



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Matheus Reich

Comunicação serial através de rede Ethernet para Automação

Blumenau
2023

Matheus Reich

Comunicação serial através de rede Ethernet para Automação

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.

Orientador: Prof. Carlos Roberto Moratelli, Dr.

Blumenau

2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Reich, Matheus

Comunicação serial através de rede Ethernet para
Automação / Matheus Reich ; orientador, Carlos Roberto
Moratelli, 2023.

57 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Campus Blumenau,
Graduação em Engenharia de Controle e Automação, Blumenau,
2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Sockets. 3.
Modbus. 4. RS-485. 5. Internet. I. Moratelli, Carlos
Roberto. II. Universidade Federal de Santa Catarina.
Graduação em Engenharia de Controle e Automação. III. Título.

Matheus Reich

Comunicação serial através de rede Ethernet para Automação

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Engenheiro de Controle e Automação” e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 14 de julho de 2023.

Banca Examinadora:

Prof. Carlos Roberto Moratelli, Dr.
Universidade Federal de Santa Catarina

Prof. Fábio Rafael Segundo, Dr.
Universidade Federal de Santa Catarina

Prof. Guilherme Brasil Pintarelli, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado à minha família, por fornecerem o maior apoio e condições possíveis durante toda a minha vida. Também dedico este trabalho a todos os meus colegas, que foram de suma importância durante a caminhada que tivemos através do curso.

AGRADECIMENTOS

Agradeço primeiramente a minha família, em especial meus pais, Claus e Marilza, e minha irmã, Greice, que forneceram todo o apoio necessário e todas as condições possíveis para que eu pudesse evoluir pessoalmente e profissionalmente durante toda a minha vida. Minha mãe sendo professora durante toda a sua carreira, sempre cobrando o máximo esforço em todos os âmbitos do curso. Meu pai por ter interesses antigos em automação e elétrica, me incentivando seguir o caminho da engenharia. Minha irmã, por entender as dores que todos inevitavelmente passamos durante a graduação, sempre estando aberta à fornecer conselhos e apoio incondicional.

Outras pessoas que foram extremamente importantes na minha vida acadêmica são os meus amigos, os quais fiz muitos durante todo o curso. Cada um sendo uma pessoa mais especial que a outra, permitindo que em todos os ambientes e situações sempre tivesse alguém que pode me ajudar. Sem vocês toda a trajetória seria praticamente impossível, pois mesmo em dias muito difíceis, estiveram lá para apoiar e motivar. Não tenho como agradecer o suficiente ou retribuir o que devo à cada um.

Também devo agradecer ao campus Blumenau da Universidade Federal de Santa Catarina, que me permitiu obter um nível de conhecimento extremamente amplo, expandindo os campos de atuação que posso ter no futuro profissional. Não posso deixar de mencionar o Prof. Dr. Carlos Roberto Moratelli, meu orientador de ambos este trabalho e do estágio obrigatório e sobretudo professor das disciplinas que mais me interessaram durante todo o curso. Parte dos conhecimentos que foram apresentados em disciplinas lecionadas pelo Prof. Carlos foram utilizados no desenvolvimento deste trabalho.

Por último, acho extremamente importante também agradecer à Vale Automação Industrial, onde realizei o meu estágio obrigatório. Com o acesso à uma enorme gama de equipamentos de automação, consegui entender melhor toda a estrutura empregada na indústria e observar pontos que poderiam ser melhorados, resultando no projeto aqui apresentado. Através da Vale, fui possibilitado de utilizar equipamentos reais de automação, realizar os mais variados testes e sobretudo entender melhor como os equipamentos são utilizados em campo. Sem sombra de dúvida, a Vale foi fundamental em viabilizar o desenvolvimento deste TCC.

“Do what you can, with what you’ve got, where you are.” (ROOSEVELT, 1913)

RESUMO

Visando facilitar a parametrização de equipamentos voltados à automação de fábricas e máquinas industriais, propõem-se um sistema que permita o acesso remoto para realizar a configuração dos equipamentos já instalados em campo, através do uso do software adequado, sem a necessidade de conexão física da porta serial. O conceito de operação se baseia no princípio da captura dos dados seriais gerados pelo software de programação do equipamento, efetuando então o envio e recebimento destes dados por meio de *sockets* para outro computador. Este segundo computador é conectado por meio de um conversor serial ao equipamento a ser parametrizado e, mediante *sockets*, realiza o envio e recebimento dos dados do equipamento ao software de programação. A estrutura do hardware se baseia na implementação da transferência de dados mediante portas, entre dois computadores com endereços IPs distintos. Com a transmissão inicial dos dados, observou-se um problema na construção da mensagem *Modbus*, impossibilitando o reconhecimento do conteúdo por parte do equipamento que é parametrizado. Por meio da estruturação de um cabeçalho informando a quantidade de *bytes* a serem processadas, a comunicação e transmissão dos dados nos dois sentidos ocorreu sem problemas. Com esta topologia, é possível conectar-se a qualquer equipamento remoto e parametrizá-los, sem a necessidade de presença física no meio de uma planta industrial, que podem apresentar situações perigosas para a vida humana. Outra possibilidade com a implementação dessa técnica é a possibilidade de comunicação à distância do operador com o seu equipamento mediante servidores na nuvem ou por meio de VPNs, dado o objetivo de implementar um sistema compatível com qualquer rede de computadores baseados em protocolo TCP/IP.

Palavras-chave: Sockets; Inversor de frequência; Servodrives; RS-485; Modbus; Internet; Cloud; IIoT.

ABSTRACT

With the goal of facilitating the parameterization of equipment aimed at automating factories and industrial machines, a system is proposed that allows remote access to perform the parameterization of already installed equipment in the field, through the use of the appropriate software, without the need for a physical connection of a serial port. The operating concept is based on the principle of capturing serial data generated by the equipment's programming software, then sending and receiving this data through *sockets* to another computer. This second computer is connected via a serial converter to the equipment to be parameterized and, using *sockets*, sends and receives data from the equipment to the programming software. The hardware structure is based on the implementation of data transfer through ports, between two computers with different IP addresses. With the initial transmission of data, a problem was observed in the construction of the *Modbus* message, making it impossible for the equipment to be parameterized to recognize the content. By structuring a header informing the amount of *bytes* to be processed, the communication and transmission of data in both directions occurred without problems. With this topology, it is possible to connect to any remote equipment and parameterize them, without the need of a personal presence in a industrial plant, which can present dangerous situations for human life. Another possibility with the implementation of this technique is the possibility of remote communication between the operator and his equipment through servers in the cloud or through VPNs, given the objective of implementing a system compatible with any computer network based on the TCP/IP protocol.

Keywords: Sockets; Variable frequency drives; Servodrives; RS-485; Modbus; Internet; Cloud; IIoT.

LISTA DE FIGURAS

Figura 1 – Exemplo de painel elétrico aberto com vários equipamentos.	17
Figura 2 – Exemplo de painel elétrico aberto com inversores de frequência.	18
Figura 3 – Inversor de frequência Schneider Electric ATV930.	19
Figura 4 – Soft-starter Schneider Electric ATS480.	20
Figura 5 – Servodrive Schneider Electric Lexium 28.	21
Figura 6 – <i>Stack</i> de comunicação do protocolo Modbus.	23
Figura 7 – Padrão elétrico e mecânico do RS-232.	24
Figura 8 – Conector DB-9.	24
Figura 9 – Porta RS-232 no IBM PC AT.	25
Figura 10 – Conexão de dispositivos RS-422.	26
Figura 11 – Diferença entre RS-485 em 2 fios e 4 fios.	27
Figura 12 – Conexão de dispositivos em uma rede RS-485.	28
Figura 13 – Conversor de USB para RS-485.	28
Figura 14 – Tela inicial do <i>software</i> SoMove.	29
Figura 15 – Configurações avançadas da conexão serial no SoMove.	30
Figura 16 – Exemplo de parâmetros modificáveis no SoMove.	31
Figura 17 – Exemplo de ligação <i>null modem</i> para RS-232.	32
Figura 18 – Esquema de ligação da interface serial do com0com.	32
Figura 19 – Camadas e protocolos do TCP/IP.	33
Figura 20 – Construção do cabeçalho de um pacote IP.	34
Figura 21 – Construção do corpo de um segmento TCP.	35
Figura 22 – Construção do corpo de datagrama UDP.	35
Figura 23 – Portas conhecidas da rede.	36
Figura 24 – Lógica de operação do NAT.	37
Figura 25 – Topologia de VPN.	39
Figura 26 – Computador de placa única da linha Raspberry Pi.	40
Figura 27 – Diagrama tempo da proposta para a solução do problema.	42
Figura 28 – Conexão dos equipamentos relacionados ao projeto.	43
Figura 29 – Conexões virtuais dentro do Windows.	45
Figura 30 – Conexões controladas pelo computador com Linux.	45
Figura 31 – Bytes de requisição e pesquisa do equipamento d SoMove.	48
Figura 32 – Bytes recebidos através da rede e socket do computador servidor.	48
Figura 33 – Diagrama da lógica de recebimento de dados através da rede.	49
Figura 34 – Estrutura de uma mensagem do protocolo <i>Modbus</i>	50
Figura 35 – Construção incorreta enviada através da interface serial.	50
Figura 36 – Diagrama da operação do protocolo proposto.	51
Figura 37 – Quantidade de bytes e a mensagem Modbus.	52

Figura 38 – SoMove comunicando-se corretamente com o inversor de frequência. . .	52
Figura 39 – Configuração da rede por meio de VPN.	53
Figura 40 – Requisições e respostas através do VPN.	54

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
AT	Advanced Technology
CLP	Controlador Lógico Programável
DCE	Data communications equipment
DNS	Domain Name System
DTE	Data terminal equipment
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machines
IHM	Interface Homem-Máquina
IMAP	Internet Mail Access Protocol
IP	Internet Protocol
ISA	Industry Standard Architecture
NAT	Network Address Translation
OSI	Open Systems Interconnections
PC	Personal Computer
PPP	Point-to-Point Protocol
PROFIBUS	Process Field Bus
PROFINET	Process Field Network
RS	Recommended Standard
RTU	Remote Terminal Unit
SBC	Single Board Computer
SLIP	Serial Line Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VPN	Virtual Private Network

SUMÁRIO

1	INTRODUÇÃO	14
1.1	OBJETIVOS	15
1.1.1	Objetivos Específicos	15
2	TECNOLOGIAS RELACIONADAS	17
2.1	PAINÉIS ELÉTRICOS	17
2.2	INVERSORES DE FREQUÊNCIA	19
2.3	SOFT-STARTERS	20
2.4	SERVODRIVES	20
2.5	MODBUS	21
2.6	INTERFACES DE REDE SERIAL	23
2.6.1	RS-232	23
2.6.2	RS-422	25
2.6.3	RS-485	26
2.7	SOFTWARE SOMOVE	28
2.8	COM0COM	29
2.9	TCP, UDP, IP, PORTAS E SOCKETS	30
2.9.1	Camada de interface de rede	32
2.9.2	Camada da Rede	33
2.9.3	Camada de transporte	34
2.9.3.1	<i>TCP</i>	<i>34</i>
2.9.3.2	<i>UDP</i>	<i>34</i>
2.9.4	Camada de aplicação	35
2.9.5	Portas e sockets	36
2.10	NAT	37
2.11	VPN E CLOUD	38
2.12	RASPBERRY PI	39
3	PROPOSTA E ARQUITETURA DA SOLUÇÃO	41
3.1	CONFIGURAÇÃO DE PORTAS SERIAIS	43
3.2	ESTRUTURA PARA ENVIO E RECEBIMENTO DE DADOS DO <i>SOFTWARE</i> VIA SOCKETS	44
3.3	INTEGRAÇÃO COM VPN	44
4	IMPLEMENTAÇÃO E RESULTADOS	47
4.1	DESENVOLVIMENTO DOS SOFTWARES INTERCEPTADOR E GERENCIADOR	47
4.2	TESTES INICIAIS	48
4.3	ANÁLISE DA CONSTRUÇÃO DAS MENSAGENS SERIAIS	49
4.4	DESENVOLVIMENTO DE PROTOCOLO E APLICAÇÃO	50

4.5	VALIDAÇÃO DE OPERAÇÃO POR MEIO DE VPN	53
4.6	LIMITAÇÕES DA TOPOLOGIA	53
5	CONCLUSÃO	55
	REFERÊNCIAS	56

1 INTRODUÇÃO

O objetivo humano de aumentar a produtividade é algo que se encontra presente na nossa realidade desde o século XIX, quando a Revolução Industrial iniciou. A partir deste momento, as indústrias que conhecemos começaram a ser criadas, como grandes núcleos de produção de bens, composto de funcionários e máquinas, as quais auxiliavam no aumento da produtividade (SANTOS; ARAÚJO, 2016). O aumento do uso de máquinas e eletrificação dos equipamentos é o princípio da automação, podendo-se considerar que o principal objetivo era gerar maior valor com menor custo, seja este tempo ou monetário. Através do século XX, observa-se um aumento muito forte da procura no aumento da eficiência da produção, reduzindo prejuízos causados por operação inadequada de máquinas, substituição de equipamentos antigos por mais modernos, aplicação de lógicas de controle e, em especial, a digitalização dos controladores de processos e máquinas, grande parte em meados dos anos 1970 (RÜSSMANN *et al.*, 2015).

A automação e modernização de plantas industriais trazem aumento na eficiência, aumento na produção, queda no número de acidentes e redução nos custos operacionais. A instalação de inversores de frequência em motores elétricos trifásicos, por exemplo, permitem o controle completo da operação do motor, reduzindo o seu consumo elétrico e garantindo maior confiabilidade na operação da máquina. Estes inversores em sua grande parte integram protocolos de comunicação, que permitem o controle de sua operação a partir de outro equipamento, como um Controlador Lógico Programável CLP, para que ele funcione em determinada lógica de operação executada pelo CLP.

Através da possibilidade da implementação de comunicação por meios digitais com equipamentos em uma planta industrial, garantindo confiabilidade e simplicidade na produção, a instalação de equipamentos modernos com capacidade de operarem em conjunto permitem uma grande expansão da complexidade de máquinas que podem ser projetadas. Isto também permite a integração de diferentes processos, tornando assim a produção em uma linha sequencial de lógicas, controladas por diferentes equipamentos, comunicando entre si. Considerando o tamanho de máquinas modernas e suas capacidades operacionais, o tamanho da área de uma fábrica pode ser grande. Portanto, é comum ser necessária a locomoção de um operador até um equipamento para ser realizada alguma alteração no seu comportamento ou correção de uma falha, apresentando riscos ao operador. Alguns inversores de frequência ou CLPs, por exemplo, permitem sua reconfiguração mediante protocolos de comunicação mais avançados, como Modbus TCP ou PROFINET. Estes protocolos são baseados em rede Ethernet e podem ser facilmente implementados para operarem em uma grande rede, permitindo assim o acesso remoto e a configuração do equipamento à distância. Isto acaba por remover a necessidade da locomoção física do operador à máquina. Entretanto, outros protocolos como Modbus RTU ou PROFIBUS, por serem construídos sob interfaces seriais, não permitem sua instalação em estrutura

de rede ampla, tornando seu uso limitado à integração e interoperação de diferentes equipamentos com alguns painéis.

