestre finaliza o processo de recepção.

- Registrador inválido

  Ocorre quando o escravo reconhece a mensagem, porém identifica que os registradores na mensagem não existem e o escravo envia uma mensagem de exceção. Com isso, o mestre finaliza o processo de recepção.

- Valor de dado inválido

  Se apresenta quando o escravo recebe a mensagem, mas identifica que o valor de algum dos dados da mensagem no seu campo de dados é inválido, assim o escravo envia uma mensagem de exceção. Com isso, o mestre finaliza o processo de recepção.

- Estado de espera

  Se passa quando o escravo recebe a mensagem, mas envia uma resposta para o mestre, em que o notifica que é preciso de mais tempo para processar a mensagem(para evitar o timeout). Com isso o mestre finaliza o processo de recepção, e dependendo da aplicação, pode permanecer monitorando as atividades do escravo.

- Dispositivo ocupado

  Ocorre quando o escravo recebe a mensagem, porém, envia uma resposta de exceção com um aviso que não pode executar a função solicitada, por estar atendendo outro comando. Assim o mestre finaliza o processo de recepção, e dependendo da aplicação, pode enviar novamente a mensagem após um período de tempo.

6 Desenvolvimento do projeto SCADA-Religador

a Conexão DNP3 Simulator - Test Harness

Após introduzir de forma generalizada o projeto de conexão do servidor SCADA com um software que simula um religador, é extremamente relevante demonstrar o desenvolvimento e passos tomados para que possa ser visualizado todo o processo da conexão. Primeiramente, iniciamos o software Test Harness que nos demonstra a seguinte apresentação:

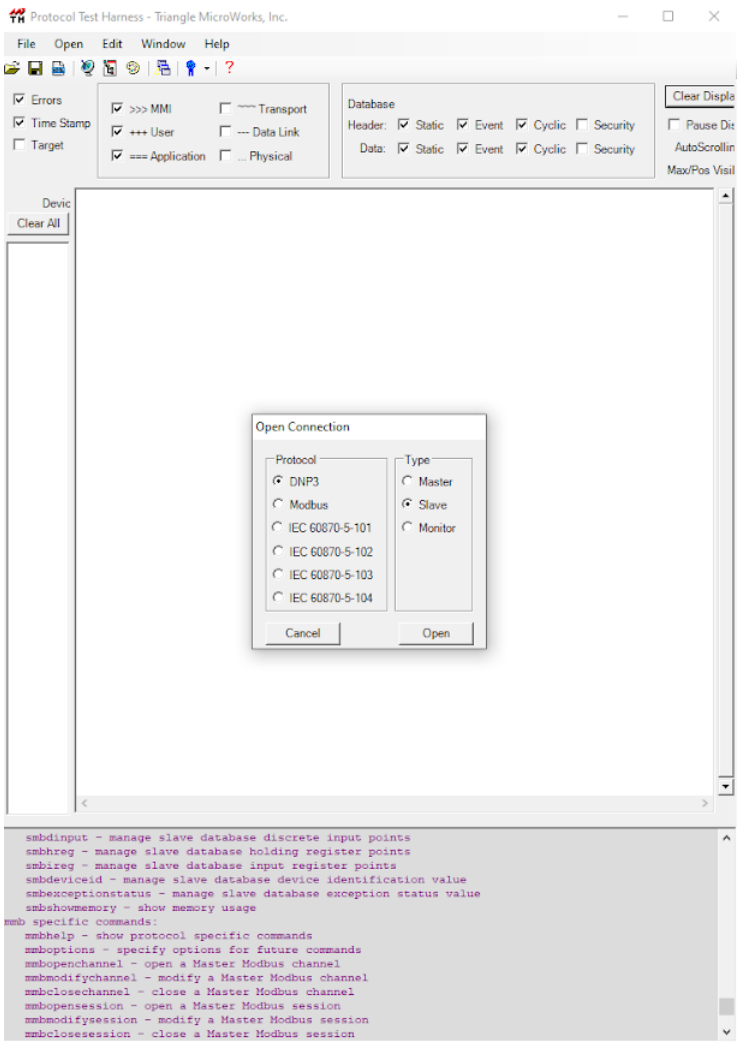

Figura 9: Página inicial do Test Harness

Ao requerir o protocolo de comunicação e o tipo de conexão que seria determinado, selecionou-se DNP3 e Slave, respectivamente, conforme a ideia inicial do projeto. Após pressionar "Open", o software efetivamente criou a conexão.

Com isso, foi possível operar as páginas demonstradas abaixo:

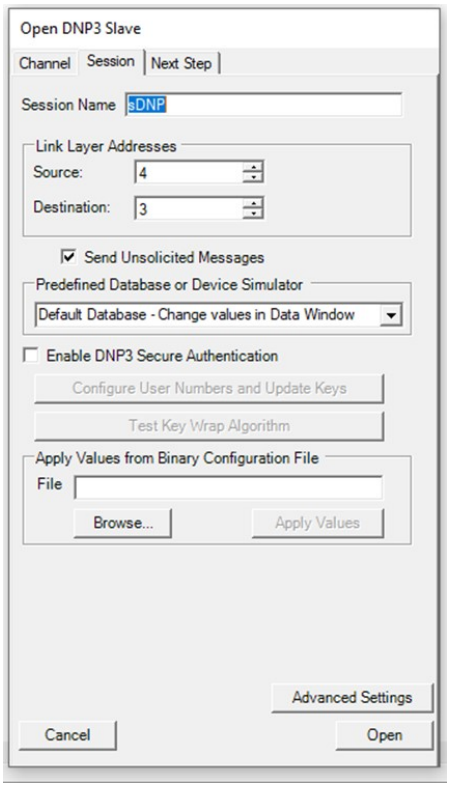

Figura 10: Configuração DNP3-TH 1

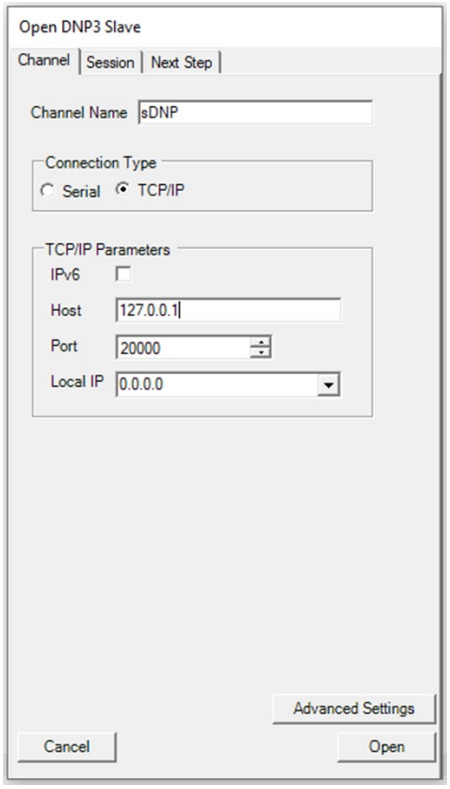

Figura 11: Configuração DNP3-TH 2

Esta etapa do projeto é fundamental, pois nela é possível determinar a fonte e o destinatário das informações oriundas do Test Harness, além do IP em que a operação ocorrerá. Com isso, conforme mostram as imagens, selecionou-se, para a camada de transmissão de dados (data link layer), a fonte e destinatário 4 e 3, respectivamente, apenas por já ser padrão do software. Além disso, também por já ser originário do Test Harness, determinamos o host como sendo o local através do IP 127.0.0.1. Além disso, selecionou-se o tipo de conexão como sendo por IP e a visualização do procedimento na janela padrão do software (ainda não simulando um religador, mas sim um slave). Por fim, clicando em "Open" o software começa a operar buscando a quem se conectar, ou seja, algum elemento que faça alguma solicitação à ele.

Para se ter maior detalhamento do projeto, escolhemos utilizar, antes do servidor SCADA, um simulador DNP3 para que fosse possível visualizar com maior clareza e instantaneidade a transmissão de informações e a efetividade da conexão entre ambos os simuladores. Para isso, utilizou-se o DNP3 Simulator.

Seu método de conexão é extremamente simples. Ao iniciar, apresenta a seguinte imagem:

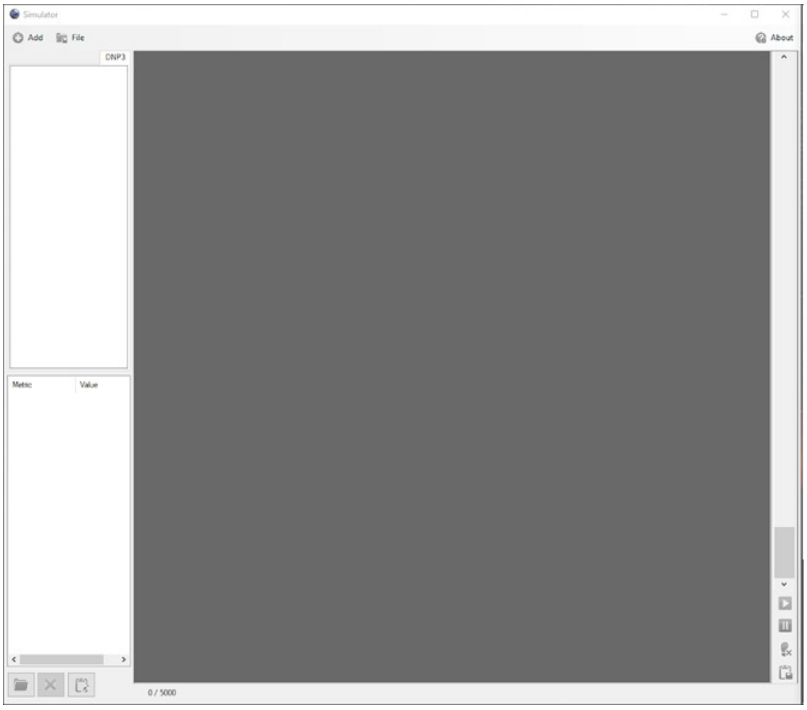

Figura 12: Página inicial do DNP3 Simulator

Ao clicar em "Add" vamos configurar como será tratado o nosso simulador, se irá operar como mestre ou slave ou, como descrito no próprio software, "Server" ou "Client". Por operar como mestre, vamos adicionar uma simulação do servidor que utiliza o protocolo DNP3 para comunicação. A imagem abaixo demonstra que, assim como anteriormente, determinou-se o IP da operação como sendo 127.0.0.1, coincidindo com o Test Harness.

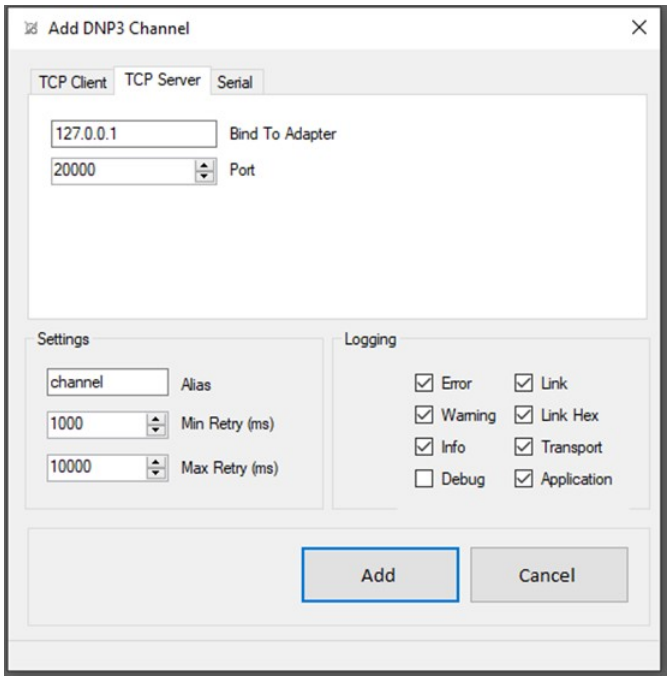

Figura 13: Escolha do canal no DNP3 Simulator

Clicando em "Add", seguimos para página que, de forma similar à descrita anteriormente, a informação da camada de dados será transmitida ao endereço 3, ou seja, o DNP3 simulator será a fonte 3. De

forma análoga, o destinatário das informações será enviado para o endereço 4. Ou seja, o Test Harness receberá informações pelo endereço 4 e enviará dados ao 3. Vale ressaltar que os softwares reconhecem este sistema de endereçamento pois estão trabalhando no mesmo IP, sendo este o local. A imagem abaixo apresenta o que foi narrado.

