PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE ENGENHARIA / ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Trabalho Final de Curso II

FABIANO PEREIRA LIMA JUNIOR
GUILHERME ALMEIDA CAVALCANTE
JOSÉ AUGUSTO DA SILVA FONSECA
RODRIGO MAIA FLORES ESSADO

PROPOSTA DE UMA REDE INDUSTRIAL SUPERVISIONADA COM O MODBUS RS-485 UTILIZANDO O RASPBERRY PI

Trabalho Final de Curso como parte dos requisitos para obtenção do título de bacharel em Engenharia de Controle e Automação apresentado à Pontifícia Universidade Católica de Goiás.

BANCA EXAMINADORA:

Prof. Bruno Quirino – Orientador. PUC Goiás.

Prof. Dr. Antônio Marcos de Melo Medeiros – Banca. PUC Goiás.

Prof. Dr. Luis Fernando Pagotti – Banca. PUC Goiás

Goiânia, 08 de Dezembro de 2021

PROPOSTA DE UMA REDE INDUSTRIAL SUPERVISIONADA COM O MODBUS RS-485 UTILIZANDO O RASPBERRY PI

Guilherme Almeida Cavalcante, Fabiano Pereira Lima Junior, José Augusto da Silva Fonseca e Rodrigo Maia Flores Essado - Engenharia de Controle e Automação, PUC Goiás.

***Resumo* — As Redes Industriais já se tornaram algo comum e em constante evolução no campo industrial. Com intuito de tornar esse conceito mais visível em um meio acadêmico, além de demonstrar as diversas aplicações possíveis com o Raspberry Pi, foi validado a comunicação utilizando do meio físico RS-485, abrindo assim uma gama de opções muito extensa para outras aplicações.**

***Palavras-chaves* — Raspberry Pi, Modbus, Redes Industriais.**

Abstract – The Industrial Networks have already become something common and constantly evolving in the industrial field. In order to make these concept noticeable in an academy field, going further to demonstrate the diverse and possible applications with the Raspberry Pi, has been validated a communication, using the physical environment RS-485, therefore opening a lot of options extenting to other applications.

Keywords* — *Raspberry Pi, Modbus, Industrail Networks.

I. INTRODUÇÃO

Com todas as inovações e avanços tecnológicos no âmbito industrial, foi necessário uma otimização e aperfeiçoamento de métodos de produção, almejando assim uma comunicação e supervisão em todos os setores de forma exata e clara, além de acessível.

Para isso, as redes industriais se concretizaram como ótima ferramenta para alcançar esses objetivos, quando implementadas de forma correta, o que requer um cuidado e um estudo muito minucioso para tal aplicação.

Redes industriais seria um método de automação industrial, que se estrutura em protocolos de comunicação, usados para supervisionar e gerenciar processos, baseando-se em trocas de informações entre sensores, computadores e máquinas de uma forma mais ágil e eficaz [1].

Para a implementação dessas redes ser possível, é necessário construir uma infraestrutura adequada, implementar um sistema de segurança e realizar conexões utilizando a Internet das Coisas (IoT), auxiliados de protocolos de comunicação como o Modbus, que será apresentado no trabalho [2].

Os conceitos utilizados ao longo do trabalho, possibilitam tornar os processos mais inteligentes e independentes, proporcionando uma ótima eficiência dos processos de produção. Tendo em vista a conjuntura delineada anteriormente, o objetivo do trabalho tem o intuito em validar a comunicação serial utilizando o Raspberry Pi, com o protocolo Modbus e meio físico RS-485 como base.

II. FUNDAMENTAÇÃO TEÓRICA

A – REDES INDUSTRIAIS

As redes industriais são protocolos de comunicação que utilizam sistemas supervisórios e de gerência, para poder automatizar as redes industriais interligando os processos e tendo mais controle entre eles.

Com os avanços tecnológicos do Brasil e do mundo são recursos fundamentais para que as indústrias se mantenha ativa no mercado. Visto que sempre maximizam a produtividade, lucratividade, e redução de gastos da indústria. E as redes industriais são uma das tecnologias capazes de fornecer esses proveitos.

As redes industriais funcionam por meio de uma rápida troca de informações entre sensores, computadores, máquinas, atuadores, entre outros equipamentos. O diferencial da tecnologia consiste no fato de que o procedimento será feito de forma automática, dedicada ao contexto e ao ambiente industrial.

Tendo as divisões em três grandes grupos: *sensorbus*, *devicebus* e *fieldbus* e também existe a Ethernet, outra modalidade de rede que deu origem a outro grupo de redes industriais.

Por fim os sistemas de gerência ou rede informação é o mais alto desses processos, este sistema tem a função de gerenciar os sistemas supervisórios e de comunicação além de trazer ferramentas para melhor controle da indústria com o planejamento de produção e administração de recursos, utilizando dispositivos como *Mainframe* e *Workstation*.