Visando evitar a necessidade de locomoção de uma pessoa através da planta industrial para realizar a parametrização de um inversor de frequência (ou qualquer outro equipamento), é apresentada uma topologia que permite a conexão remota ao equipamento que venha integrar protocolos baseados em interfaces seriais, viabilizando a parametrização dos equipamentos até mesmo através da *cloud* ou *Virtual Private Network* (VPN). Com esta topologia, um operador pode conectar-se aos seus equipamentos a partir de qualquer lugar e realizar as modificações necessárias, por meio de uma presença remota na planta industrial. Dependendo do equipamento, esta topologia permite a coleta de dados da operação para sua análise de comportamento sem a necessidade de conexão física ao equipamento, evitando, portanto distúrbios na operação da máquina.

1.1 OBJETIVOS

Simular a conexão física do inversor de frequência ao computador, no qual é executado o software de parametrização. Com esta topologia, o software não identifica que o inversor de frequência não está fisicamente conectado ao computador, garantindo a sua operação conforme especificado. Objetiva-se descrever o processo com detalhes, para permitir o entendimento de como a comunicação entre equipamentos funciona, permitindo a replicação de lógicas similares para os mais variados usos. Como resultado, planeja-se permitir a comunicação direta e sem dificuldades entre o operador e o seu equipamento, a partir de qualquer local e de forma completamente similar à conexão direta e física entre o equipamento e seu computador.

1.1.1 Objetivos Específicos

- Estudar como a comunicação de um software de programação funciona;
- Estudar como a rede serial RS-485 é implementada;
- Analisar a construção de um pacote de dados do protocolo *Modbus*;
- Discutir diferentes meios para implementar a comunicação através da Internet;
- Determinar a viabilidade de generalização do uso da topologia desenvolvida;
- Estudar a operação de *sockets* entre o Windows e sistemas operacionais baseados em Linux
- Determinar como é a configuração padrão para a comunicação com equipamentos em protocolo *Modbus Remote Terminal Unit* (RTU)/RS-485;
- Especificar o hardware e software necessário para replicar a aplicação em diferentes ambientes computacionais;

- Desenvolver soluções em software para realizar a comunicação;
- Analisar o desempenho necessário que permita a conexão estável entre o hardware envolvido.

2 TECNOLOGIAS RELACIONADAS

Neste capítulo, estão presentes os conteúdos relevantes para o desenvolvimento da solução do problema sugerido.

2.1 PAINÉIS ELÉTRICOS

Os painéis elétricos são utilizados nas plantas industriais visando centralizar, padronizar e facilitar a instalação dos equipamentos necessários para permitir a automação de uma determinada máquina ou processo. Em sua grande parte, é onde se concentra os circuitos de comando e de potência, permitindo o controle dos equipamentos por meio de botões ou IHMs, que facilitam a interação com o processo, como observado na Figura 1.

Figura 1 – Exemplo de painel elétrico aberto com vários equipamentos.



Fonte: Vale Automação Industrial (2023).

Figura 2 – Exemplo de painel elétrico aberto com inversores de frequência.



Fonte: Vale Automação Industrial (2023).

Portanto, os painéis são responsáveis por viabilizar a modernização de plantas industriais, facilitando a concentração das informações relevante ao processo. Atualmente, em painéis elétricos, implementa-se na maioria dos projetos a instalação de manoplas estendidas para os disjuntores ou chaves seccionadoras, que permitem desenergizar o painel externamente e, caso a porta seja aberta, todo o painel é desenergizado por meio de um circuito voltado apenas a segurança. A comunicação dos equipamentos instalados em painéis, quando possuem esta opção, é feita por meio de cabos. Nos casos onde o controle do processo é feito por um Controlador Lógico Programável CLP em outro equipamento, o barramento de rede é instalado mediante bifurcações (dependendo do protocolo utilizado), chegando até o CLP, por exemplo.

2.2 INVERSORES DE FREQUÊNCIA

Inversores de frequência são utilizados para efetuarem a partida e frenagem de motores, assim como efetuar o controle durante toda a sua operação. Por serem equipamentos que retificam as correntes alternadas trifásicas para corrente contínua, gerando então uma nova onda de corrente alternada a partir da corrente contínua, com base nos parâmetros processados pelo microprocessador do dispositivo, podem manipular todos os aspectos dos motores conectados ao inversor (KALE *et al.*, 2017). Controlam a tensão, frequência, corrente e outros aspectos da alimentação trifásica fornecida ao motor, garantindo assim o controle completo da operação do motor, fornecendo ao operador todos os tipos de informações possíveis sobre o motor instalado. Um exemplo de inversor de frequência pode ser observado na Figura 3. Exemplos instalados em painéis elétricos estão presentes na Figura 2.

Figura 3 – Inversor de frequência Schneider Electric ATV930.



Fonte: Schneider Electric (2023).

Os inversores de frequência da Schneider Electric, por exemplo, possuem comunicação *Modbus* RTU de fábrica, permitindo a conexão destes equipamentos diretamente com CLPs da própria Schneider Electric (ou outras marcas se compatíveis), controlando a sua operação sem a necessidade da intervenção humana no processo.

2.3 SOFT-STARTERS

Estes equipamentos são amplamente utilizados nos casos onde é necessário apenas efetuar a partida e frenagem de motores controladamente, permitindo controlar o torque ou tempo/curva da aceleração e parada. Não são avançados em sua operação comparado ao inversor de frequência. Um exemplo de soft-starter pode ser observado na Figura 4.

Figura 4 – Soft-starter Schneider Electric ATS480.



Fonte: Schneider Electric (2023).

Assim como nos inversores de frequência, a Schneider Electric inclui a comunicação através do protocolo *Modbus* RTU nos seus soft-starters, permitindo que o mesmo seja controlado por através de outros equipamentos, como CLPs.

Os soft-starters operam através do uso de *tiristores* para variar a tensão aplicada nos motores trifásicos, permitindo assim uma partida com menor corrente que outros métodos menos complexos (estrela-triângulo ou partida direta) (GRITTER; WANG; HABELTLER, 2000). Portanto, o objetivo no seu uso é fornecer um produto de menor custo que um inversor de frequência, entretanto permitir uma partida/parada mais suave dos motores e com menor corrente.

2.4 SERVODRIVES

Os servodrives são controladores específicos para servomotores, que são motores síncronos com encoders integrados. Com o *feedback* fornecido pelo encoder, o servodrive consegue gerenciar o posicionamento exato do motor assim como aplicar torque constante

no eixo com base na carga aplicada (PFAFF; WESCHTA; WICK, 1984). Um exemplo de servodrives pode ser observada na Figura 5.

Figura 5 – Servodrives Schneider Electric Lexium 28.



Fonte: Schneider Electric (2023).

Os servodrives, por possuírem uma característica de operação mais sensível e ser um equipamento com alto nível de complexidade, geralmente possuem protocolos de comunicação mais avançados. São geralmente integrados com o protocolo *CANopen* e em certos dispositivos, com *SERCOS*. Por último, um método mais simples de comunicação e controle é o uso de sinais analógicos para definir a presente posição do eixo do servomotor. Estes sinais são geralmente interpretados por CLPs. No caso, há também modelos que integram a rede de comunicação *Modbus RTU*, porém por esta ser um protocolo de menor velocidade de comunicação, é geralmente limitado ao uso básico, como velocidade, ir/parar e outras informações do equipamento.

2.5 MODBUS

O protocolo Modbus foi criado pela empresa *Modicon* em 1979. Atualmente a *Modbus Organization* gerencia a evolução do protocolo em comum acordo entre todos os membros da organização. Entretanto, por ser desde sua origem um protocolo aberto, é integrado nos mais variados equipamentos e dispositivos de uma grande gama de marcas (SOUZAO, 2008). Utiliza um padrão de comunicação baseado no conceito mestre-escravo, onde apenas um equipamento no barramento (mestre) realiza as requisições de informações dos dispositivos (escravos).

É atualmente o protocolo mais popular em todo o ambiente industrial, com mais de 7 milhões de nós de barramento Modbus implementados apenas na América do Norte e Europa (MODBUS ORGANIZATION, 2023). Devido a sua simplicidade de operação e capacidade de abranger grande parte das demandas apresentadas na indústria, é um protocolo que apesar da idade e simplicidade em relação a protocolos mais modernos, ainda consegue ser implementado em larga escala. Sua falta de segurança, robustez dos dados e expansão de tipos de variáveis se apresentam como fatores que o tornam desinteressante para alguns usos, como em ambientes de plantas químicas.

Independente disto, o protocolo Modbus é construído com uma mensagem simples, onde os dados podem ser construídos a partir de caracteres no padrão *American Standard Code for Information Interchange* (ASCII), com 7 bits de comprimento, ou então no modo *Remote Terminal Unit* (RTU), onde cada mensagem é construída com dois caracteres hexadecimais de 4 bits.

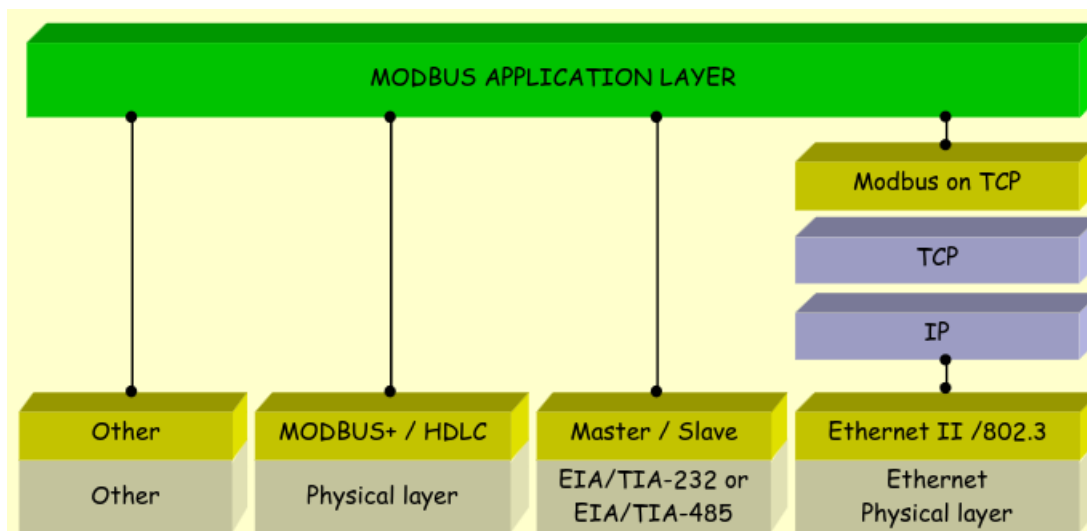
O protocolo Modbus pertence à camada de aplicação e pode ser implementada em diferentes barramentos/redes:

- TCP/IP: o protocolo usa a rede Ethernet, enviando seus pacotes de mensagens sob a rede entre dois endereços do *Internet Protocol* (IP), por meio da porta 502. Esta porta é de uso exclusivo para Modbus TCP;
- Transmissão serial assíncrona sob vários meios: por ser necessário apenas conseguir enviar as mensagens padronizadas pelo protocolo Modbus, os dados podem ser enviados/recebidos sob qualquer tipo de rede, como rádio, fibra, RS-232, RS-422 ou RS-485. O último é o mais utilizado na indústria, sendo considerado um “padrão” do protocolo Modbus RTU.

O Modbus é estruturado seguindo o *stack* de comunicação observável na Figura 6, seguindo um mesmo protocolo implementado em diferentes interfaces ou redes. Com isto, a implementação em diferentes equipamentos é facilitada, pois o mesmo conjunto de instruções é utilizado, independente de qual meio de comunicação é implementado.

Para realizar a transmissão dos dados, o dispositivo mestre do barramento Modbus faz uma requisição a um determinado endereço de escravo. São suportados endereços entre 1 a 247, sendo este o máximo de dispositivos suportados no Modbus. Há basicamente duas tarefas que podem ser realizadas, sendo escrita ou leitura. Com a requisição do mestre a um determinado endereço de escravo, informando então em qual posição/local da memória deseja realizar esta requisição, o escravo então executa a tarefa e retorna uma resposta. Esta resposta pode ser o valor requisitado, uma confirmação do escravo para o mestre que o valor foi “escrito” na memória, ou então um erro informando que houve uma falha (o endereço desejado não existe ou a memória não pode ser escrita, por exemplo). Desta forma simples, um barramento serial simples que suporte múltiplos dispositivos é capaz facilitar a integração de uma abundância de dispositivos em um único protocolo de rede.

Mais detalhes sobre a construção de uma mensagem baseada no protocolo Modbus

Figura 6 – *Stack* de comunicação do protocolo Modbus.

Fonte: Modbus Organization (2012).

RTU serão abordados nas seções futuras.

2.6 INTERFACES DE REDE SERIAL

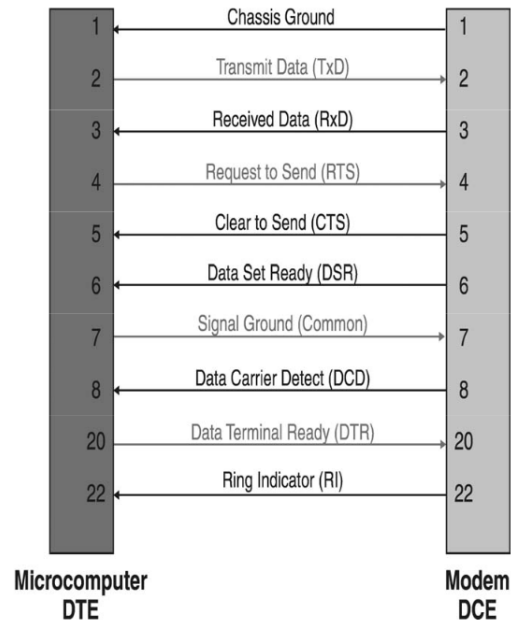
As redes de comunicação serial foram a base na comunicação entre diferentes computadores durante grande parte dos anos 1960 até meados dos anos 1990. O seu uso foi substituído nos computadores por outros tipos de barramentos, como o USB ou a porta *Ethernet*. Entretanto, mesmo com a evolução de outros meios de comunicação, a porta serial persiste presente em computadores de mesa, muitas vezes como um opcional. Praticamente todas as portas seriais de computadores utilizam um padrão definido há mais de 40 anos, sendo raros os casos onde outras interfaces são empregadas. Neste caso, trata-se do padrão conhecido como RS-232.