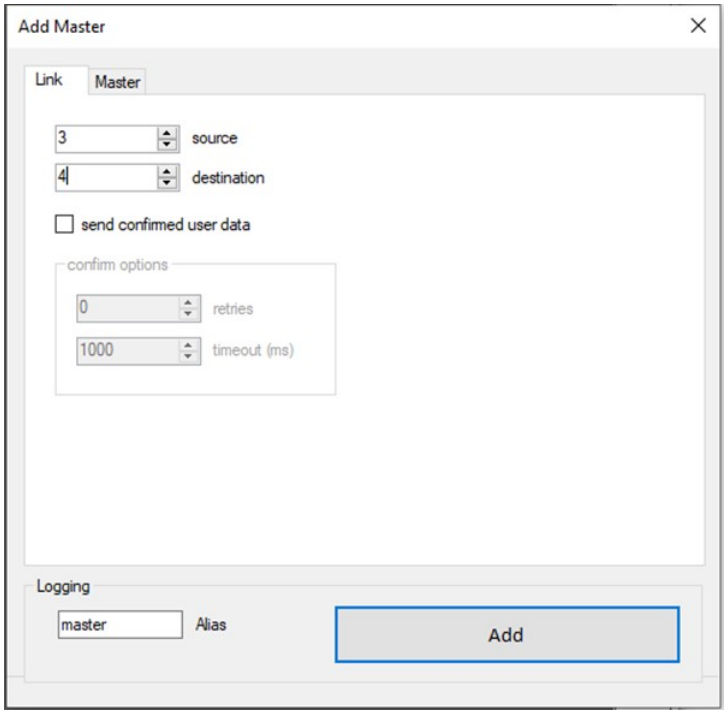

Figura 14: Escolha da fonte e destinatário no DNP3 Simulator

Com isso feito, clicando em "Add" e com o Test Harness operando da maneira como foi descrito, finalizamos a conexão com ambos os softwares atuando em conjunto.

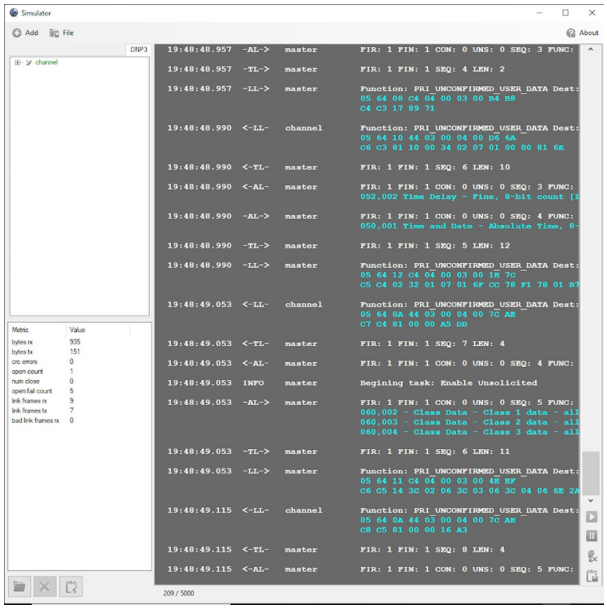

Figura 15: DNP3 Simulator conectado

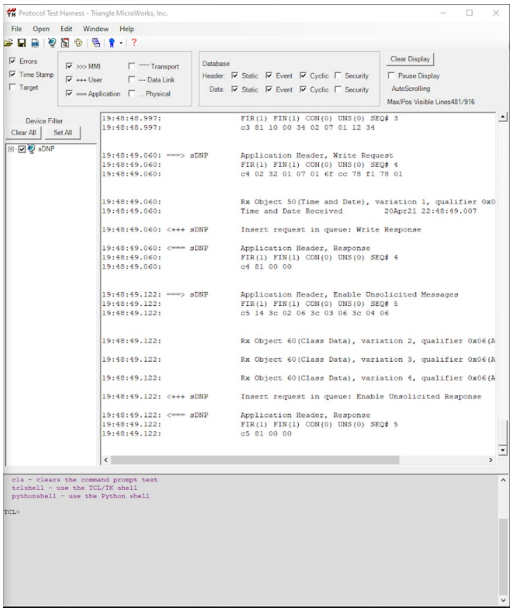

Figura 16: Test Harness conectado

Vale ressaltar que não foi possível operar o Test Harness. Isso se dá pois o DNP3 Simulator não opera como servidor SCADA, que requisita e envia dados. No entanto, como podemos observar na imagem

acima, é vasta a quantidade de informações presentes apenas devido à conexão de ambas as partes concretizada.

Mesmo sem operar, ressalva-se que os softwares enviam informações ininterruptamente entre si, sempre checando se um se comunica com o outro. Vemos a seguir, respectivamente, duas situações distintas, a primeira em que o mestre (simulador DNP3) é desconectado e a segunda em que o slave (Test Harness) é desconectado:

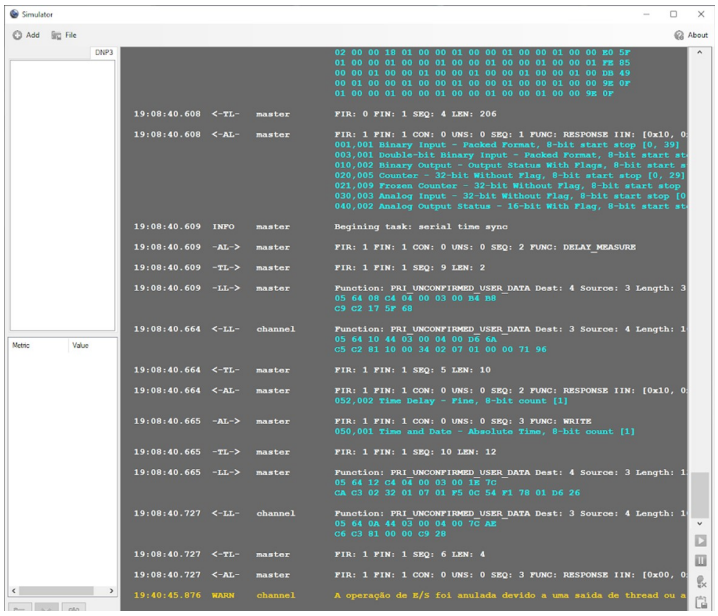

Figura 17: DNP3 Simulator desconectado

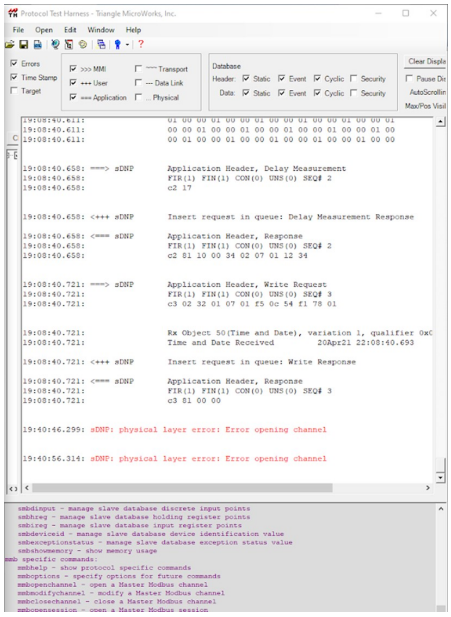

Figura 18: Test Harness operando

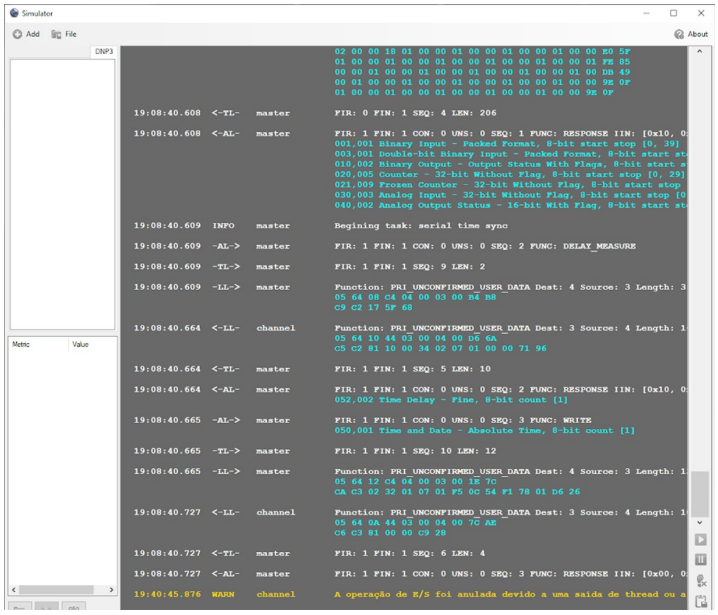

Figura 19: DNP3 Simulator operando

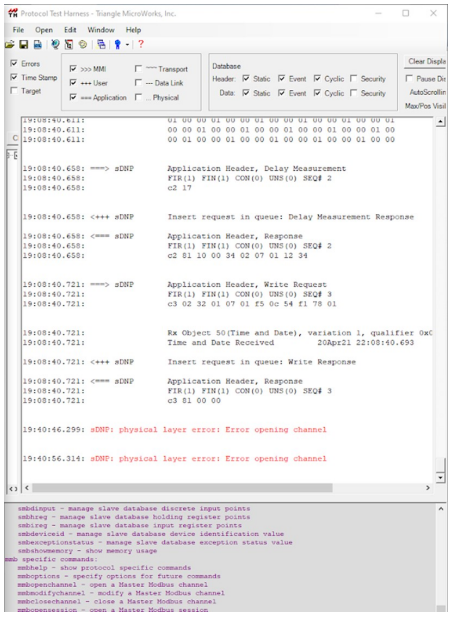

Figura 20: Test Harness desconectado

Em ambos os casos, o status de falha é apresentado pelo software que continua operando, demonstrando que a conexão foi um sucesso.

b Conexão SPA1 - Test Harness

Após obtermos o conhecimento e experiência para conectar um servidor que se utiliza do protocolo DNP3 à um meio definido como slave, já podíamos progredir e utilizar um servidor SCADA e, além disso, configurar o software Test Harness para, efetivamente, operar como um simulador de religador.

O software SPA1 é um operador SCADA. Nele vamos iniciar a configuração para podermos recolher informações relevantes do religador e enviar requisitos para o mesmo, estes vão desde ligar e desligar o religador até bloquear o funcionamento deste aparelho. Ambos vão operar no mesmo computador apenas por méritos de simualção.

Iniciamos a configuração através da figura 21 que demonstra a página inicial do SPA1.

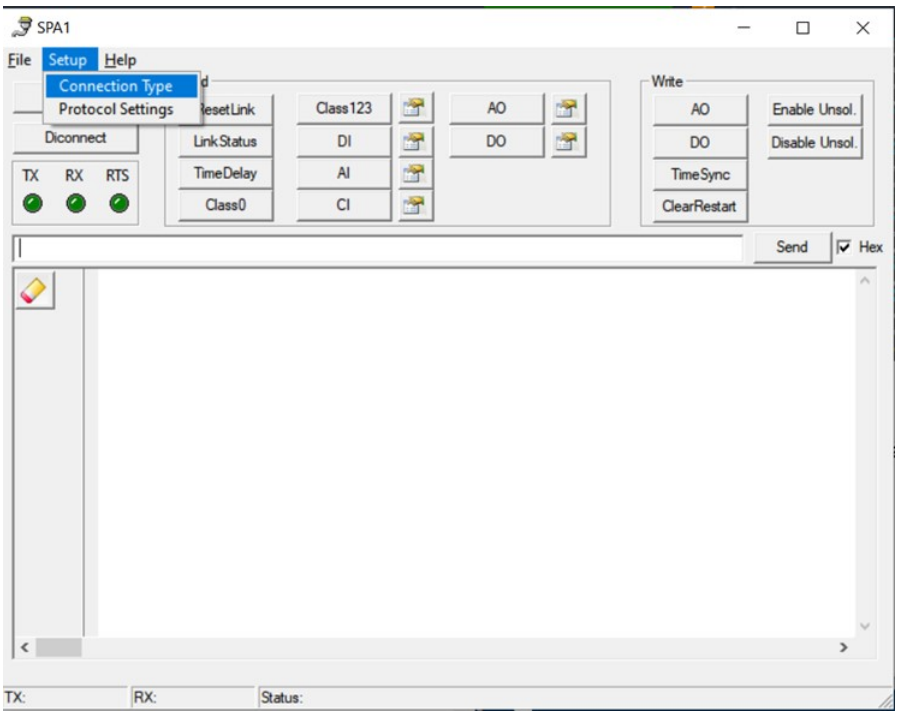

Figura 21: Inicializando a configuração do servidor SCADA

Clicando em ”Connection Type”, teremos a opção de selecionar o IP da operação da conexão. De maneira similar à utilizada no DNP3 Simulator, vamos nos comunicar no IP padrão do pré selecionado, 127.0.0.1, assim como a porta já padrão, 20000. Por fim, deve-se clicar em ”OK”para salvar as alterações.

Além disso, necessitamos configurar as informações para a conexão protocolar entre os dois softwares e, assim como anteriormente, como mostra a figura 25, a fonte para envio de dados será de endereço 3 e o destinatário o 4.