Este conceito surgiu a partir do século XX juntamente com a 3° Revolução Industrial e foi sendo aprimorado com a Indústria 4.0, e com isso trouxe um melhor controle do processo de fabricação dos produtos, além de minimizar os custos e o aumento de operacionalidade. De acordo com [3], a utilização de redes e protocolos digitais prevê um significativo avanço nas seguintes áreas:

- Custos de instalação, operação e manutenção;
- Facilidade de diagnóstico da rede;
- Procedimentos de manutenção com gerenciamento de ativos;
- Fácil expansão e *upgrades*;
- Informação de controle e qualidade;
- Determinismo (permite determinar com precisão o tempo necessário para a transferência de informações entre os integrantes da rede);

- Baixos tempos de ciclos;
- Várias topologias e padrões abertos;
- Redundância em diversos níveis;
- Menor variabilidade nas medições com a melhoria das exatidões e medições multivariáveis.

B – SISTEMAS EMBARCADOS

Sistema embarcado (ou sistema embutido), é um sistema microprocessado em que um composto é totalmente dedicado ao sistema que deve controlar. É um sistema completo e independente, desenvolvido para realizar tarefas específicas epré-estabelecidas. Cujo o programa não pode ser acessado pelo usuário final, ele pode interagir através de interfaces, como *displays*, teclados e entre outros periféricos [4]. Dando alguns exemplos de sistemas embarcado na Tabela 1.

Tabela 1: Sistemas Embarcados.

Mercado	Sistemas
Automobilístico	Ignição, controle de motor e sistema de freios.
Eletroeletrônicos	Televisões, DVD's, videogames, refrigeradores, telefones, *smartphones*, câmeras, entre outros.
Medicina	Bomba de infusão, máquinas de diálise, monitores cardíacos, entre outros.
Indústria	Sistemas robóticos.
Redes	Roteadores, *hubs* e *gateways*
Automação de escritório	Impressora, fax, monitores, *scanners*, entre outros

Fonte – NOERGAARD, 2013 [5].

Para que um sistema embarcado funcione com as definições pressupostas, um sistema embarcado é constituído por alguns componentes necessários e atribuições básicas [6].

- Processador: Depende da área de atuação que vai estar situado para levar em consideração a velocidade de processamento. A complexidade e a quantidade de tarefas devem ser levadas em consideração
- Memória: Componente que promove o armazenamento das informações para a inicialização e execução das tarefas contidas no sistema
- Algoritmos: Códigos e comandos que permitem o *software* exercer as tarefas. O algoritmo processa matematicamente as informações externas de forma a interpretá-las e gerar sinais de controle, usados pelo sistema.
- *Software*: Aspecto que define o que o sistema faz, assim como a performance
- Periféricos: A comunicação de um sistema embarcado é dada através de dispositivos de fora e o que permite isto são os periféricos. Podem ser, sensores em geral, saídas de áudio e vídeo, LEDs.

C – RASPBERRY PI

O conceito de sistemas embarcados será utilizado para esta aplicação com o microprocessador Raspberry modelo 3B. O Raspberry Pi será utilizado como unidade central de processamento (CPU) para o desenvolvimento do trabalho.

O Raspberry Pi é um computador de placa única desenvolvido para diminuir os custos da computação tornando a programação acessível para um público mais abrangente. Com tamanho sofisticado, com dimensões de um cartão de crédito o aparelho possui o suficiente para simular um computador de forma mais compacta, mas não deixa de ser um microprocessador potente que é o necessário para atender as especificações do projeto. O modelo usado no projeto é o 3B, dispondo das seguintes especificações técnicas Figura 1.

Fig. 1. Raspberry PI modelo 3B [7].

- Quad. Core 1.2GHz Broadcom BCM2837, 64 bits, ARMv8 CPU;
- BCM43438 *Wireless* LAN & *Bluetooth* Low Energy (BLE);
- *Fast Ethernet*;
- 1GB RAM;
- GPIO de 40 pinos;
- 4 portas USB 2.0;
- DSI & CSI;
- Micro USB 5V, 1A;
- Áudio & Vídeo Jack;
- HDMI *Full Size*.

A escolha do Raspberry Pi se dá por diversas características que favorecem a execução do trabalho em si, como maior conectividade, facilitando o vínculo com outros dispositivos de controle ou supervisórios via Wi-Fi ou cabo UTP (*Unshielded Twisted Pair*) e maior poder de processamento dentre às demais.

Falando sobre o *software* disponibilizado pela empresa Raspberry, o *Raspbian OS*, é uma distribuição Linux criada para rodar nos microprocessadores Raspberry Pi. O *software* é ideal para um público que têm interesse em aprender mais sobre eletrônica, desenvolvimento de *software*, redes e tecnologias em geral [8].

O *Raspbian OS* está longe de um rival para o *Windows* 10 ou *macOS*, mas cumpre o que promete, além de conter o navegador de internet *Chromium*, *Transmission* para baixar torrentes, além de contar com as ferramentas de desenvolvimento como *Python* e Java.