2.6.1 RS-232

Na sua nomenclatura, o *Recommended Standard* (RS) implica que era apenas um padrão recomendado, facilitando o uso entre dispositivos de diferentes marcas, aumentando a integração e facilitando a expansão de redes de servidores e terminais de usuários, propósito o qual este padrão foi inicialmente desenvolvido para operar. É importante salientar que o padrão RS-232 define apenas uma interface, não um protocolo. Portanto, apenas os detalhes elétricos e mecânicos do padrão foram especificados, mas a construção das mensagens enviadas/recebidas não. Desta forma, qualquer tipo de protocolo pode ser implementado sob uma mesma interface, como, por exemplo, o já mencionado Modbus RTU. Para o RS-232, foram especificados apenas a conexão entre dois dispositivos, o *Data*

Terminal Equipment (DTE) e o *Data Carrier Equipment* (DCE), que seguem a ligação elétrica e mecânica apresentada na Figura 7.

Figura 7 – Padrão elétrico e mecânico do RS-232.



Fonte: Boston Technology (2020).

O conceito seria no qual o DTE é um dispositivo como um microcomputador e o DCE seria um modem para acesso a uma rede de computadores via ligação telefônica, por exemplo. O conector mais comum a ser utilizado com o RS-232 passou a ser o DB-9 (Figura 8), que foi primeiramente padronizado pela *International Business Machine* (IBM) no lançamento da linha de computadores conhecidos como IBM PC AT, como poder ser visto na expansão *Industry Standard Architecture* (ISA) do computador (slot 6), na Figura 9.

Figura 8 – Conector DB-9.



Fonte: Eltima (2020).

Com a padronização difundida pela IBM, o tipo de porta serial permanece o mesmo até os dias atuais nos computadores que possuem esta porta. Em raras situações há a presença de outro tipo de padrão serial disponível para ser utilizado pelo usuário, entretanto por ser tão difundido, seu uso é constante em microcontroladores como o *Arduino*, que possui conexões *RX/TX*, implementando o padrão RS-232.

Figura 9 – Porta RS-232 no IBM PC AT.



Fonte: Vintage Computer (2021).

2.6.2 RS-422

Com o avanço da eletrônica digital, novos padrões foram desenvolvidos para substituir os mais antigos, como o RS-449 ou o RS-423. Ambos não são comuns atualmente, sendo amplamente ignorados por fornecerem poucas melhorias em relação ao já estabelecido RS-232. Ainda no início dos anos 1970, o RS-422 foi disponibilizado. Este padrão define uma interface de comunicação de dados balanceado ou diferencial, usando dois fios separados para cada sinal, permitindo altas taxas de transmissão e reduzindo problemas de aterramento, pois esta interface não usa o aterramento como referência de tensão, mas sim a relação entre os dois fios de comunicação. O RS-422 é considerado uma melhoria do RS-423, permitindo:

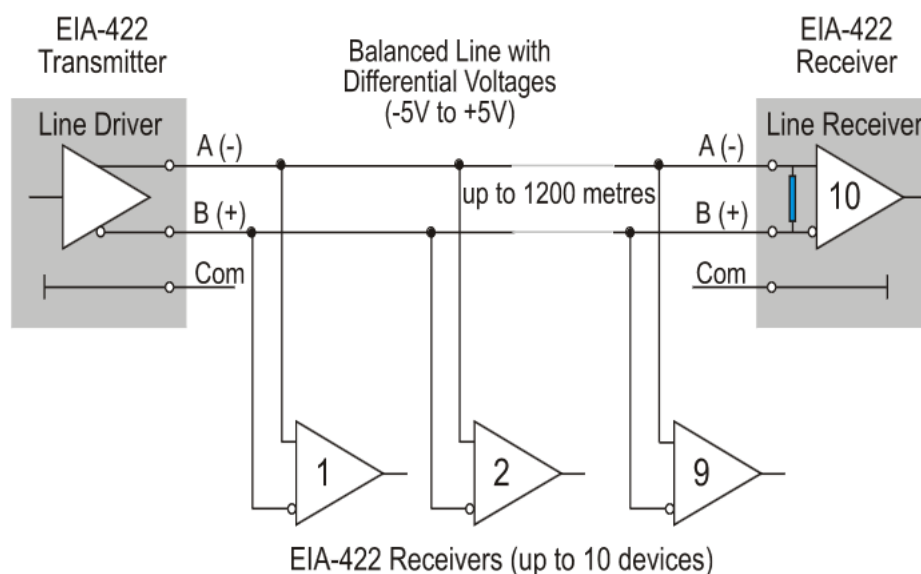
- Dados serem transmitidos com distâncias de até 1200 m, o que era similar ao RS-423;
- Taxas de transmissão de até 10 Mbps, um aumento de 100 vezes em relação ao anterior;
- Apenas um controlador/emissor de rede por barramento;
- Até 10 receptores, que eram controlados pelo emissor;

Por operar com a diferença de potencial de $\pm 5V$ entre dois fios ($-6 V$ a $+6 V$), com lógicas representando os sinais binários, o RS-422 utiliza a nomenclatura de linhas A (ou -) e B (ou +), diferentemente do RS-232, que seriam apenas RX e TX, com os bits 0 e 1 sendo representados por sinais $0 V$ (baixo) e $+ 5 V$ (bit 1, alto), respectivamente. A representação binária no RS-422 é dada por:

- $-2 V$ a $-6 V$ em relação à linha B para representar um sinal binário 1 (alto);
- $+2 V$ a $+6V$ em relação à linha B para representar um sinal binário 0 (baixo).

O RS-422 se tornou bastante comum no uso industrial, fornecendo uma simples interface para conectar vários equipamentos entre si e permitindo o controle de todos sem a necessidade de redes complexas. A lógica de conexão dos dispositivos ao transmissor/emissor é apresentada na Figura 10 e, como pode-se observar, é bastante simples. Algumas marcas de equipamentos de automação, como a *Mitsubishi Electric*, ainda utilizam este protocolo em alguns de seus dispositivos.

Figura 10 – Conexão de dispositivos RS-422.



Fonte: Boston Technology (2020).

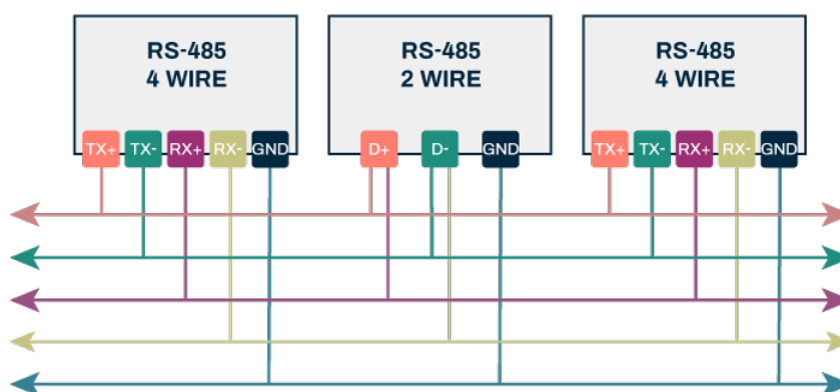
2.6.3 RS-485

Visando expandir o número de dispositivos conectados ao mesmo barramento, desenvolveu-se o padrão RS-485. O objetivo inicial do RS-485 seria manter as mesmas características de distância máxima (1200 m), taxas de transmissão (10 Mbps), mas aumentar o número de controlador/emissores para 32 dispositivos, assim como os receptores para até 32 dispositivos, todos em um mesmo barramento. A maior diferença entre o RS-422 e o RS-485 é que o último permite que os controladores do barramento entrem em

um modo de alta impedância, conhecido como modo “desativado”. Com isto, é possível alternar entre os 32 controladores do barramento, mas tendo certeza que somente um destes está ativo por vez. Com alta coordenação entre os dispositivos controladores do barramento, pode-se alternar entre todos para operarem em conjunto. Outras pequenas diferenças entre o RS-422 e o RS-485 existem, mas não são relevantes para este capítulo.

Diferentemente do RS-232 ou RS-422, o RS-485 possui um modo de operação de 4 fios, conhecido como “*full-duplex*”, onde o envio dos dados é realizado em um par de fios e o recebimento de dados em outro par de fios. Esta construção de 4 fios tem o objetivo de criar um sinal diferencial para cada sentido dos dados, aumentando a sua robustez eletromagnética. A diferença para o modo 2 fios, ou “*half-duplex*”, é que neste o envio e recebimento dos dados ocorre sob o mesmo par de fios, mas também com um sinal diferencial. A diferença entre os dois meios de comunicação pode ser visualmente interpretada na Figura 11 e a construção da conexão de múltiplos dispositivos no modo em 2 fios, a qual seria a mais comum na indústria, pode ser observada na Figura 12. A ligação entre um dispositivo mestre e seus escravos é realizado por uma simples conexão paralela, tornando assim a sua implementação bastante simples.

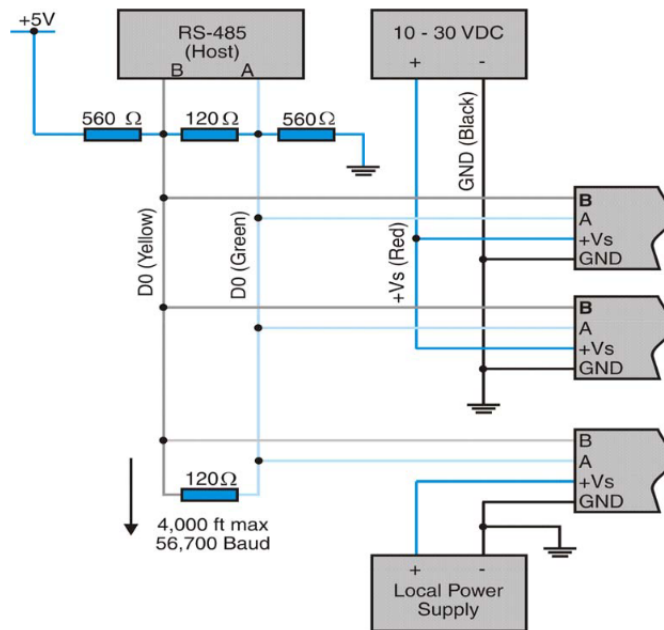
Figura 11 – Diferença entre RS-485 em 2 fios e 4 fios.



Fonte: Altium (2021).

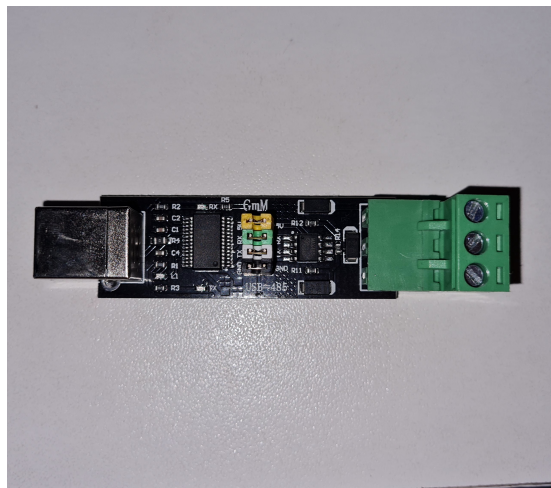
Por ser um meio protocolo de comunicação empregado atualmente, há disponível no mercado vários tipos de conversores *Universal Serial Bus* (USB) para RS-485 de baixo custo, como o exemplo observado na Figura 13, permitindo facilmente efetuar a conexão de equipamentos que utilizam esta interface aos computadores modernos. O seu uso é bastante comum na conexão de inversores de frequência e *soft-starters* a computadores, para permitir a comunicação direta de *softwares* com estes equipamentos. É possível assim parametrizá-los mais rapidamente, efetuar backups de suas configurações, assim como acompanhar a sua operação.

Figura 12 – Conexão de dispositivos em uma rede RS-485.



Fonte: Boston Technology (2020).

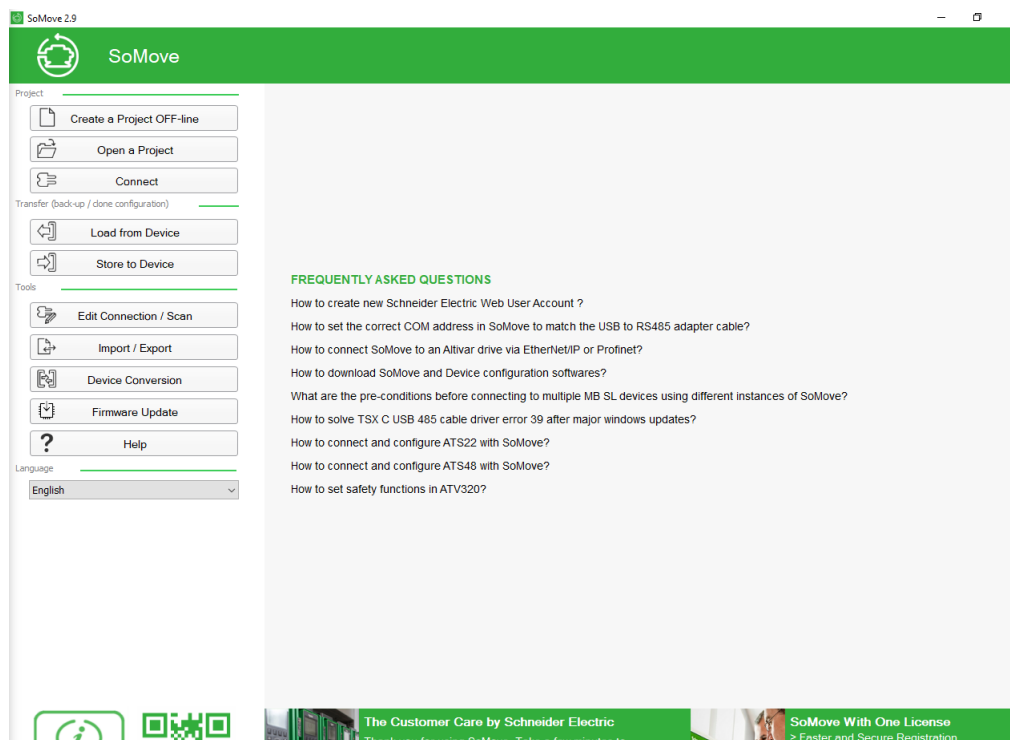
Figura 13 – Conversor de USB para RS-485.



Fonte: Do autor (2023).

2.7 SOFTWARE SOMOVE

Para efetuar a parametrização de inversores de frequência, *soft-starers* e servodrivens, o *software* fornecido para parametrizar os equipamentos da Schneider Electric é conhecido como *SoMove*. Este *software* é responsável por comunicar, identificar, parametrizar e efetuar o backup dos parâmetros dos equipamentos previamente mencionados, sendo gratuito e disponível no site da Schneider Electric para realizar o download.