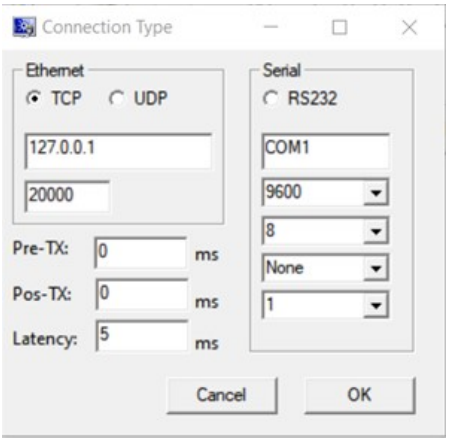

Figura 22: Página de configuração do tipo de conexão do SPA1

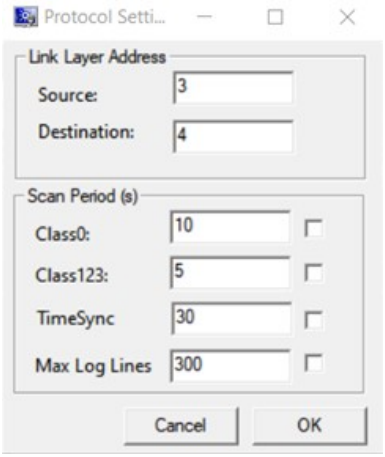

Figura 23: Página de configuração do meio do protocolo de comunicação do SPA1

Com estas duas etapas já configuradas, basta fazer o passo a passo do Test Harness da mesma maneira do caso anterior, apenas alterando, na etapa de seleção do ”Predefined Database or Deviece Simulator”, o operador padrão, pelo simulador de religador ABB. Também poderíamos selecionar o religador da marca SEL, mas preferimos optar pelo da ABB por nos dar uma gama de opções e operações maior que o outro.

Por fim, clicando em ”Connect”(abaixo de ”Setup”na página inicial, como mostra a figura 21), a operação é iniciada imediatamente e de forma ininterrupta até que um dos softwares seja desconectado. As imagens abaixo demonstram ambos em funcionamento:

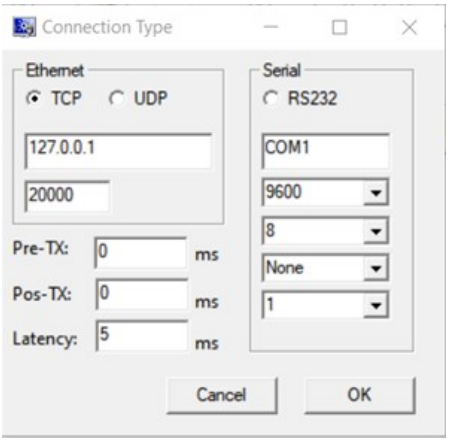

Figura 24: Página de configuração do tipo de conexão do SPA1

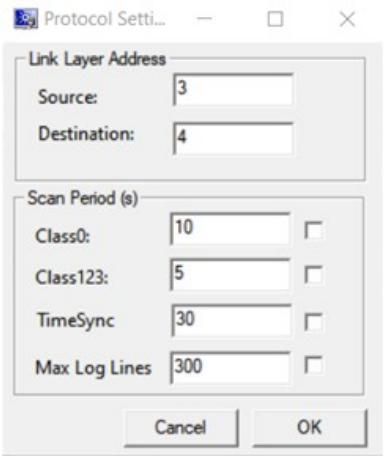

Figura 25: Página de configuração do meio do protocolo de comunicação do SPA1

7 Desenvolvimento do projeto SCADA-Arduino

Conforme descrito, foi visto como necessário desenvolver uma nova vertente no projeto do Trabalho de Conclusão de Curso para se ter uma maior base ao tratarmos de um sistema SCADA atuando em uma rede elétrica inteligente. Analisamos as possibilidades para simular equipamentos que fossem viáveis financeiramente e pudessem ser controlados através de um sistema SCADA.

a Determinação dos componentes

1 Arduino

Consideramos o Arduino como sendo o dispositivo a ser utilizado pois já o tínhamos a disposição e o conhecimento prévio sobre este micro controlador podería nos auxiliar consideravelmente ao longo dos testes, como ocorrido.

Figura 26: Micro controlador utiliziado

Para o modelo do Arduino selecionamos o Uno pois, após pesquisas, vimos que o mesmo se encaixava perfeitamente em nossas condições. Este equipamento funcionará como o escravo da conexão. Isso se dá devido ao fato do SCADA requisitar dados ou enviar pedidos e o Arduino fornecê-los para supervisão e análise.

Com o micro controlador selecionado, foi necessário determinar os dispositivos que seriam conectados ao mesmo para que pudéssemos monitorar seus comportamentos no sistema SCADA. Sendo assim, acoplamos os seguintes equipamentos ao Arduino Uno:

- Potenciômetro: Considerando que seria possível observar em tempo real a variação de seus dados analógicos na interface SCADA conforme variamos fisicamente no próprio dispositivo;
- LED: Este foi utilizado visto que, a princípio, além de monitorar, o nosso sistema também poderia enviar dados e comandos, o que se concretizou por sermos capazes de desligar e ligar o LED através do SCADA;
- Sensor de temperatura e umidade: Por fim, inserímos este dispositivo para ver a compatibilidade de comunicação e monitoramento deste com o sistema SCADA.

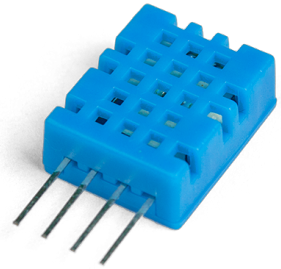

Figura 27: Sensor de umidade utilizado

Na codificação do Arduino utilizou-se a biblioteca DHT, que inclui as funcionalidades deste sensor e, além dela, a SimpleModbusSlave que, por sua vez, possui os requisitos para enviar os dados através do protocolo de comunicação Modbus.

2 SCADABR

Após concluirmos esta etapa do Arduino, analisou-se os possíveis meios de utilização do SCADA. Com isso, definimos que o SCADABR seria o software com a melhor relação viabilidade-funcionalidade dentre os que buscamos.

O SCADABR é um software livre, gratuito e de código-fonte aberto, para desenvolvimento de aplicações de Automação, Aquisição de Dados e Controle Supervisório. Este pode ser tratado como um servidor que possui a capacidade de monitorar equipamentos e controlar parâmetros de acordo com a demanda do cliente. O fato do SCADABR ser *open source* que o tornou viável ao nosso projeto, principalmente tendo em vista que maioria destes softwares ser consideravelmente custosa e por termos a facilidade de acessá-lo em qualquer navegador de internet em computadores ou celulares.