D – LINGUAGEM PYTHON

Criada no início da década de 90 pelo programador ematemático holandês Guido Van Rossum. Vindo como uma facilitadora para a programação a redução de caracteres "esotéricos". A linguagem *Python* é uma linguagem de código aberto que permite modificações e estudos sobre o código, além de contar com uma grande biblioteca padrão que possui vários módulos para diversas necessidades que possa ter durante o código [4].

O *Python* pode ser citado como uma linguagem portátil, ou seja, que pode ser executada em diferentes tipos de computadores sem sofrer muitas alterações, diferentemente de linguagens de baixo nível que devem ser reescritas quando executados em sistemas diferentes. Algumas características que tornam o *Python* uma linguagem de alto nível:

- Estrutura construída de forma consistente (classe, módulos e pacotes);
- Estruturas de controle usuais com *if*, *if-else*, *while* e uma coleção abrangente para a estrutura *for*;
- Extensão para as linguagens C ou C++;
- Linguagem de código aberta que permite modificações e estudos sobre o código;
- Multiplataforma, roda em qualquer sistema que possua o interpretador. Exemplos como: Unix (Linux, FreeBSD, Solaris, MacOs X, etc), *Windows*, .NET. Caso não venha a ter algum interpretador, basta que tenha um compilador C disponível e gerar o *Python* a partir do código fonte;
- Há "dialetos" disponíveis como *Jython* (forma de escrever *Python* em Java), *CPython* (implementação principal de *Python* escrita em linguagem C), a *IronPytho*n (linguagem *Python* escrita em C# para plataforma .NET e Mono) e entre outros.

Com a grande comunidade ativa que a linguagem *Python* tem, várias bibliotecas com diferentes funcionalidades tornam o uso da linguagem ainda mais ascecivel. Essas bibliotecas são *minimalmodbus* e *pymodbus*, fundamentais para o projetoem si.

Estas dispõem de estruturas lógicas para consolidar a utilização de interface gráfica de usuário, promover a comunicação via protocolo Modbus, além de oferecer todos os recursos necessários para transmitir, decodificar e receber dados seriais através de dispositivos que utilizam do protocolo.

E – ELIPSE E3 STUDIO

O Elipse E3 é um software SCADA/HMI com alta performance de comunicação e conectividade com mais de 400 equipamentos. É uma ferramenta para monitoramento e controle de processos, oferecendo constante evolução para diversos tipos de aplicações, desde simples HMI até projetos mais complexos.

Permite visualizar e operar o sistema via *tablets* e *smartphones*, gerenciando indicadores, alarmes e manipulando grandes massas de dados.

Por ter a possibilidade de utilização de diversos *drivers* de comunicação, ele pode realizar conexão com a maioria dosequipamentos do mercado. Por exemplo: A utilização do *driver* MBPROT para comunicação com o protocolo Modbus ou o *driver* ABCIP para comunicação com o *Softlogix* da *Rockwell*.

Através do uso de bibliotecas, o Elipse E3 possibilita aos desenvolvedores a padronização das aplicações e como consequência uma redução no desenvolvimento e na manutenção dos projetos.

Principais características [9]:

- Multiusuários e multiprojetos: Permite editar e executar diversos projetos simultaneamente;
- Redundância nativa com sincronismo de dados históricos e alarmes;
- Bibliotecas de objetos gráficos e estruturas de dados reutilizáveis;
- Editor de telas completo e poderoso;
- Conexão nativa transparente entre servidores remotos;
- Segurança e compactação na transmissão de dados;
- Fácil gerenciamento da aplicação;
- Grande flexibilidade na gestão de alarmes e eventos;
- Poderosa ferramenta de *scripts*;
- Acesso nativo a bancos de dados comerciais;
- Ferramenta de *logs*, consultas e relatórios integrados;
- Alta segurança e rastreabilidade de acordo com a norma FDA CFR 21 Part 11;
- OPC *Classic* e UA;
- Integração com o *Windows Active Directory*.

Componentes:

- E3 *Studio*: é a plataforma universal de desenvolvimento. É onde são criadas as telas de operação, criação de *scripts*, configuração de comunicação. Possibilita que um mesmo arquivo seja editado por diversos usuários;
- E3 *Server*: É onde são gerenciados os principais processos do sistema;
- E3 *Viewer*: É a interface de operação com o usuário. Permite visualizar e operar em qualquer computador a aplicação que está no servidor. Pode ser executada via *browser*.

F – PROTOCOLO MODBUS

A rede Modbus foi criada em 1979 com a finalidade de realizar a comunicação entre dispositivos mestre-escravo/ cliente-servidor (Figura 2).