Figura 14 – Tela inicial do *software* SoMove.

Fonte: Do autor (2023).

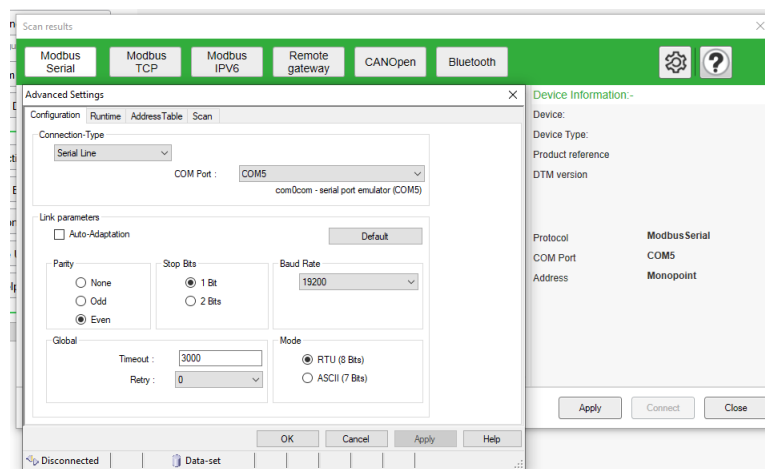
Este *software* permite efetuar a conexão com os seus dispositivos mediante uma porta serial disponível no computador. Um conversor USB/RS-485, como o modelo apresentado anteriormente, pode ser utilizado para conectar o equipamento a ser parametrizado ao computador. As configurações avançadas do serial podem ser observadas na Figura 15. Todos os detalhes mencionados na tela são relacionadas diretamente com o comportamento da interface serial.

Com o modelo do equipamento a ser parametrizado já configurado no SoMove, pode-se então efetuar as modificações necessárias (na tela observada na Figura 16), aplicá-las e então realizar o *download* dos parâmetros (em relação ao equipamento). O *software* dispara os pacotes formatados em protocolo *Modbus* e, com base nas configurações aplicadas na tela observada na Figura 15, aguarda um retorno de confirmação do equipamento. Esta resposta do dispositivo é primordial para informar que as informações foram salvas corretamente, evitando assim problemas graves. Com o equipamento conectado, é possível observar detalhes da operação, como corrente, tensão, frequência, entre outros.

2.8 COM0COM

Para permitir a manipulação dos dados de portas seriais em um ambiente *Windows*, é necessário ou possuir portas seriais físicas/reais disponíveis no hardware do computador.

Figura 15 – Configurações avançadas da conexão serial no SoMove.



Fonte: Do autor (2023).

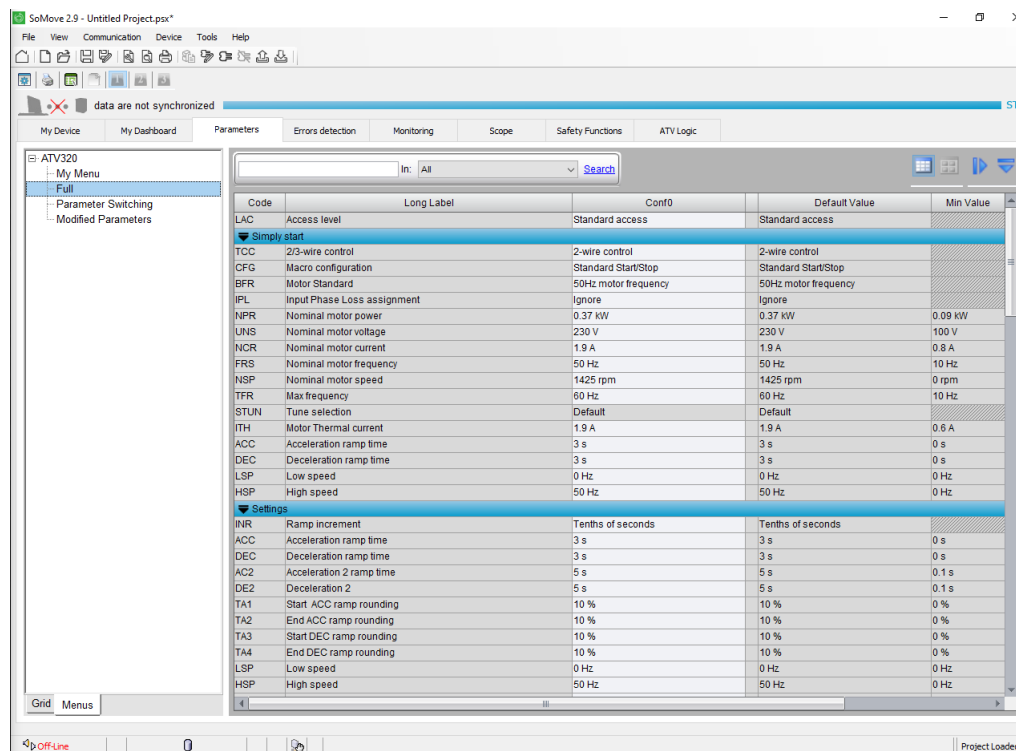
Dada a defasagem deste tipo de porta nos computadores reais, torna-se necessário o uso de conversores USB, ou então a criação de portas virtuais internas. O uso de portas virtuais internas permitem a simulação da conexão física entre dois *softwares*, podendo ser utilizado, por exemplo, para conectar dois *softwares* que se comunicam somente via portas seriais. A criação de portas seriais virtuais no Windows deve ser realizado por meio de serviços executados ao nível de *kernel*, não sendo possível criá-las em um processo comum. É com este propósito que o projeto “open-source” *com0com* é utilizado, pois ele instala um driver no kernel do *Windows*, que permite criar pares de portas seriais interconectadas (COM0COM PROJECT, 2023). Estas portas são “conectadas” ao nível de *software* em um método conhecido como “*null modem*”, inicialmente utilizado para conectar dois DTEs, portanto dois computadores, surgindo daí a ideia de um “modem nulo” (não existente) (MICHAEL A. BANKS, 1988). Um exemplo de ligação entre dois dispositivos seriais no padrão *null modem* pode ser observado na Figura 17.

Com a ligação *null modem*, é possível que um *software* controle uma das portas seriais, sendo a outra porta seja controlada por outro *software*. Portanto, pode-se assim conectar dois programas que possivelmente apenas têm comunicação por portas seriais. O esquema de ligação simulado pelo *com0com* pode ser observado na Figura 18. Desta forma, o *com0com* é um *software* utilizado principalmente para testes com comunicação serial e a intercomunicação de dispositivos virtuais ou programas.

2.9 TCP, UDP, IP, PORTAS E SOCKETS

Os meios de comunicação de grande parte dos dispositivos atuais são baseados nos protocolos *TCP/IP* e, respeitando estes protocolos, pode-se conectar diferentes equipamentos entre si. Através das padronizações dos protocolos, que ocorreram em meados dos anos

Figura 16 – Exemplo de parâmetros modificáveis no SoMove.



Fonte: Do autor (2023).

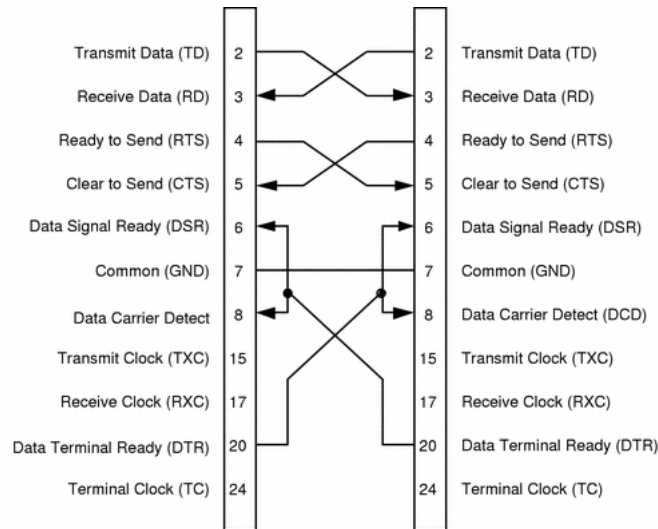
1980, usuários conseguiam conectar-se mediante servidores, para realizarem a troca de dados. Este é o princípio da *Internet* como conhecemos atualmente. A um nível bastante básico, o *Internet Protocol* (IP) é uma linguagem padrão entre bilhões de dispositivos, permitindo a comunicação entre estes de forma simples e eficiente.

Diferentemente de uma conexão serial, onde o objetivo é a transmissão local de dados (entre um mestre e vários escravos, ou então dois dispositivos), a construção dos protocolos TCP/IP serve o propósito de permitir a conexão remota de computadores (ou quaisquer dispositivos) entre si. Implementa um padrão muito mais robusto dos pacotes de dados, com correções de erros, por exemplo, sendo este algo inexistente na troca de dados das interfaces RS-232 ou RS-485.

Para padronizar os tipos de trocas de informações, permitindo assim um comportamento uniforme por meio de toda a rede, a arquitetura TCP/IP define protocolos internos. Visando separar as tecnologias envolvidas, os protocolos são divididos em “camadas” (*stack* em inglês), conforme podem ser observados na Figura 19.

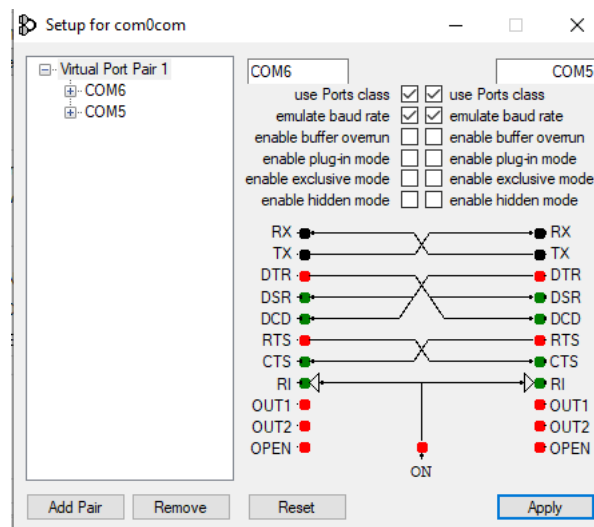
O melhor entendimento e aprofundação na apresentação dos protocolos da camada de aplicação (*Application Layer*) apresentados na Figura 19 saem do escopo deste documento, sendo então apenas aprofundado o conhecimento sobre os protocolos TCP, UDP, IP, assim como as camadas. A divisão entre camadas facilita o entendimento e a função de cada um dos protocolos, sendo estes descritos nas próximas seções.

Figura 17 – Exemplo de ligação *null modem* para RS-232.



Fonte: Oracle (2010).

Figura 18 – Esquema de ligação da interface serial do com0com.



Fonte: Do autor (2023).

2.9.1 Camada de interface de rede

Como os protocolos TCP/IP foram desenvolvidos considerando a possibilidade de serem implementados nos mais variados tipos de meios de transmissão (rádio, chamada telefônica, sem-fio, entre outros), alguns protocolos que permitem essa implementação “agnóstica” foram criados. O *Serial Line Internet Protocol* (SLIP) e o *Point-to-Point Protocol* (PPP) são ambos implementados para permitir que um computador remoto possa se conectar diretamente a um servidor através do IP, ao invés de uma conexão

Figura 19 – Camadas e protocolos do TCP/IP.

Application Layer	HTTP POP3/IMAP Time/NTP	FTP SMTP Whois	Telnet Finger Gopher TACACS+	SSH BGP SSL	DNS DNS RADIUS Traceroute	SNMP Archie tftp	RIP Ping
Transport Layer			TCP		UDP		ICMP OSPF
Internet Layer				IP			ARP
Network Interface Layer	Ethernet/802.3 Frame Relay Fibre Channel PPP	Token Ring (802.5) SMDS DDS/DS0/T-carrier/E-carrier HDLC	ATM Wireless (WAP, CDPD, 802.11) SONET/SDH SLIP/CSLIP	SNAP/802.2 X.25 FDDI ISDN DWDM Cable Modem (DOCSIS)			

Fonte: Gary Kessler (2010).

assíncrona (interface serial).

O SLIP é implementado como um protocolo para enviar/receber os bytes de um pacote TCP ou datagrama UDP em uma *stream* de dados em uma interface serial, provendo a conexão à Internet por meio de modems/roteadores. Era bastante implementado com computadores quando estes eram conectados à rede através de suas portas seriais, padronizado para facilitar a intercomunicação de equipamentos da rede e os computadores. Foi amplamente substituído pelo PPP quando as conexões à Internet nos computadores começaram a migrar para a porta Ethernet. Entretanto, o SLIP ainda é bastante empregado em microcontroladores, pela sua simplicidade.

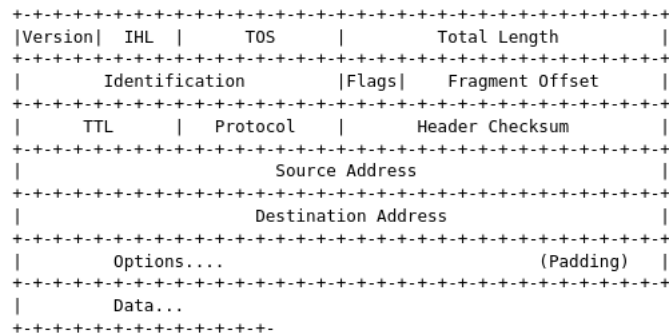
O uso do PPP permite que os dados sejam encapsulados e enviados por um meio de transmissão até um servidor, estabelecendo assim uma conexão. Este protocolo permite que vários parâmetros da comunicação, como criptografia, endereços IP, métodos de compressão e encriptação sejam pré-determinados na troca de informações, sem mesmo a necessidade da pré-determinação de um endereço IP dos dispositivos conectados. Desta forma, por permitir toda uma organização prévia da comunicação, facilidade no uso em dispositivos plugáveis (*plug-and-play*) como o caso das redes Ethernet que temos nos computadores, o PPP substituiu o SLIP, onde o último não permitia estas facilidades.

2.9.2 Camada da Rede

Os serviços fornecidos por esta camada são similares a camada de rede do modelo *Open Systems Interconnection* (OSI), este sendo estabelecido em 1983, dividindo os meios de comunicação em diferentes camadas, similarmente a observada neste capítulo. A construção do cabeçalho de um pacote IP pode ser observado na Figura 20.

O objetivo principal do pacote IP é identificar qual seria o endereço do dispositivo remetente e do destinatário, com vários dados fornecidos em conjunto que venham permitir

Figura 20 – Construção do cabeçalho de um pacote IP.



Fonte: Gary Kessler (2010).

a interpretação correta da informação recebida. Portanto, o objetivo desta camada é ofertar meios de diferenciar os equipamentos disponibilizados em uma rede.