Podemos citar algumas aplicações fundamentais do SCADABR:

- Sistemas de Energia (Geração, Distribuição e Transmissão);
- Eficiência Energética (incluindo Multimedidores e Medição Setorial)
- Aplicações para Redes de Sensores sem-fio;
- Automação Industrial/ Controle de Processos;

Ainda tratando deste software, seria necessário, assim como no projeto SCADA-Religador, utilizarmos um protocolo de comunicação comum a ambos os programas e que nos fosse viável operar. Com isso, viu-se que uma das possibilidades era a do protocolo Modbus que, por ser extremamente difundido mundialmente em indústrias e comércios, consideramos que fosse a melhor opção de comunicação entre o SCADABR e o Arduino. Mais a frente, ao tratarmos da execução da conexão, será indicado como foi aplicado o Modbus entre os dois softwares. Além deste protocolo, o SCADABR também é capaz de aquisitar dados em diversos outros protocolos, como: OPC, IEC, ASCII, HTTP, dentre outros.

b Execução da conexão

1 Arduino

Iniciamos a conexão escrevendo o código (A) para efetivamente enviar os dados requisitados ao SCADABR. Após configurar os parâmetros iniciais, como a porta serial, taxa de transmissão, dentre outros, foi necessário, no *loop* do Arduino, incitar os comandos para que enviem as informações continuamente. Com isso, tivemos as seguintes linhas demonstradas na figura 28:

```
void loop()
{
  modbus_update();
  holdingRegs[POT] = analogRead(A1);
  digitalWrite(8, holdingRegs[LED]);
  holdingRegs[HUM] = dht.readHumidity();
  holdingRegs[TEMP]= (dht.readTemperature());
}
```

Figura 28: *Loop* do código do Arduino

As linhas indicam os seguintes casos, respectivamente:

- Modbus_update(): Sempre enviar as informações relacionadas ao protocolo Modbus;
- holdingRegs[POT] = analogRead(A1): Representa a leitura do potenciômetro na porta analógica A1 do Arduino;
- digitalWrite(8, holdingRegs[LED]): Representa a interação do LED com a porta digital 8 do Arduino;
- holdingRegs[HUM] = dht.readHumidity(): Executa a leitura da umidade através do sensor;

- holdingRegs[TEMP]= (dht.readTemperature()): Executa a leitura da temperatura através do sensor;

Os nomes POT, LED, HUM e TEMP, são nomes pré definidos por nós para facilitar o entendimento do código.

Abaixo podemos observar o circuito fisicamente completo:

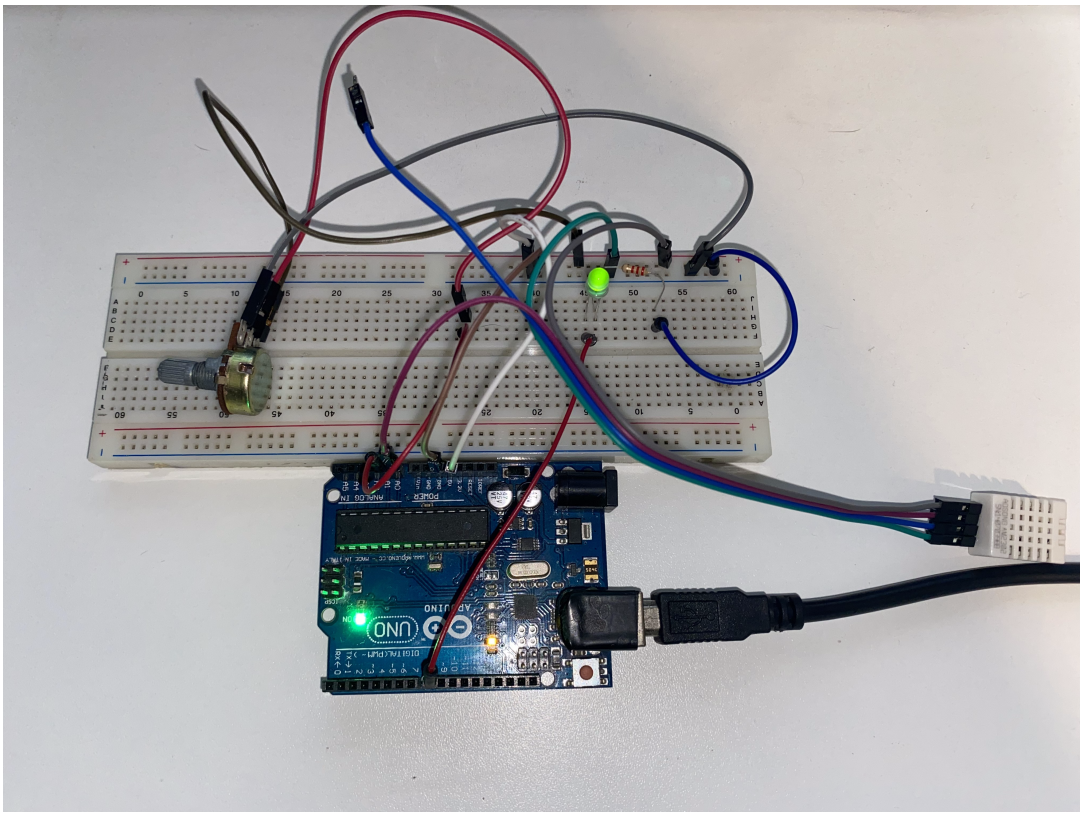

Figura 29: Circuito com Arduino Uno desenvolvido

2 SCADABR

Com o envio de informações do equipamento tratado como escravo já definido, foi possível operar o SCADABR para podermos monitorar e ajustar os parâmetros do circuito do Arduino Uno.

Após a instalação do software SCADABR, nos deparamos com opção de entrarmos na página de *data sources* onde será possível configurar os dados de entrada a serem monitorados pelo servidor.

Figura 30: Tela de *data sources* do SCADABR

Conforme pode-se observar na imagem 30, temos uma vasta lista de protocolos de comunicação que o SCADABR nos oferece. Tendo isto, considerando a viabilidade do projeto, determinamos que o Modbus Serial seria o mais apropriado. O fato de termos selecionado a comunicação serial se dá devido ao fato de estarmos utilizando um Arduino Uno com diversas portas seriais, abrangendo ainda mais as nossas capacidades de conexões.