Por se tratar de um protocolo de comunicação, um conjunto de regras pré-estabelecido rege a troca de dados entre os sistemas computacionais. Esse protocolo é um dos mais utilizados em automação industrial devido a fácil implementação, podendo ser utilizado em diversos padrões de meio físico que se diferem quanto a velocidade de comunicação, número máximo de dispositivose comprimento máximo de rede.

Esses padrões de meio físico podem ser:

- RS-232 (*Recommended Standart-232*) ou EIA-232 (*Electronic Industries Alliance-232*) permite apenas 2 dispositivos na rede, pois é utilizado apenas em comunicações do tipo ponto a ponto (Mestre-escravo / cliente-servidor). O RS-232 é um cabo que oferece comunicação com topologia do tipo duplex, ou seja, o dispositivo é capaz de receber e transmitir dados ao mesmo tempo. A velocidade máxima padrão está em torno de 115Kbps, a distância máxima entre os dispositivos está em torno de 30m.
- RS-485 (*Recommended Standart-485*) ou EIA-485 (*Electronic Industries Alliance-485*) é um dos padrões mais utilizados pelo protocolo Modbus. Esse padrão permite trabalhar com taxas de comunicação que podem chegar a 12Mbps e em alguns casos até 50Mbps, notando que quanto maior o comprimento da rede menor será a velocidade de comunicação, sendo que a distância máxima da rede está em torno de 1200m, e o número máximo de dispositivos no barramento da rede é de 32. O RS-485 é um cabo que oferece topologia do tipo Half-duplex, ou seja, é possível a transmissão de dados apenas um após o outro [10].
- O padrão Ethernet no protocolo Modbus possui algumas variações, podendo chegar a 100Mbps ou até 10Gbps. A distância máxima pode variar de 100m até próximo de 200m dependendo do tipo de cabo utilizado e das condições de instalação do mesmo. Em alguns casos é possível utilizar redes em fibra óptica, fato que permite alcançar distâncias maiores e melhores taxas de comunicação, bem como utilizar comunicação *wireless*. Nesse caso, ao utilizar o meio físico Ethernet, o protocolo Modbus opera com o mecanismo de controle de acesso CSMA-24 CD, que é próprio da rede Ethernet, com mensagens no modelo cliente-servidor [10].

Por ser um protocolo que se baseia em um relacionamento mestre-escravo (requisição-resposta), a comunicação sempre ocorre em pares, em que um dispositivo inicia a requisição de dados e então aguarda a resposta. Em geral, o mestre é representado por uma IHM (Interface Homem-Máquina) ou um sistema SCADA (*Supervisory Control and Data Acquisition*) e o escravo é um CLP (Controlador Lógico Programável), sensor ou um CPA (Controlador Programável para Automação).

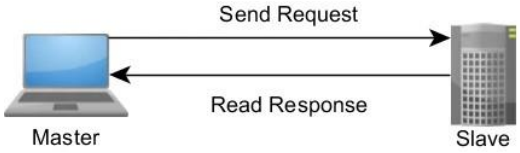

Fig. 2. Comunicação mestre-escravo/cliente-servidor [11].

Para que a comunicação entre os dispositivos ocorra, é necessário que o Escravo possua um endereço próprio, possibilitando assim que o Mestre tenha acesso e possa realizar ações de leitura ou escrita dos dados fornecidos pelo Escravo. Os dados transmitidos podem ser discretos ou numéricos, ou seja, é possível enviar valores numéricos como temperatura e pressão ou enviar um bit para ligar e desligar um motor.

Cada mensagem possui código e endereçamento próprio e o quadro de mensagens no protocolo Modbus é mais bem explicado pela Figura 3 abaixo:

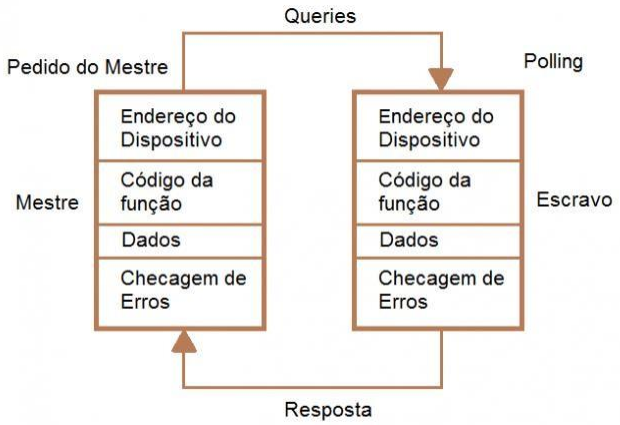

Fig. 3. Quadro de mensagens [12].

- Endereços: O protocolo Modbus possui 256 endereços, entre eles, os endereços presentes entre 1 e 247 representam os endereços disponíveis para os escravos. Entre 248 e 255 estão os endereços reservados. O endereço 0 é um endereço de broadcast, ou seja, permite direcionar a informação a todos os escravos.
- Código da função: É onde é especificado pelo dispositivo mestre o tipo de serviço ou

função solicitada ao escravo (leitura, escrita, etc).