2.9.3 Camada de transporte

Esta camada possui os protocolos voltados a transmissão dos dados em si, os quais são empacotados como *Transmission Control Protocol* (TCP) ou *User Datagram Protocol* (UDP). São similares às camadas de transporte e de sessão do modelo OSI.

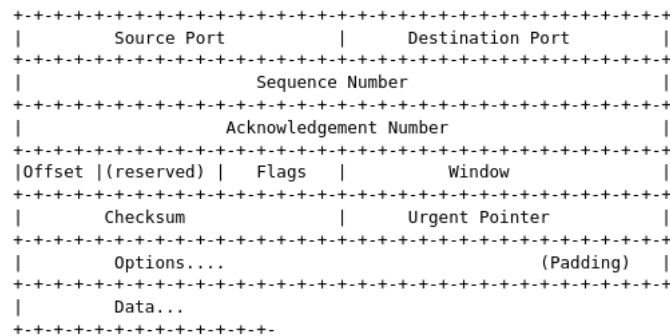
2.9.3.1 TCP

É o protocolo mais utilizado pelos serviços da camada de aplicação (será abordada na seção 2.9.4), por garantir um controle do fluxo de pacotes, sequenciamento correto dos pacotes e a correção de erros. As unidades de dados do TCP são chamados de segmentos, pois o TCP não reconhece as mensagens, mas realiza apenas o transporte do *stream* de dados entre os dispositivos. Portanto, não é relevante reconhecer o comprimento ou tipo de informação enviada, pois todos os *bytes* são enviados na ordem correta (sequenciamento) e garantidamente (controle de fluxo, onde há o controle do envio de pacotes caso não ocorra o recebimento da confirmação de que os pacotes chegaram ao destinatário). O formato padrão de um segmento (corpo) do TCP é apresentado na Figura 21.

2.9.3.2 UDP

Os datagramas UDP é de construção mais simples que o TCP, sendo geralmente mais utilizados para o envio de dados ponto-a-ponto, que servem para efetuar requisições e recebimento de respostas curtas. Em certos casos, pelo fato de ser um protocolo mais simples, há menos tempo gasto com a criação de *streams* de dados, como no caso do TCP. Não fornece nenhum tipo de controle de fluxo de pacotes, nem sequenciamento correto de

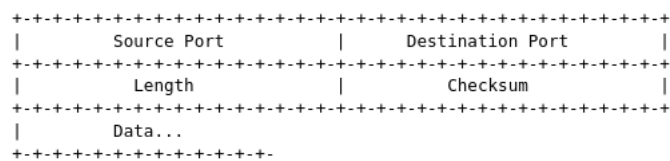
Figura 21 – Construção do corpo de um segmento TCP.



Fonte: Gary Kessler (2010).

dados, sendo então evitado em demandas onde as informações precisam chegar com certa ordem. A construção de um datagrama UDP pode ser observado na Figura 22.

Figura 22 – Construção do corpo de datagrama UDP.



Fonte: Gary Kessler (2010).

2.9.4 Camada de aplicação

Como pode-se observar na Figura 19, a quantidade de protocolos presentes na camada de aplicação é bastante extensa, notando-se que estes são apenas alguns dos mais comuns. Portanto, é interessante apenas mencionar alguns dos mais utilizados, para maior conhecimento. São estes:

- *Domain Name System* (DNS): serviço utilizado para converter endereços alfanuméricos em endereços IP, permitindo que o usuário digite um endereço da *Web* como “www.ufsc.br” e, através do DNS, o endereço é automaticamente convertido para o endereço IP “150.162.2.10”;
- *Hypertext Transfer Protocol* (HTTP): este protocolo é a base da troca de informações na “*World Wide Web*” (Internet), onde as páginas que trafegam neste protocolo são processadas através do *Hypertext Markup Language* (HTML);
- *Internet Message Access Protocol* (IMAP): serviço utilizado para efetuar a conexão a servidores de *e-mail* online, efetuando o download destes para um programa no computador do usuário, como o “Outlook” da Microsoft.

2.9.5 Portas e sockets

As portas são utilizadas para facilitar o direcionamento dos pacotes TCP e UDP. Estas, junto ao endereço IP, compõe a funcionalidade dos chamados “*sockets*”, os quais são interfaces do sistema operacional que permitem um processo enviar e receber dados através da rede de comunicação. O *socket* do remetente e o *socket* do destinatário formam a base das comunicações entre dois dispositivos, sendo:

- Porta do remetente;
- Endereço IP do remetente;
- Porta do destinatário;
- Endereço IP do destinatário.

Algumas portas são reservadas para usos exclusivos, como já mencionado anteriormente no capítulo 2.5, onde o protocolo *Modbus TCP* utiliza a porta 502. Alguns outros serviços populares também tem suas portas padronizadas, como listado na Figura 23.

Figura 23 – Portas conhecidas da rede.

Port #	Common Protocol	Service	Port #	Common Protocol	Service
7	TCP	echo	80	TCP	http
9	TCP	discard	110	TCP	pop3
13	TCP	daytime	111	TCP	sunrpc
19	TCP	chargen	119	TCP	nntp
20	TCP	ftp-control	123	UDP	ntp
21	TCP	ftp-data	137	UDP	netbios-ns
23	TCP	telnet	138	UDP	netbios-dgm
25	TCP	smtp	139	TCP	netbios-ssn
37	UDP	time	143	TCP	imap
43	TCP	whois	161	UDP	snmp
53	TCP/UDP	dns	162	UDP	snmp-trap
67	UDP	bootps	179	TCP	bgp
68	UDP	bootpc	443	TCP	https (http/ssl)
69	UDP	tftp	520	UDP	rip
70	TCP	gopher	1080	TCP	socks
79	TCP	finger	33434	UDP	traceroute

Fonte: Gary Kessler (2010).

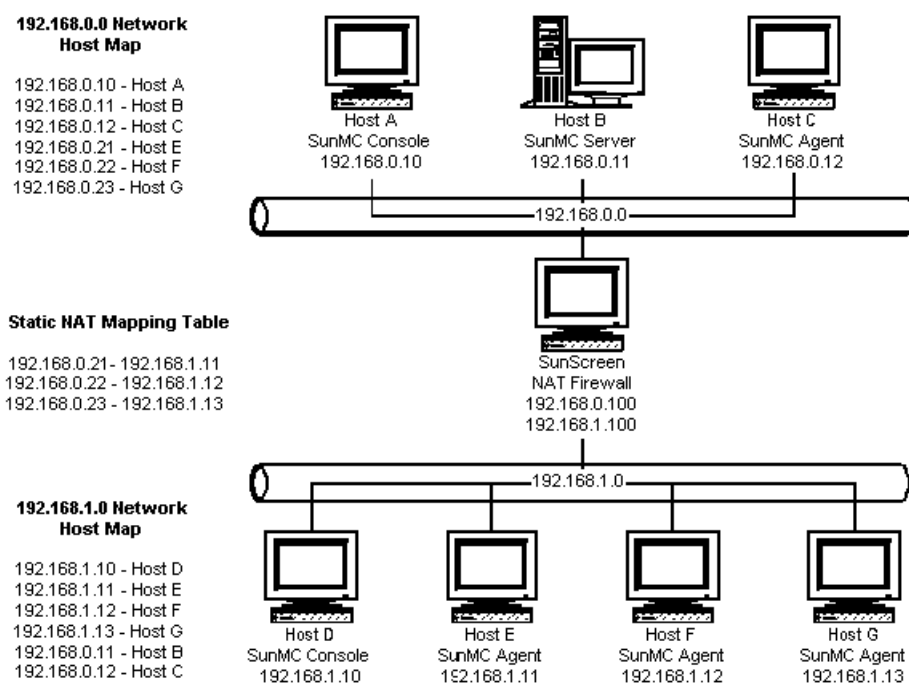
Sendo assim, com um programa desenvolvido em um computador, onde este permanece “ouvindo” em uma determinada porta da rede, é possível receber qualquer tipo de informação. Assim como é possível “ouvir”, pode-se também escrever em uma porta, sabendo o endereço IP do dispositivo no qual será enviado os dados. As funções de bibliotecas nativas dos sistemas operacionais que implementam os *sockets* realizam as tarefas de estabelecer e manter aberta a conexão entre dois dispositivos, mantendo um canal

de comunicação aberto constantemente entre dois computadores em uma mesma rede. Permitem assim a troca de informações, como arquivos, por exemplo.

2.10 NAT

Para permitir que a enorme quantidade de dispositivos e computadores que temos conectados na rede atualmente possam continuar conectando-se entre si, um mecanismo chamado de *Network Address Translation* (NAT) é implementado geralmente em roteadores e modems, por exemplo. A função deste mecanismo é realizar a conversão de um endereço IP de uma rede interna para o endereço IP da rede externa (público), definido pelo provedor de Internet. O endereço IP da rede interna não é visível na Internet, sendo apenas possível identificar o endereço IP público (KESSLER, 1994-2019). Com este mecanismo de tradução, a quantidade de dispositivos que podem se conectar à Internet são amplificados, pois apenas um endereço IP público pode possuir milhares de dispositivos atrelados na rede interna.

Figura 24 – Lógica de operação do NAT.



Fonte: Sun Microsystems (2010).

No exemplo apresentado na Figura 24, há três computadores conectados a um roteador que está na rede “192.168.0.0” com sub-máscara “255.255.255.0” (também conhecido como máscara de rede “/24” ao final do endereço IP) e outro roteador com endereço “192.168.1.0/24”. Tecnicamente, estas redes não conseguem comunicar-se entre si, por possuírem apenas no máximo 254 endereços disponíveis a partir de 192.168.X.1

até 192.168.X.254 (com X sendo 0 ou 1 para as redes mencionadas). Ou seja, o computador “Host A” (com endereço 192.168.0.10) não consegue enviar dados diretamente ao computador “Host E” (com endereço 192.168.1.11), por estarem localizados em sub-redes diferentes. O que ocorre neste caso é que o computador “Host A” sabe que o endereço do “Host E” seria o 192.168.1.100, com porta 2 (supostamente). Então, através do computador que executa o NAT (computador central), os dados do “Host A” são reencaminhados através do mapeamento para o “Host E”, pois o NAT sabe que a porta 2 do endereço IP “192.168.1.100” é o endereço “192.168.1.11” na sub-rede (SUN MICROSYSTEMS, 2010).

2.11 VPN E CLOUD

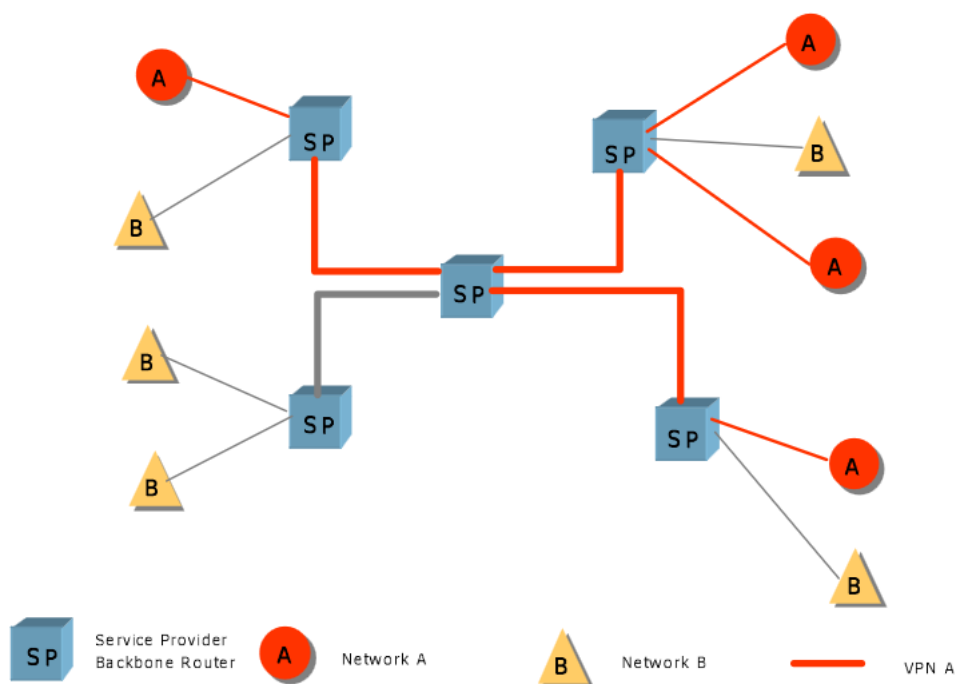
O conceito de comunicação entre computadores em uma mesma rede é algo que naturalmente constituí-se em um mesmo local/ambiente, conhecido por rede privada. Tratando-se de qualquer conexão a uma rede externa, espera-se uma comunicação por meio de redes públicas, sendo a Internet. Portanto, ao ter o desejo de estabelecer uma rede privada entre computadores localizados em diferentes localidades, a solução mais implementada é o *Virtual Private Network* (VPN). Como citado em (CISCO SYSTEMS, 2010), “uma VPN é uma rede privada construída numa infraestrutura pública, como a Internet global.”

O uso de uma rede privada virtualizada é benéfica em situações onde alguns programas ou arquivos estão disponíveis apenas em uma determinada rede, permitindo assim que o usuário possa acessá-los remotamente. Um VPN permite integrar o computador conectado por meio do sistema remoto à rede, efetivamente como se o computador estivesse localmente conectado, com os dados sendo criptografados na maioria dos serviços deste tipo, para garantir que as informações não sejam transmitidas pela Internet sem proteção. Uma topologia de VPN pode ser observada na Figura 25.

Assim como um VPN permite que uma rede de computadores locais possa ser expandida sem considerar limites de hardware local, o uso de um servidor na nuvem, ou “*cloud*”, pode permitir uma função similar, interligando dois computadores mediante um ponto comum na Internet. O uso de um servidor na Internet que possa ter um endereço IP fixo (incomum em ambientes residenciais), permite criar uma ponte fixa entre dois dispositivos que variem constantemente de endereço IP, permitindo assim uma conexão similar a um NAT, como visto na seção 2.10, onde os dispositivos nas pontas não sabem o endereço real do outro, mas sabem o endereço público de um ponto comum, que efetua a conexão entre os computadores.

Esta correlação se torna similar no caso de VPNs, pois um computador que está em um endereço IP público variável pode conecta-se a uma rede local, que define um endereço IP fixo dentro desta rede. Os computadores na rede local não identificam que o computador conectado via VPN está em outro local, identificando apenas o seu IP local. Estes mecanismos diferem entre si, mas possuem similaridades que permitem desenhar

Figura 25 – Topologia de VPN.



Fonte: Cisco Systems (1998).

correlações nas suas funcionalidades, porém cada serviço com seu objetivo diferente.