Após selecionarmos o "Modbus Serial", já podemos adicionar a conexão. Com isso, teremos a seguinte página:

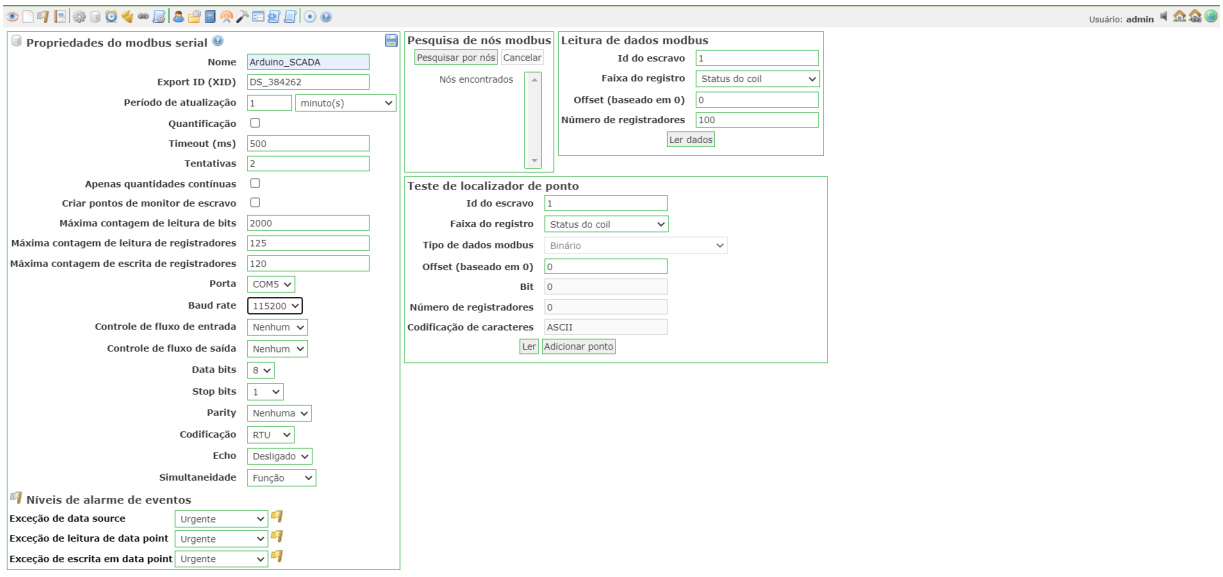

Figura 31: Página de definição de parâmetros da conexão Modbus Serial

Nesta imagem podemos ressaltar dois parâmetros alterados para que a conexão fosse efetivada:

- Porta: COM5.

  Este item é importante considerando o momento de atuação da IDE (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado) do Arduino.

O processo se dá pelo comunicação direta de atuação do envio de informações do Arduino para o software SCADA. Durante esse processo, a IDE do Arduino é completamente desativada para que a porta COM5 possa operar esta troca de mensagens e comandos entre os softwares.

- Baud rate (taxa de transmissão): 115200

  Este parâmetro trata da velocidade na qual servidores conseguem transmitir dados. Basicamente, é o número de eventos que ocorrem em um segundo. Coincidindo o Baud rate do SCADABR e do Arduino, concluímos que teremos uma comunicação apropriada.

Após a definição destes dados, clicamos "Pesquisar por nós"para que o SCADABR procure o local da operação em que serão recebidas as informações. Posteriormente ao encontro do nó, podemos salvar o procedimento e seguir para a descrição detalhada dos pontos de dados. A imagem 32 demonstra este detalhamento.

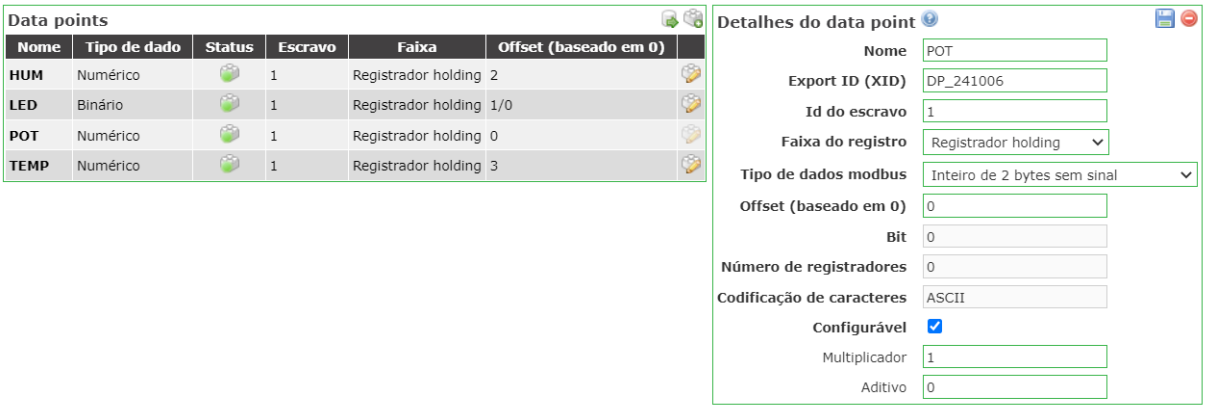

Figura 32: Detalhamento dos pontos de dados

No exemplo acima estamos desenvolvendo o dado do potenciômetro. E vamos ressaltar os seguintes fatores:

- Faixa do registro: Registrador holding.

  Este parâmetro, fundamental no protocolo Modbus, define a capacidade que temos para ler e escrever os comandos que serão enviados entre o usuário mestre e o escravo.

- Tipo de dados modbus: Inteiro de 2 bytes sem sinal.

  Para este caso, conforme definimos no próprio Arduino, o valor enviado do potenciômetro seria um inteiro de 2 bytes. Para os outros três parâmetros, os definimos de acordo com a codificação do IDE do Arduino.

Com todos os parâmetros detalhados e definidos, foi possível estabelecer e finalizar a conexão.

Clicando na *watch list* (lista de reprodução dos dados), podemos observar nossos equipamentos em operação. A imagem abaixo demonstra os quatro parâmetros em funcionamento:

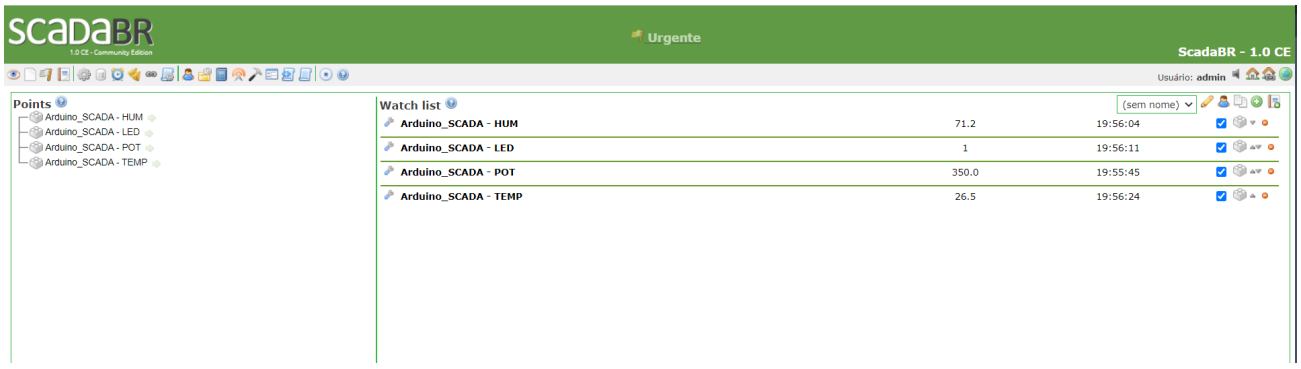

Figura 33: Reprodução e operação dos parâmetros

Podemos observar que o projeto foi concluído com sucesso pois é possível operar e ler os parâmetros descritos anteriormente com valores que são perfeitamente coinciêdentes e plausíveis com o que estava presente no momento. Ressaltando-se o valor do potenciômetro, que, por sua vez, varia de 0 a 1023, tendo em vista que é um valor analógico. Além disso, trazemos a tona o LED, que podemos acender e apagar diretamente do software SCADABR.