O protocolo Modbus possui nas especificações dois modos de transmissão, sendo eles o RTU (*Remote Terminal Unit*) e o ASCII (*American Standard Code for Information Interchange*). Esses modos de transmissão definem a forma de como serão transmitidas as informações e como ela será empacotada e descompactada. Na comunicação em rede Modbus utilizando o ASCII, cada byte presente na mensagem é enviado como 2 caracteres ASCII, que podem ser legíveis pela tabela ASCII. Já pelo método RTU, cada *byte* presente na mensagem é enviado como dois caracteres hexadecimais de quatro *bits* cada. O modo de transmissão RTU tem como principal vantagem sobre o modo de transmissão ASCII a maior densidade de caracteres, que permite um melhor processamento de dados.

O Modbus TCP é uma implementação do protocolo Modbus, que é baseado em TCP/IP. Para realização da comunicação, é adicionado ao quadro Modbus um cabeçalho chamado MBAP (Modbus *Application Protocol*).

G – CONVERSOR I485 NOVUS

Com a intuito de fazer a comunicação entre o Raspberry eo CLP, é preciso de um conversor para comunicar ambas as interfaces. De acordo com o manual do Novus.

> O conversor **USB-i485** é a solução rápida e segura para a interface entre o PC e barramentos de comunicação industrial RS485 ou RS422. Ao ligar o **USB-i485** à porta USB de um PC, ele é automaticamente detectado e instalado como uma porta COM nativa, compatível com qualquer aplicativo existente de comunicação serial. Múltiplos conversores podem ser instalados utilizando hubs USB, permitindo a fácil configuração de um sistema multi-serial, sem qualquer preocupação com configurações de IRQ ou DMA [13].

O modelo usado no projeto é o Novus I485, dispondo das seguintes especificações técnicas, conforme a Figura 4.

Fig 4. Conversor I485 Novus.

As especificações deste equipamento estão apresentas a seguir:

- Interface USB (V1.1, V2.0 e V3.0) Plug and Play;
- Driver de porta serial virtual para sistemas operacionais: Windows 10/7/Vista/XP/2008Server/2003Server/98/ ME/2000/CE, MAC e Linux 2.4.20 ou superior. Opções de 64 bits para sistemas operacionais mais recentes;
- Interfaces de campo: RS485 Half Duplex (dois barramentos), RS485 Full Duplex ou RS422;
- Número máximo de dispositivos na rede RS485:
  Half Duplex: 2 x 32 dispositivos
  Full Duplex: 32 dispositivos;
- Isolação: 1500Vcc entre a interface USB e a interface RS485/RS422;
- Proteção no barramento RS485/422: ±60 Vcc, 15 kV ESD;
- Conexão RS485/422: Conector para fios até 1,5 mm2 (16 AWG);
- Controle de fluxo automático para RS485 Half Duplex;
- Alimentação: Pelo barramento USB. Consumo <100 mA.

III. DESENVOLVIMENTO TEÓRICO

A aplicação tem como intuito uma comunicação entre dispositivos, de forma serial, através do meio físico RS-485, simulando assim uma rede industrial. Para tal comunicação é usado toda a teoria exposta anteriormente, desde a parte da montagem e programação até o entendimento dos resultados encontrados.

Para o melhor entendimento de como elaborar a validação dessa comunicação foi separado três tópicos explicando e exemplificando como foi realizada a montagem, programação e os resultados encontrados.

A - MONTAGEM

Para a montagem 01, O conversor I485 é ligado no Raspberry Pi através da porta USB, então o positivo, o negativo e o GND são conectados ao CLP através das portas D0, DŌ e GND. A Figura 5 mostra a conexão entre os dispositivos.

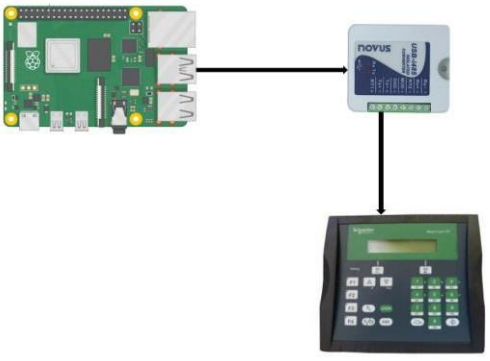

Fig 5. Montagem 01.

Para ligar os dois CLP's na montagem 02, é feita a mesma configuração utilizada na montagem 01, com a adição de uma ligação em série nas portas D0, DŌ e GND de um CLP para o outro, como pode ser visto na Figura 6.