2.12 RASPBERRY PI

Single Board Computer (SBC), ou computador de placa única em tradução direta, são microcomputadores completos, executando em sua grande parte processadores da arquitetura ARM. Por serem extremamente customizáveis e de fácil acesso, criou-se todo um mercado ao redor deste gênero de dispositivos, na maioria considerados projetos “open-source”, ou abertos. Por possuírem CPU, RAM, vídeo e a disponibilidade de várias portas USB, porta Ethernet (conector RJ-45), módulos de expansão e especialmente baixo consumo energético, são amplamente implementados em projetos *hobistas*/caseiros. Também são empregados em demandas comerciais/industriais, onde seja necessário um hardware extremamente eficiente, mas capaz, sendo facilmente integrados nos mais variados ambientes. Um exemplo pode ser observado na Figura 26.

Sua facilidade em executar distribuições *Linux*, como o *Raspberry Pi OS* (que é uma vertente do bastante conhecido *Debian*), facilita o uso do hardware, podendo o próprio sistema operacional ser customizado para atender as demandas do projeto.

Figura 26 – Computador de placa única da linha Raspberry Pi.



Fonte: Raspberry Pi (2023).

3 PROPOSTA E ARQUITETURA DA SOLUÇÃO

Como já introduzido no Capítulo 2, o processo de parametrização de equipamentos aplicados na automação geralmente necessita a locomoção de uma pessoa diretamente a frente do painel elétrico para configurá-lo. Dependendo do modelo, pode ser necessário abrir a porta de um painel elétrico, o que seguindo as regras da NR-12 é considerado “processo de manutenção” e, então, os circuitos internos devem desenergizar toda a alimentação elétrica do painel. Com este processo, ocasiona-se uma parada de máquina, por exemplo, o que pode ser indesejado na maioria das situações que se é necessário configurar o seu equipamento. Com este propósito, os gerentes de produção estipulam períodos de manutenção procurando evitar perda de produção.

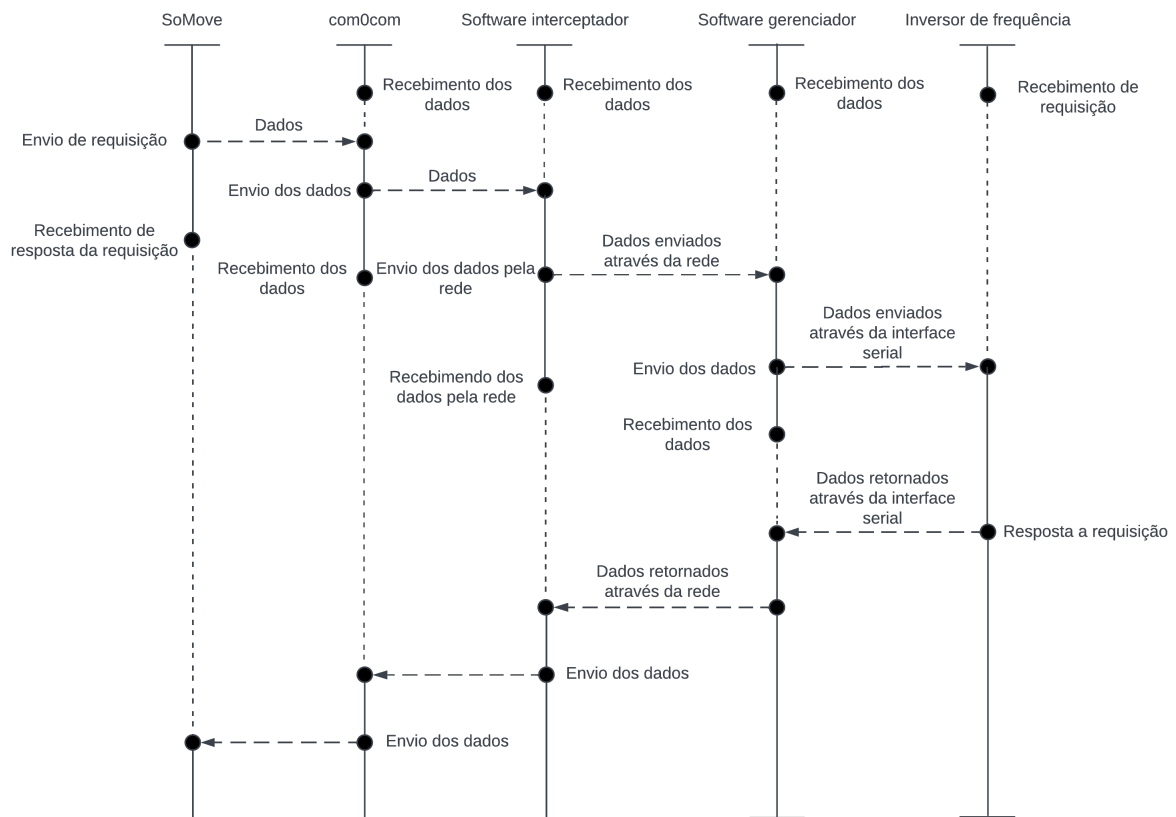
Portanto, visando evitar ao máximo a abertura de painéis elétricos ou então a locomoção exclusiva de pessoal na planta industrial, propõem-se um método que permita a operação mais contínua possível de inversores de frequência, servodrives ou soft-starters, abordados nas Seções 2.2, 2.4 e 2.3, respectivamente. A topologia desenvolvida é baseada na seguinte proposição:

1. O usuário possui um computador (chamado de “cliente”) onde executa o *software* de parametrização do seu equipamento;
2. Este *software* faz o envio dos seus dados para o equipamento por meio de um protocolo serial;
3. Um programa (chamado de “interceptor”) executado no computador cliente recebe estes dados e retransmite para outro computador (chamado de “servidor”);
4. O computador servidor recebe os dados e transmite via uma porta serial os dados para o equipamento;
5. Após o recebimento dos dados pelo equipamento, ele responde à requisição e envia os dados através da comunicação serial ao computador servidor;
6. O computador servidor retransmite estes dados para o computador cliente;
7. Com o recebimento dos dados pelo computador cliente, o programa interceptor reencaminha os dados via uma porta serial interna do computador cliente ao *software* de parametrização do usuário.

Com esta topologia definida, a lógica de operação é apresentada no fluxograma da Figura 27, com um exemplo de uma respectiva estrutura física do método proposto observável na 28.

No caso, o computador cliente onde será executado o *software* de parametrização não necessita ter nenhuma especificação especial, exceto executar uma versão mais atual do sistema operacional *Microsoft Windows* e possuir acesso a uma rede no qual o computador

Figura 27 – Diagrama tempo da proposta para a solução do problema.



Fonte: Do autor (2023).

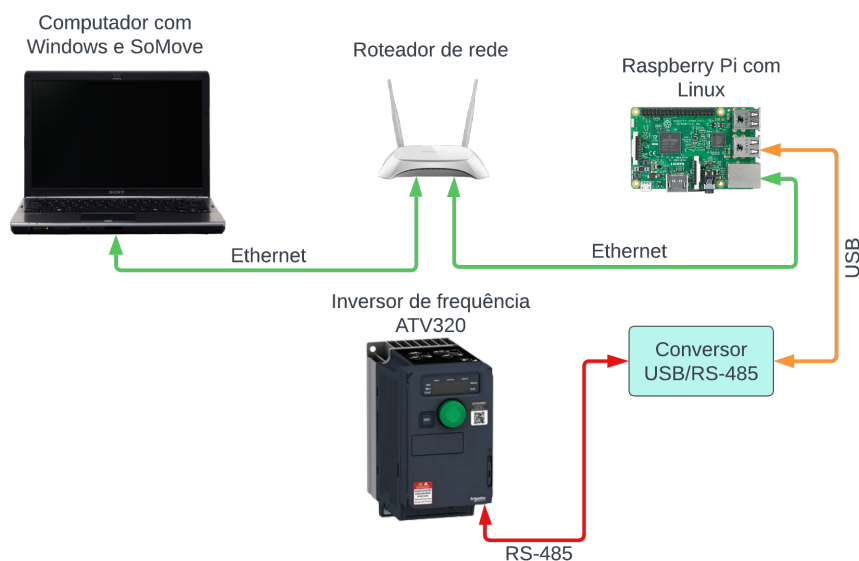
servidor está disponível. Já no computador servidor é interessante um hardware conhecido e específico, uma vez que é este que fará interface direta com a máquina via serial.

Considerando a necessidade da instalação de parte dos dispositivos necessários para a implementação da solução em painéis elétricos, é preciso considerar alguns pontos importantes, como:

- Capacidade de tolerar altos níveis de interferência eletromagnética ou na rede elétrica;
- Ocupar pouco espaço, podendo ser instalado até mesmo em trilhos de painéis;
- Ser capaz de operar de forma confiável, estável, com segurança e especialmente customizável;
- Permitir ser substituído por outro item similar no caso de problemas, sem a necessidade de modificações.

Com isto, uma escolha conveniente para esta aplicação, especialmente por permitir uma expansão futura do projeto, é a linha de computadores de placa única *Raspberry Pi*. Como já apresentado na Seção 2.12, são expansíveis, possuem uma multitude de portas

Figura 28 – Conexão dos equipamentos relacionados ao projeto.



Fonte: Do autor (2023).

e meios de comunicação e populares. Além destes pontos, dado o fato de serem capazes de executar sistemas operacionais baseados no *kernel* Linux, todo o *software* é confiável e amplamente documentado, facilitando a customização por parte do usuário, caso tenha interesse. Pelo poder de processamento que estes computadores implementam, também é possível executar funções ou tarefas em paralelo, podendo então serem adaptados para os mais variados usos. São então uma opção interessante de computador servidor que permanece instalado em um painel, conectando-se ao equipamento de automação a ser controlado mediante conversores seriais para portas USB.

3.1 CONFIGURAÇÃO DE PORTAS SERIAIS

Como o uso de portas seriais requer parâmetros específicos para a operação correta, e visando realizar a comunicação através do protocolo *Modbus RTU*, deve-se definir o modo de operação das portas seriais, tanto as portas virtuais do computador cliente (com Windows), como a porta serial física do computador servidor (sistema operacional baseado em Linux). Estas características são de extrema importância para a construção da mensagem enviada/recebida pelos dispositivos, sendo a base de toda a comunicação serial. É preciso determinar um conjunto de modos de operação que sejam compatíveis com os padrões de fábrica dos equipamentos.

Sabe-se que a Schneider Electric define os parâmetros observados na Figura 15 como o padrão para a comunicação serial com seus equipamentos. Portanto, sabe-se que para realizar a comunicação serial corretamente com o equipamento, todas as portas seriais

devem ser configuradas seguindo estas características:

- Paridade: par;
- Bits de parada: 1 bit;
- Taxa de transmissão (*baud rate*): 19200 baud;
- Modo de operação: RTU (8 bits de dados).

3.2 ESTRUTURA PARA ENVIO E RECEBIMENTO DE DADOS DO *SOFTWARE* VIA SOCKETS

Dada a necessidade de realizar a captura dos dados gerados pelo *software* de parametrização, assim como enviá-los para a máquina destino, torna-se necessário implementar um método que venha permitir manipular as informações seriais. Porém, os programas são proprietários e não são *open-source*. Além disto, utilizar programas que funcionem sem modificá-los permite sobretudo replicar o projeto com menores dificuldades. Portanto, existem algumas restrições impostas ao uso de portas seriais em sistemas operacionais Windows. São estes:

- As portas seriais são controladas apenas por um processo (programa);
- São gerenciadas pelo *kernel* do Windows ou por serviços executados ao nível do *kernel*;
- O sistema operacional disponibiliza um *buffer*, para ser possível comunicar por meio da porta serial;
- Não é possível realizar a injeção/leitura de informações em uma porta serial controlada, pois ela possui uma “entrada/saída” pré-definida. Um *software* lê e escreve em um lado da porta serial e sistema operacional faz o processo de envio/recebimento para o hardware.

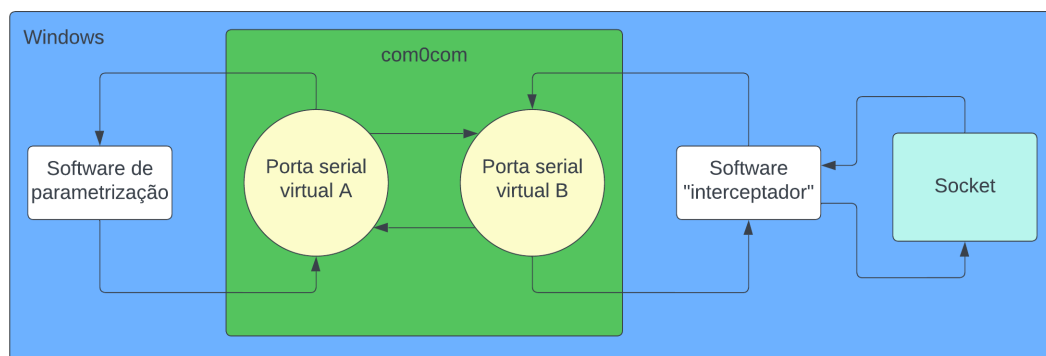
Com estas características, é necessário implementar outro método para permitir a manipulação do fluxo de dados. Neste caso, uma possível solução é o *software open-source* criado pelo projeto “*com0com*” (ver Seção 2.8). Com este *software*, é possível implementar a topologia observada na Figura 27. No computador onde o *software* de parametrização é utilizado, executa-se a lógica de conexões presente na Figura 29.

A partir do *socket* no computador cliente, os dados são enviados/recebidos a partir do computador servidor, como já abordado anteriormente. Neste dispositivo, a lógica é bastante similar, como pode ser observado na Figura 30.

3.3 INTEGRAÇÃO COM VPN

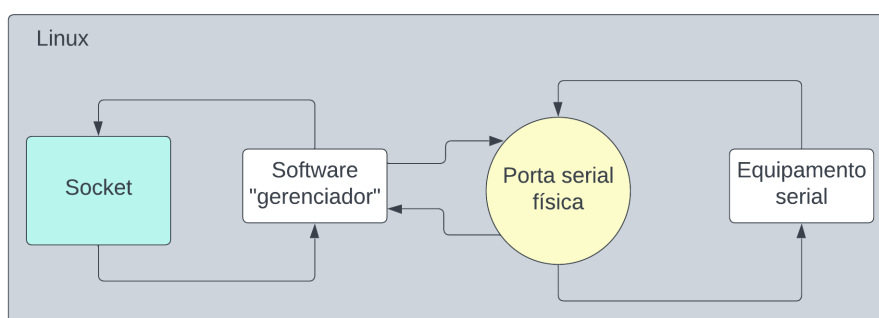
Através do conceito já apresentado na Seção 2.11, entende-se que por meio de um VPN a topologia apresentada na Figura 27 permanece idêntica, com apenas a rede

Figura 29 – Conexões virtuais dentro do Windows.



Fonte: Do autor (2023).

Figura 30 – Conexões controladas pelo computador com Linux.



Fonte: Do autor (2023).