8 Conclusão

Após o desenvolvimento aprofundado do projeto incluso no Trabalho de Conclusão de Curso, foi possível concluir que temos diversas maneiras diferentes de operar um sistema SCADA, intensificando a viabilidade de termos, de forma mais difundida ao redor do mundo, redes elétricas inteligentes.

Neste trabalho operamos um software simulador de religador mas poderíamos tratar de outros como chaves automatizadas, reguladores de tensão, transformadores de corrente e potência, bancos de capacitores, medidores de energia, sensores de falta e meteorológicos. Tal fato demonstra o descrito anteriormente, tratando-se da vasta capacidade de aplicações de um sistema SCADA para tornar as redes de transmissão elétricas cada vez mais eficiêntes e viáveis financeiramente tanto para as concessionárias, quanto para a população.

Além da parte prática, tratamos também da operação interna de comunicação entre servidor mestre e escravo. Isto fez com que tivéssemos uma maior familiaridade com os protocolos de comunicação Modbus e DNP3, tornando possível entender melhor o comportamento tanto do servidor SCADA, quanto do escravo, sendo ele o religador ou o próprio Arduino.

Vale ressaltar que utilizamos o Arduino para simular diferentes equipamentos de uma rede elétrica inteligente. Além de apenas monitorar, foi possível enviar comandos para o LED, representando alguma operação de ligar e desligar algum regulador de tensão, por exemplo. Além disso, temos o potenciômetro que pode simular certas operações em medidores de energia que poderiam ser visualizadas no servidor SCADA.

Referências

[1] M. M. P.-P. Wijesekera, *DNPSec: Distributed Network Protocol Version 3 (DNP3) Security Framework.* Springer, 2007.

[2] (2000) Protocolo de comunicação modbus rtu/ascii. [Online]. Available: https://www.dca.ufrn.br/~affonso/FTP/DCA447/modbus/modbus_manual.pdf

[3] E. W. B. Gordon Clarke CP Eng, Deon Reynders Pr.Eng, *Practical Modern SCADA Protocols: DNP3, 60870.5 and Related Systems*. Newnes, 2004.

[4] BlackBrix. (2018) Simple-modbus-slave. [Online]. Available: https://github.com/BlackBrix/Simple-Modbus-Slave

[5] adafruit. (2018) Dht-sensor-library. [Online]. Available: https://github.com/adafruit/DHT-sensor-library

[6] (2012) Ieee standard for electric power systems communications-distributed network protocol (dnp3). [Online]. Available: https://ieeexplore.ieee.org/document/6327578

[7] (2001) Ieee recommended practice for data communications between remote terminal units and intelligent electronic devices in a substation. [Online]. Available: https://ieeexplore.ieee.org/document/912952

[8] D. U. Group. (2002) DNP3 SPECIFICATION. [Online]. Available: https://cdn.chipkin.com/assets/uploads/imports/resources/DNP3Introduction-Draft_G.pdf

[9] DNP 3.0 Protocol Database Implementation. [Online]. Available: https://npeinc.com/manuals/Adobe%20E-Manuals/AC%20Stuff%20from%20voyten/Reclosers/DNP%203%20v324%20rev34.pdf

[10] Kepware. (2020) DNP3 and Control Relay Output Block Command. [Online]. Available: https://www.kepware.com/getattachment/ae44d711-0ccb-4cf3-b1e5-6a914bd9b25e/DNP3-Control-Relay-Output-Block-Command.pdf

[11] I. A. S. MODICON, Inc. (1996) Modicon Modbus Protocol Reference Guide. [Online]. Available: https://www.modbus.org/docs/PI_MBUS_300.pdf

[12] MultiTrode. (1996) PROTOCOL Translator DNP3 User Manual. [Online]. Available: https://www.xylem.com/siteassets/brand/flygt/flygt-resources/flygt-resources/protocol-translator-dnp3-user-manual.pdf

[13] S. Toolbox. TOP Server 5 DNP Advanced Operations Control Relay Output Block Commands. [Online]. Available: https://support.softwaretoolbox.com/ci/fattach/get/104276/1473251427/redirect/1/session/L2F2LzEvdGltZS8xNjE5MzgxNTQ2L3NpZC9hYkFiZW85cA==/filename/DNP3_AdvancedOperation_CROBCommands.pdf

[1] [3] [2] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13]

A Apêndice

Segue abaixo o código do Arduino desenvolvido:

```
#include < SimpleModbusSlave.h >
#include < DHT.h >; //INCLUSODEBIBLIOTECA
#include < DHT_U.h >
#defineDHTPINA0//PINODIGITALUTILIZADOPELODHT22
#defineDHTTYPEDHT22//DEFINEOMODELODOSENSOR(DHT22/AM2302)
DHTdht(DHTPIN, DHTTYPE); //PASSAOSPARMETROSPARAAFUNO
enum
{
//justaddorremoveregistersandyourgoodtogo...
//Thefirstregisterstartsataddress0
POT,
LED,
HUM,
TEMP,
HOLDING_REGS_SIZE//leavethisone
//totalnumberofregistersforfunction3and16sharethesameregisterarray
//i.e.thesameaddressspace
};
unsignedintholdingRegs[HOLDING_REGS_SIZE]; //function3and16registerarray
//////////////////////////////////////////////////////////////
voidsetup()
{
//sensor.begin();
modbus_configure(&Serial, 115200, SERIAL_8N1, 1, 2, HOLDING_REGS_SIZE, holdingRegs);
//modbus_update_comms(baud, byteFormat, id)isnotneededbutallowsforeasyupdateofthe
//portvariablesandslaveiddynamicallyinanyfunction.
modbus_update_comms(115200, SERIAL_8N1, 1);
pinMode(8, OUTPUT);
}
voidloop()
{ modbus_update();
holdingRegs[POT] = analogRead(A1);
digitalWrite(8, holdingRegs[LED]);
holdingRegs[HUM] = dht.readHumidity();
holdingRegs[TEMP] = (dht.readTemperature());
}
```