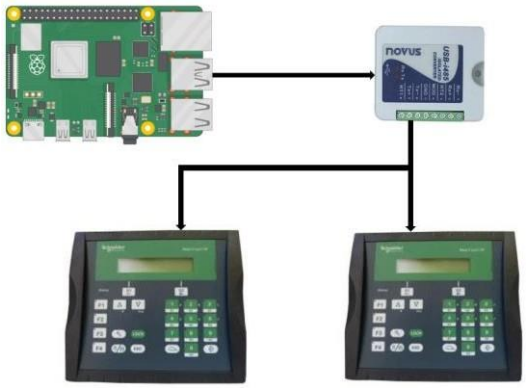

Fig 6. Montagem 02.

Para entender de forma mais clara, seguindo uma ordem cronológica, de todos os processos e configurações necessárias para tornar possível esta comunicação, o diagrama de blocos auxilia para esta compreensão, conforme a Figura 7.

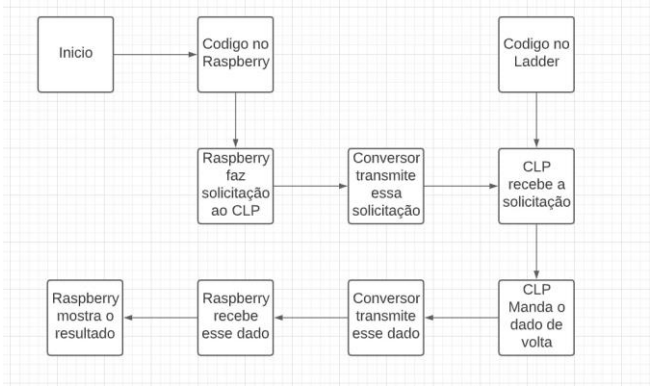

Fig 7. Diagrama de blocos.

B – PROGRAMAÇÃO

Para as aplicações dos projetos, foi utilizada a biblioteca MinimalModbus, um módulo Python de fácil uso para realização de comunicação Mestre-Escravo/Cliente-Servidor. Esse software dá suporte para utilização do Modbus em RTU e em ASCII e pode ser destinada para diversos sistemas operacionais, entre eles o Linux (*Raspbian*).

Para realizar a instalação dessa biblioteca, é utilizado o seguinte código no *Prompt pi* do Raspberry: *sudo pip3 install -U minimalmodbus*

Após a instalação, foram utilizados dois códigos no *Prompt pi* do Raspberry para identificação e realização de atualizações disponíveis e atrasadas para que não haja erros nas aplicações.

sudo apt update - Pega a lista de atualizações disponíveise atrasadas;

sudo apt upgrade - Faz as atualizações que estão disponíveis ou atrasadas.

De acordo com a documentação do conversor I485 NOVUS, a partir da versão 2.4.20, o Kernel já inclui o *driver* necessário para o funcionamento do conversor, então foi utilizado o comando a seguir para identificação da versão no *Prompt pi* do Raspberry: *uname*

Após a verificação da versão e constatação de que o Kernel estava atualizado, foi utilizado o comando *lsmod* para verificar quais módulos estavam carregados no Kernel e foram então identificados os drivers do conversor.

Com todos esses procedimentos realizados, se torna possível a programação e a comunicação entre o Raspberry (com o uso do conversor) e o CLP (Atos Expert BF 2850.30). Para validação da comunicação Modbus RS-485, é necessário a programação em *ladder*, de um código que habilita a variável de sistema %SX143 que é responsável por habilitar o protocolo Modbus na porta RS-485. Após isso, é necessária a programação em *ladder* para se realizar procedimentos de escrita (linha 002) e leitura (linha 003). [9]

Nas linhas 002 e 003 foram utilizadas as variáveis de saída e entrada. Essas variáveis foram declaradas e *linkadas* aos endereços de memória %I1.1.1, sendo referente a entrada e o %Q1.1.1 à saída. A Figura 8 e 9 apresenta o código em *ladder* e o campo de endereçamento das variáveis.

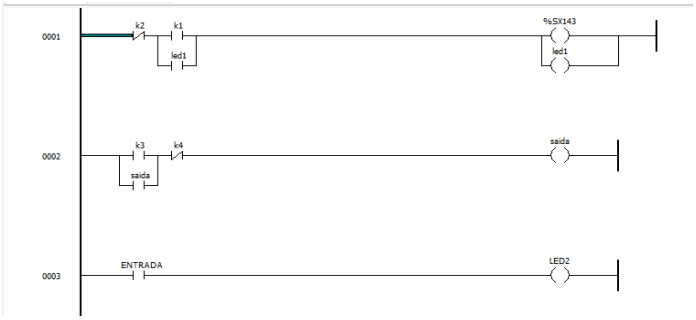

Fig 8. Programação em *ladder*.