Ethernet local substituída por um serviço que executa a tunelização de todos os pacotes da rede, centralizado em um servidor na Internet. Através deste VPN, pode-se conectar os dispositivos por meio de uma rede virtual privada, anônima e segura na Internet. Serviços como o “*LogMeIn Hamachi*” ou o gratuito “*ZeroTier*” permitem criar redes virtuais privadas entre dispositivos através da Internet.

Em uma comunicação através da rede que venha depender da infraestrutura de operadoras de Internet gera um agravante bastante importante no processo de comunicação serial, sendo a latência. Pelo fato de uma interface serial ser intrinsecamente direta e local, a latência tem a tendência de ser baixa. Tempos de processamento dos equipamentos são considerados em toda a interface serial, sendo geralmente padronizado como um “*time out*” (expiração/falha) da comunicação o tempo de 100ms sem respostas. Geralmente tenta-se efetuar a requisição novamente por algumas vezes até ser considerada “conexão perdida”. Portanto, a escolha de um serviço de VPN ou servidor na “*cloud*” que tenha bom desempenho (baixa latência) é desejado para garantir uma comunicação serial estável

e confiável, devido comportamento padrão de baixo tempo de *time out* dos equipamentos.

4 IMPLEMENTAÇÃO E RESULTADOS

Através da preparação do ambiente para realizar a comunicação entre os dispositivos virtualmente, empregou-se a topologia observada anteriormente na Figura 28. É importante salientar que a implementação desta topologia pode ser rearranjada para vários ambientes, como substituindo a conexão direta com um roteador por uma conexão por meio de VPN ou então da *cloud*, ambos abordados na Seção 2.11. Após a preparação do *hardware* e *software*, pode-se prosseguir para a execução prática do projeto.

4.1 DESENVOLVIMENTO DOS SOFTWARES INTERCEPTADOR E GERENCIADOR

Para permitir a comunicação entre dois computadores diferentes é preciso construir um programa que venha executar essa função. Considerando a necessidade da topologia incluir um computador com Windows e outro SBC executando Linux, fica clara a necessidade do suporte de ambas as plataformas, especialmente se tratando das suas variações na implementação da comunicação mediante sockets, da diferença entre um programa servidor e outro cliente e, sobretudo, as variações na comunicação através da interface serial.

Portanto, com estas variações de plataformas, estipulou-se o programa interceptador, que reside no computador cliente, e o programa gerenciador, executado no computador servidor. O *software* gerenciador é construído para disponibilizar a conexão de outro computador através da sua porta e endereço IP, dado que o computador servidor não deve variar seu endereço na rede. Portanto, sabendo qual é o seu endereço IP e a porta aberta no computador servidor, qualquer computador cliente que esteja na mesma rede conseguirá conectar. O *software* interceptador deverá ser construído de forma que irá estabelecer a conexão com o computador servidor (também chamado de servidor nessa estrutura de comunicação).

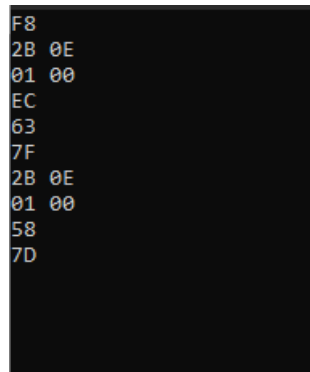
Seguindo as topologias observadas nas Figuras 27 e 28, o *software* interceptador é desenhado para realizar a leitura e escrita de uma das portas seriais virtuais do com0com, com a outra porta sendo controlada pelo SoMove. Por meio da leitura e escrita dos dados, realiza-se o envio/recebimento dos bytes através das portas dos computadores cliente e servidor, com o *software* gerenciador recebendo/enviando os bytes da mesma forma no computador servidor. Considerando o *software* gerenciador, ele efetua o mesmo tipo de comunicação através uma das portas de rede e, com isto, consegue receber as informações do SoMove ou então enviar os dados do inversor de frequência para o SoMove, em operação no computador cliente. Assim como o *software* interceptador realiza a leitura/escrita em uma das portas seriais virtuais do com0com, o *software* gerenciador efetua a leitura/escrita da porta serial física, disponibilizada pelo conversor USB/RS-485. Portanto, a construção é similar entre ambos os *softwares*, porém a maior diferença é apresentada na construção

da comunicação via sockets, onde o computador cliente conecta-se ao computador servidor, sem a possibilidade de efetuar esta comunicação a partir do computador servidor.

4.2 TESTES INICIAIS

Após a conexão dos equipamentos e a inicialização dos devidos programas nos devidos computadores, foi executada a funcionalidade de pesquisa de equipamento no SoMove, ocasionando os disparos de dados para reconhecimento de equipamento no barramento *Modbus*, o que pode ser visto na Figura 31. Como resultado, espera-se receber estes mesmos caracteres no computador executando Linux, o que de fato ocorre, conforme a Figura 32.

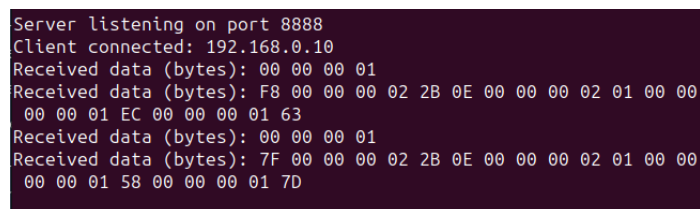
Figura 31 – Bytes de requisição e pesquisa do equipamento d SoMove.



```
F8
2B 0E
01 00
EC
63
7F
2B 0E
01 00
58
7D
```

Fonte: Do autor (2023).

Figura 32 – Bytes recebidos através da rede e socket do computador servidor.



```
Server listening on port 8888
Client connected: 192.168.0.10
Received data (bytes): 00 00 00 01
Received data (bytes): F8 00 00 00 02 2B 0E 00 00 00 02 01 00 00
00 00 01 EC 00 00 00 01 63
Received data (bytes): 00 00 00 01
Received data (bytes): 7F 00 00 00 02 2B 0E 00 00 00 02 01 00 00
00 00 01 58 00 00 00 01 7D
```

Fonte: Do autor (2023).

No conversor USB/RS-485, observou-se que há atividade apenas na sua porta “TX” (saída, indicada por LED de atividade), portanto está enviando dados ao inversor de frequência. Entretanto, não é gerada uma resposta por parte do equipamento, mantendo-se completamente “em silêncio” em todas as requisições do SoMove. Pode-se teorizar o que ocasiona este comportamento:

- As configurações do conversor serial não estão sendo aplicadas corretamente;

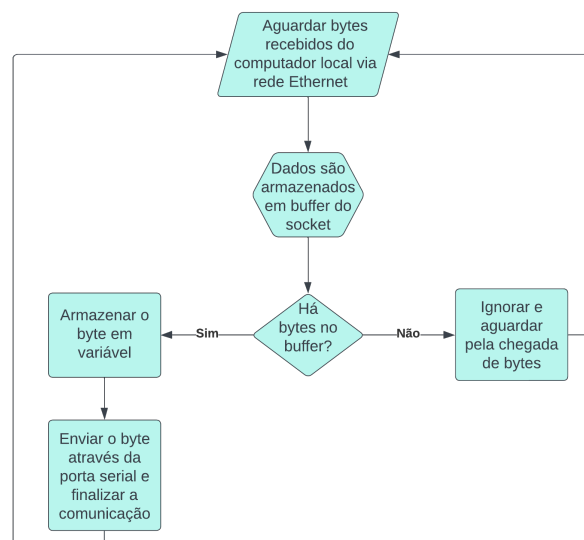
- Há muita interferência no ambiente dos testes, impedindo que um sinal com boa qualidade chegue à porta do inversor de frequência;
- Os dados enviados através da rede pelos *sockets* dos sistemas operacionais estão desformatando a mensagem;
- Os bytes são disparados fora de ordem no computador servidor;
- A construção da mensagem é incorreta.

Desta maneira, é preciso analisar a construção das informações que o computador servidor está recebendo através de sua porta e, caso necessário, aplicar algum meio para corrigir e reestruturar a mensagem *Modbus*, para garantir o envio correto dos *bytes* ao conversor serial.

4.3 ANÁLISE DA CONSTRUÇÃO DAS MENSAGENS SERIAIS

Por meio da interpretação do código em linguagem C desenvolvido para o recebimento e encaminhamento dos bytes no computador servidor, observa-se a representação da lógica operacional em forma de diagrama na Figura 33.

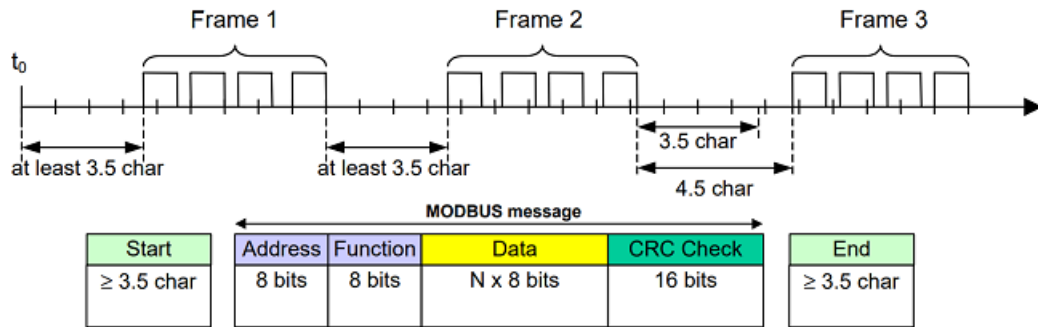
Figura 33 – Diagrama da lógica de recebimento de dados através da rede.



Fonte: Do autor (2023).

Portanto, com o recebimento de quaisquer dados através do *socket* do computador servidor, este byte é imediatamente disparado através da interface serial ao inversor. Analisando a construção dos bytes da Figura 31 e comparando com a construção da Figura 32, há uma clara distinção. O disparo imediato dos bytes sem considerar a construção de uma mensagem *Modbus* (Figura 34) acarreta problemas na comunicação com os equipamentos.

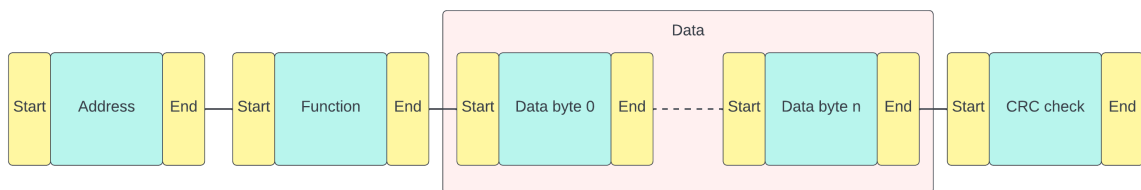
Figura 34 – Estrutura de uma mensagem do protocolo *Modbus*.



Fonte: Ozeki 10 (2020).

O problema gerado pelo disparo incorreto dos bytes impede que o inversor de frequência compreenda a requisição, dado que logo após cada byte disparado há um “*stop bit*”, gerado pelo próprio hardware da interface serial. Como toda a construção da mensagem *Modbus* depende de múltiplos bytes, para então ser enviado o *stop bit*, torna-se impossível o inversor de frequência decifrar a informação recebida, sendo esta a causa pela qual o mesmo não está respondendo aos bytes enviados. Uma comparação entre a Figura 34 e a Figura 35 permite visualizar o problema que ocorre, onde a mensagem completa deveria ser composta somente por um único “*start*” e um “*stop*”, o disparo contínuo está produzindo estes bits para cada byte enviado através da interface serial.

Figura 35 – Construção incorreta enviada através da interface serial.



Fonte: Do autor (2023).

Considerando esta característica e a dificuldade em determinar qual seria o byte que representa o final de uma mensagem *Modbus*, propõem-se que é mais viável criar um protocolo que seja capaz informar a quantidade de bytes que precisam ser enviados em uma única sequência, com o *stop bit* disparado somente após esta “*string*” de informações.

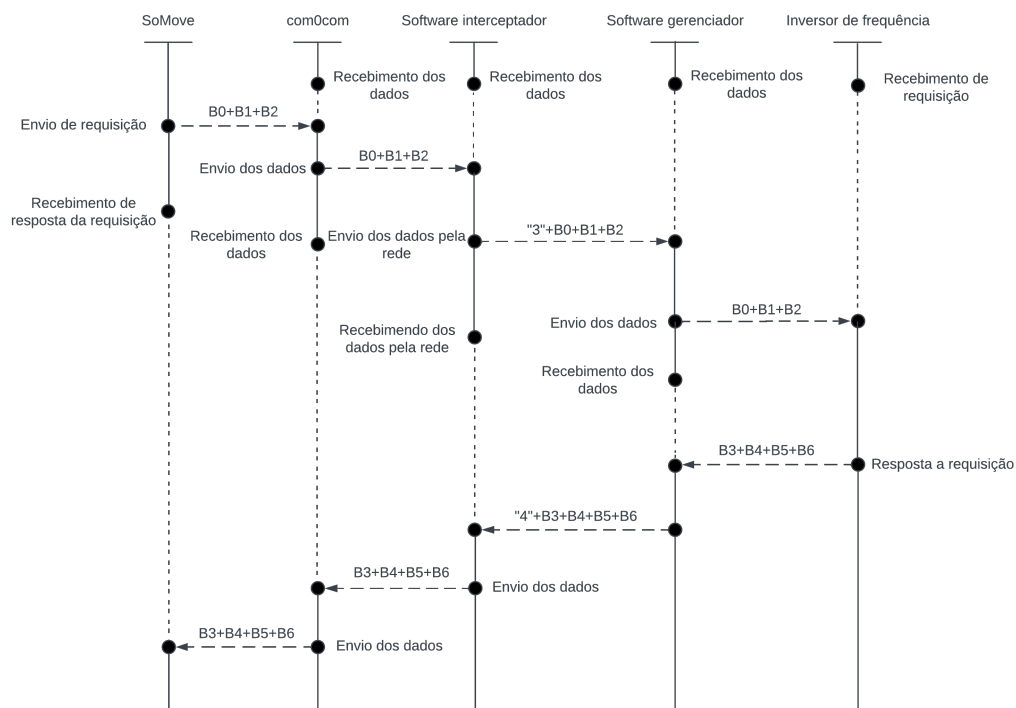
4.4 DESENVOLVIMENTO DE PROTOCOLO E APLICAÇÃO

Pela construção incorreta das mensagens *Modbus*, fica claro que o envio dos bytes através da rede por meio dos *sockets* dos computadores não gera uma saída idêntica à

entrada. Necessita-se então de um protocolo que venha possibilitar a reconstrução da mensagem original, com o “*start*” e um “*stop*” ocorrendo nos mesmos pontos que foram originalmente enviados. Um dos métodos mais simples e confiáveis para realizar essa estruturação é por meio da “contagem” dos bytes que foram originalmente enviados pelo SoMove, sabendo que esta é a mensagem *Modbus* completa. Desta forma, com o SoMove enviando sua mensagem constituída de uma determinada quantidade de bytes, realiza-se o envio destes bytes através da rede para o computador servidor, informando ao *software* a quantidade de bytes que precisam ser lidas do buffer do socket e disparando através da interface serial esta exata quantidade de bytes em sequência, com o *stop bit* ocorrendo apenas os bytes enviados. A lógica implementada é baseada na construção observada no diagrama da Figura 36.