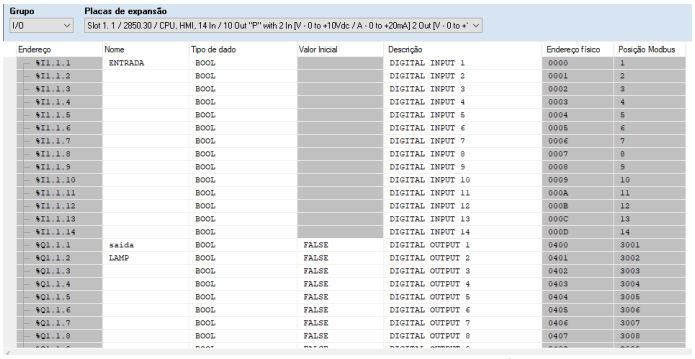

Fig 9. Campo de endereçamento de variáveis.

Após isso é feita a configuração do *baud rate*, da paridade, dos *data bits*, do *stopbits* e do n° de estação no Atos. A Figura 10 apresenta como foi *setado* os parâmetros.

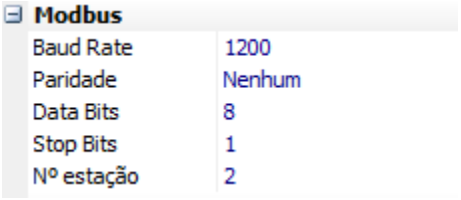

Fig 10. Configuração de parâmetros Modbus.

Então, com os procedimentos todos prontos para teste, foi realizada a programação para comunicar o Raspberry com um CLP Atos Expert BF 2850.30 através do protocolo Modbus RS-485, para uma operação de leitura.

Na Figura 11 é feita a importação das bibliotecas através do *import*. Na linha 5 é utilizado o diretório de onde se encontra a porta do dispositivo Conversor I485, a porta é identificada através do código *ls /dev/tty** no *Prompt pi* do Raspberry e o próximo parâmetro preenchido com o valor dois, indica o endereço do *slave*.

Após isso, são feitas as configurações de *baudrate*, *bytesize*, paridade, *stopbits* e *time out* de acordo com as configurações já feitas no Atos A1. Na linha 11 é configurada a biblioteca como modo RTU. Na linha 16 é utilizada a função *read_bit(x,y)* em que x é o endereço físico (o código da variável em hexadecimal deverá ser convertido para decimal) e o y é o código da função, ou seja, o tipo de solicitação que o mestre vai pedir para o escravo [12].

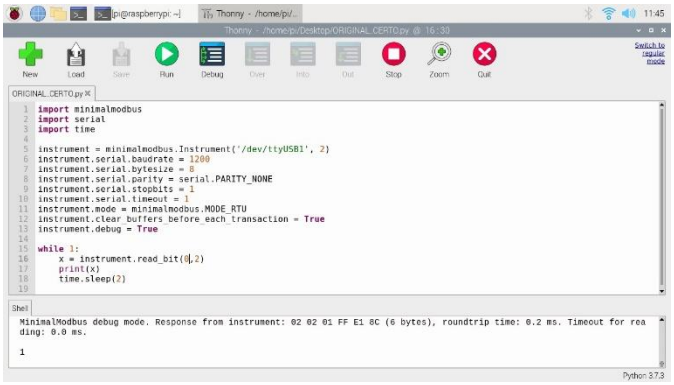

Fig 11. Tela do Raspberry Pi – Primeiro teste.

O código da função é definido através da tabela de funções no Protocolo Modbus representado na Tabela 2.

Tabela 2: Código da função do Modbus.

Código da Função	Descrição
1	Leitura de bloco de bits do tipo *coil* (saída discreta)
2	Leitura de bloco de bits do tipo de entradas discretas
3	Leitura de bloco de registradores do tipo *holding*
4	Leitura de bloco de registradores do tipo *input*
5	Escrita em um único bit do tipo *coil* (saída discreta)
6	Escrita em um único registrador do tipo *holding*
7	Ler o conteúdo de 8 estados de exceção

Fonte – FREITAS, 2014 [12].

Também foi feita a programação para se realizar a comunicação entre 2 CLP's e o Raspberry Pi.

Nessa programação são utilizados os mesmos parâmetros e configurações do teste com um CLP. Assim, fazendo os passos do primeiro teste e os incrementando com algumas mudanças no código.

Na linha 18, o *time.sleep* representa um valor de *delay* em segundos que segura o funcionamento do *while*. Na linha 19 é utilizado a função *break* para quebrar o ciclo do *while* e seguir para as próximas linhas. Após isso nas linhas seguintes são configurados os mesmos parâmetros já realizados com uma única mudança na linha 21, em que se troca o valor de endereço de *slave* de 2 para 3. Logo após isso é utilizado a função de *read_bit* para se realizar a leitura, e o valor é *printado* na tela.

As Figuras 12 e 13 mostram a seguir a programação em Python.