Neste diagrama, suponha-se que o SoMove realiza o disparo de três bytes de dados, chamados de “B0+B1+B2”. Como são no total 3 bytes, o *software* interceptador adiciona o valor “3” logo antes da mensagem e faz o envio de 4 bytes no total. O processo contrário é o mesmo, com o inversor de frequência enviando 4 bytes chamados de “B3+B4+B5+B6”, então o *software* gerenciador realiza a contagem de 4 bytes, adiciona o valor “4” logo à frente da mensagem principal e faz o envio de 5 bytes no total.

Figura 36 – Diagrama da operação do protocolo proposto.



Fonte: Do autor (2023).

Com a construção do protocolo acima, modifica-se o código do programa disparador

no computador servidor para observar o comportamento entre a comunicação realizada nos dois sentidos, permitindo assim observar na Figura 37 parte do processo realizado durante a comunicação serial.

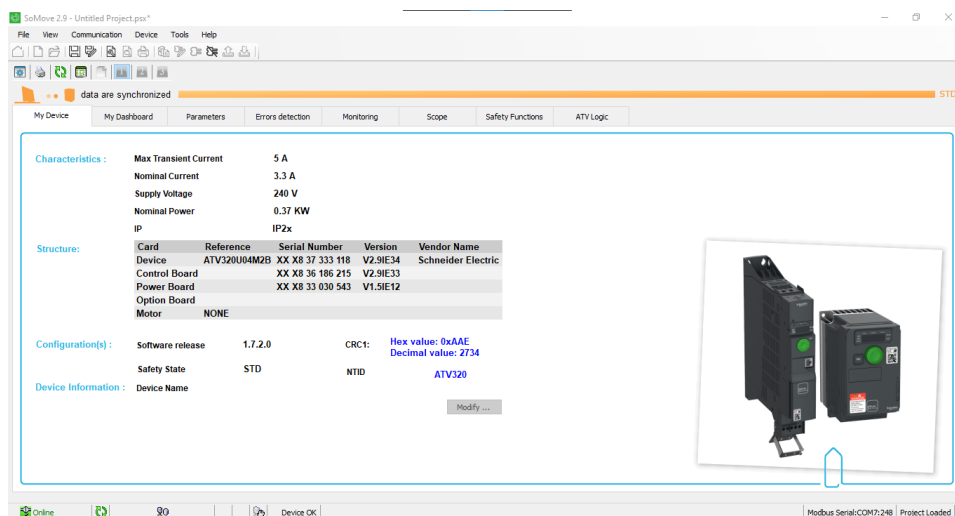
Figura 37 – Quantidade de bytes e a mensagem Modbus.

```
Received data size from socket: 1
Received data from socket (bytes): B5
Received data size from socket: 1
Received data from socket (bytes): 14
Received data size from serial: 8
Received data from serial port (bytes): F8 42 01 80 02 00 6C D8
Received data size from socket: 1
Received data from socket (bytes): F8
Received data size from socket: 2
Received data from socket (bytes): 42 01
Received data size from socket: 2
Received data from socket (bytes): 03 00
Received data size from socket: 1
Received data from socket (bytes): 00
Received data size from socket: 2
Received data from socket (bytes): 9C 50
Received data size from serial: 16
Received data from serial port (bytes): F8 42 01 01 03 13 FF 00 BF C0 20 00 13
00 02 02
Received data size from serial: 11
Received data from serial port (bytes): 90 00 03 08 02 07 D0 00 40 9A 99
Received data size from socket: 1
Received data from socket (bytes): F8
Received data size from socket: 2
Received data from socket (bytes): 42 01
Received data size from socket: 2
Received data from socket (bytes): 04 01
```

Fonte: Do autor (2023).

Por meio desta lógica, a comunicação serial ocorreu corretamente, com o SoMove podendo comunicar-se sem dificuldades com o inversor de frequência através da rede Ethernet.

Figura 38 – SoMove comunicando-se corretamente com o inversor de frequência.



Fonte: Do autor (2023).

4.5 VALIDAÇÃO DE OPERAÇÃO POR MEIO DE VPN

O processo de estabelecimento de uma rede virtual por meio de VPN é realizada utilizando o serviço gratuito “*ZeroTier One*”, compatível com sistemas operacionais Windows e Linux (dentre outros). Esta característica é de extrema importância nas condições onde se planeja o uso da comunicação através da Internet, não somente sobre uma rede local, dado que o serviço é executado similarmente a uma rede local física, possuindo sua própria porta Ethernet virtual. Pode-se até mesmo traçar um comparativo interessante entre o projeto desenvolvido neste trabalho e o conceito de operação de um VPN, onde ambos visam simular uma conexão direta e física entre dispositivos que se encontram em locais completamente diferentes (sem viabilidade de conexão física).

Por meio de uma página online, onde pode-se criar uma rede virtual, é possível obter um endereço da rede no qual o *software* do *ZeroTier One* executado no computador conecta-se para estabelecer a rede virtual. Com a conexão dos computadores nesta rede comum, obtém-se a estrutura de rede observada na Figura 39.

Figura 39 – Configuração da rede por meio de VPN.

Auth?	Address	Name/Description	Managed IPs	Last Seen	Version	Physical IP
<input checked="" type="checkbox"/>	5df9e72c5b 0e:8a:d7:52:6d:d2	Computador Local Onde o SoMove é executado	172.26.26.26 + 172.26.0.x	LESS THAN A MINUTE	1.10.6	181.223.27.157
<input checked="" type="checkbox"/>	fa23ae1afe 0e:ad:0d:1b:5b:77	Computador Remoto Conectado ao o Inversor de Freq	172.26.134.9 + 172.26.0.x	LESS THAN A MINUTE	1.10.6	UNKNOWN

Fonte: Do autor (2023).

Com a rede configurada, modifica-se o endereço IP do computador servidor para refletir o endereço gerado pelo *ZeroTier One*. Após isto, com a efetuação da requisição de pesquisa do inversor de frequência na interface serial no SoMove, o *software* pôde identificar o equipamento corretamente, realizando a comunicação sem dificuldades. O endereço IP do computador cliente no VPN, os comandos *Modbus* de pesquisa gerados pelo SoMove, assim como a resposta do inversor através de sua interface serial podem ser observados na Figura 40.

Essa validação de operação mediante VPN (ou algum serviço na “*cloud*”) confirma a teoria do comportamento afirmado anteriormente, onde pode-se implementar a topologia apresentada no projeto para permitir a conexão remota dos equipamentos com o *software* de parametrização.

4.6 LIMITAÇÕES DA TOPOLOGIA

Como fator limitante relacionado ao tempo empregado para realizar os estudos das tecnologias relacionadas a topologia apresentada, assim como a complexidade de testes

Figura 40 – Requisições e respostas através do VPN.

```
Server listening on port 8888
Client connected: 172.26.26.26
Received data size from socket: 1
Received data from socket (bytes): F8
Received data size from socket: 2
Received data from socket (bytes): 2B 0E
Received data size from socket: 2
Received data from socket (bytes): 01 00
Received data size from socket: 1
Received data from socket (bytes): EC
Received data size from socket: 1
Received data from socket (bytes): 63
Received data size from serial: 9
Received data from serial port (bytes): F8 2B 0E 01 02 00 00 03 00
Received data size from serial: 26
Received data from serial port (bytes): 12 53 63 68 6E 65 69 64 65 72 20 45 6C 65 63 74
72 69 63 01 0C 41 54 56 33 32
Received data size from serial: 15
Received data from serial port (bytes): 30 55 30 34 4D 32 42 02 04 30 32 30 39 F8 78
Received data size from socket: 1
Received data from socket (bytes): F8
Received data size from socket: 2
Received data from socket (bytes): 2B 0E
Received data size from socket: 2
Received data from socket (bytes): 02 03
Received data size from socket: 1
Received data from socket (bytes): AC
Received data size from socket: 1
Received data from socket (bytes): 92
Received data size from serial: 5
Received data from serial port (bytes): F8 AB 02 0E C0
```

Fonte: Do autor (2023).

e validações das propostas anteriores, o desenvolvimento de etapas mais avançadas que viabilizam a comercialização e distribuição da solução é impactada. Levando isto em conta, a implementação torna-se limitada, especialmente em relação aos seguintes pontos:

- A topologia suporta apenas a conexão entre dois computadores, assim como apenas gerenciar uma única porta serial;
- Não há uma interface amigável, necessitando a operação por meio de *terminal*;
- Uma mudança nas configurações de porta serial, porta ou endereço IP requer a recompilação do código em C;
- Apenas há suporte para o computador Windows (onde o *software* de parametrização é executado) e Linux (onde a porta serial física é empregada);
- Os programas não conseguem lidar com exceções/erros, encerrando assim que algo falha na sua execução;

Portanto, entende-se que é necessário um desenvolvimento mais aprofundado da solução proposta para facilitar a implementação em ambientes industriais.

5 CONCLUSÃO

Os objetivos previamente estabelecidos para viabilizar a comunicação à distância e aumentar a segurança na planta industrial foram atendidas, funcionando similarmente à ligação física. O esquema apresentado permite que, em determinadas situações onde surge a necessidade de uma indústria implementar uma topologia de operação à distância de seus equipamentos, é possível evitar acidentes. Um exemplo poderia ser aplicado em ambientes explosivos. A facilidade e simplicidade dos equipamentos envolvidos tornam interessantes, dado seu baixo custo operacional.

A comunicação através da interface serial RS-485 com um conversor USB permite o uso com uma quantidade variada de equipamentos e de marcas. Pelo fato dos programas desenvolvidos não dependerem exclusivamente de um tipo de interface serial ou apenas do protocolo *Modbus*, abre-se as possibilidades para implementá-lo nas situações que uma transmissão de dados seriais entre dois dispositivos, indo além do uso com equipamentos industriais. Outro ponto interessante, seria o desenvolvimento de uma interface amigável, aplicação de criptografia na transmissão dos dados e a compatibilidade com mais de uma porta serial no computador remoto são todas opções que viabilizam o uso e a migração do projeto para um serviço ofertado com viés comercial. A comunicação com portas seriais pode permitir até mesmo o uso de uma versão melhorada do projeto para a implementação de um sistema supervisor conectado diretamente aos equipamentos, remotamente. Através da expansão do número de portas seriais simultâneas, assim como múltiplas conexões de computadores clientes, torna-se muito interessante seu uso em ambientes onde há o desejo de aplicação de sistemas supervisórios, mas os equipamentos não implementam protocolos de comunicação expansíveis, como Ethernet. Esta última possibilidade tornando o uso de uma topologia como a proposta neste trabalho interessante, por reduzir custos de instalação com módulos e acessórios em equipamentos já instalados.

No contexto amplo, o desenvolvimento da base do projeto permite expandir a sua implementação para além do escopo limitado aqui abordado, permitindo estudos sobre novas necessidades e a solução de demandas além da indústria, através do uso de equipamentos de baixo custo. Uma possibilidade seria na implementação de ambientes comerciais, com impressoras seriais. Vários computadores conectando-se a uma mesma impressora, a qual possa ser antiga, mas de alto custo, tornando inviável sua substituição por algo mais moderno. Com a continuação do desenvolvimento da topologia apresentada, pode-se expandir a possibilidade de implementação para os mais variados ambientes.

REFERÊNCIAS

- COM0COM PROJECT. [S.l.], 2023. Disponível em: <https://com0com.sourceforge.net/>. Acesso em: 30 abr. 2023.
- FERGUSON, Paul; HUSTON, Geoff. **What is a VPN?** [S.l.], 2010. Disponível em: https://cpham.perso.univ-pau.fr/ENSEIGNEMENT/COMMUN/vpn_ferguson.pdf. Acesso em: 2 mai. 2023.
- GRITTER, D.; WANG, D.; HABETLER, T.G. **Soft starter inside delta motor modeling and its control.** [S.l.], 2000. 1137–1141 vol.2.
- KALE, Amit; KAMDI, Nikhil R.; KALE, Priya; YEOTIKAR, Ankita A. **A Review Paper on Variable Frequency Drive.** [S.l.], 2017. Disponível em: <https://www.irjet.net/archives/V4/i1/IRJET-V4I1229.pdf>. Acesso em: 15 abr. 2023.
- KESSLER, Gary C. **An Overview of TCP/IP Protocols and the Internet.** [S.l.], 1994-2019. Disponível em: <https://www.garykessler.net/library/tcpip.html>. Acesso em: 30 abr. 2023.
- MICHAEL A. BANKS. **The Modem Reference: The Complete Guide to PC Communications.** 4. ed. Medford, New Jersey, EUA: Information Today Inc., 1988. P. 19.
- MODBUS ORGANIZATION. **Modbus FAQ: About The Modbus Organization.** [S.l.], 2023. Disponível em: <https://modbus.org/faq.php>. Acesso em: 18 abr. 2023.
- PFAFF, Gerhard; WESCHTA, Alois; WICK, Albert F. Design and Experimental Results of a Brushless AC Servo Drive. **IEEE Transactions on Industry Applications**, IA-20, n. 4, p. 814–821, 1984.
- RÜSSMANN, Michael; LORENZ, Markus; GERBERT, Philipp; WALDNER, Manuela; JUSTUS, Jan; ENGEL, Pascal; HARNISCH, Michael. **Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries.** [S.l.], 2015. Disponível em: https://www.bcg.com/publications/2015/engineered_products_project_business_industry_4_future_productivity_growth_manufacturing_industries#chapter1. Acesso em: 12 abr. 2023.
- SANTOS, Lourival Santana; ARAÚJO, Ruy Belém de. **A Revolução Industrial.** [S.l.], 2016. Disponível em: <https://cesad.ufs.br/ORBI/public/uploadCatalogo/>

10264518102016Historia_economica_geral_e_do_brasil_Aula_03.pdf. Acesso em: 10 abr. 2023.

SOUZAO, Vitor Amadeu. **O Protocolo Modbus**. [S.l.], 2008. Disponível em: <https://www.cerne-tec.com.br/news/Modbus.pdf>. Acesso em: 18 abr. 2023.

SUN MICROSYSTEMS. **Sun Management Center 3.6 Installation and Configuration Guide - NAT Examples**. [S.l.], 2010. Disponível em: <https://docs.oracle.com/cd/E19062-01/sun.mgmt.ctr36/817-7958/nat-20/index.html>. Acesso em: 2 mai. 2023.