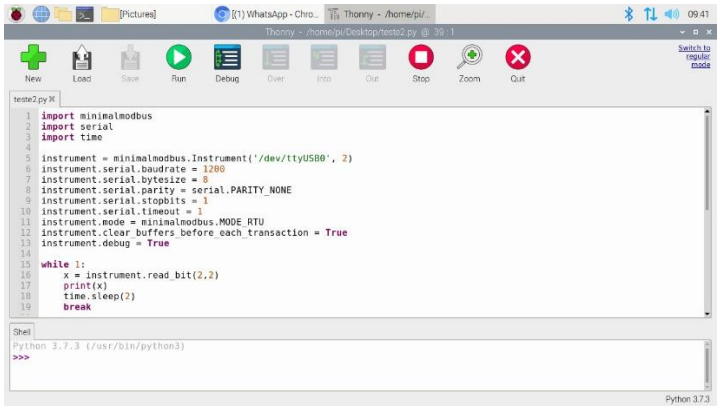

Fig 12. Tela do Raspberry Pi – Segundo teste – Parte 1.

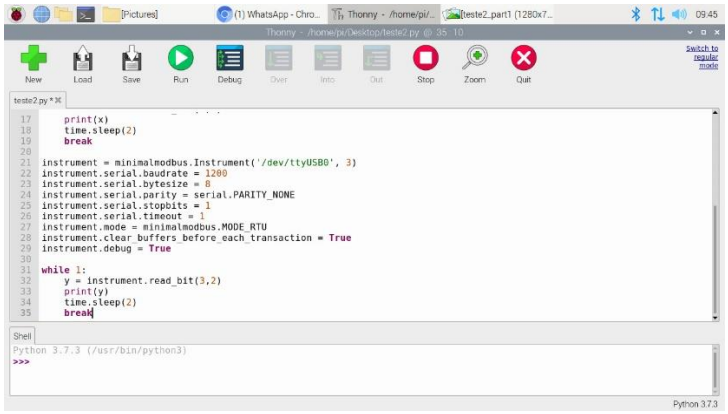

Fig 13. Tela do Raspberry Pi – Segundo teste – Parte 2.

Para se realizar a comunicação entre o Elipse E3 e o Raspberry, é necessário o emprego de uma biblioteca que faça o Raspberry atuar em modo escravo e a utilização de um *driver .dll* que possibilita a comunicação em modo Modbus no Elipse E3 que atuará como mestre. Esse *driver* possui algumas configurações padrão que precisam ser empregadas. Começando pela configuração do *Modbus Mode* na aba Modbus, onde escolhida a opção RTU Mode. A Figura 14 mostra a seguir.

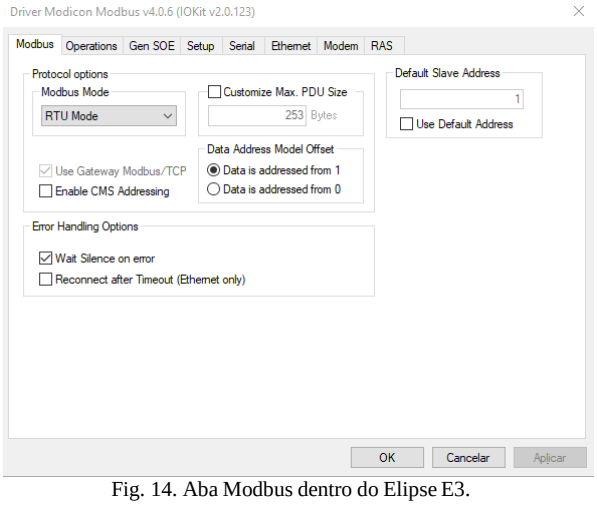

Fig. 14. Aba Modbus dentro do Elipse E3.

Na Figura 15 mostra a aba de Setup, no campo de *Physical Layer* é selecionada a opção “Serial”, para que o meio físico seja uma porta serial.

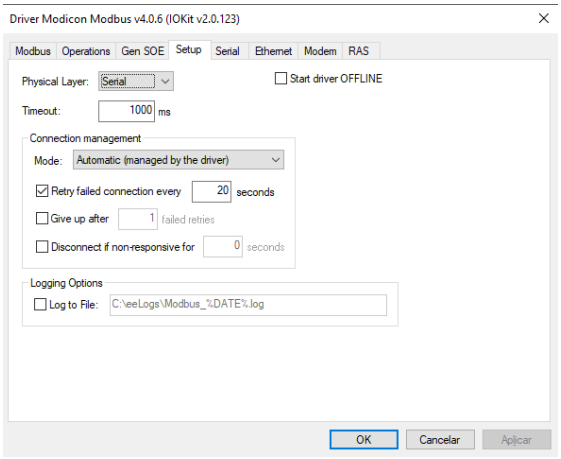

Fig. 15. Aba Setup dentro do Elipse E3.

Na Figura 16 tem a aba *Serial*, onde são configurados os valores de *baud rate*, a porta, os *data bits*, *stop bits*, e bit de paridade. O valor de 1200 de *baud rate* foi escolhido para melhor qualidade da resposta de acordo com os testes realizados anteriormente.

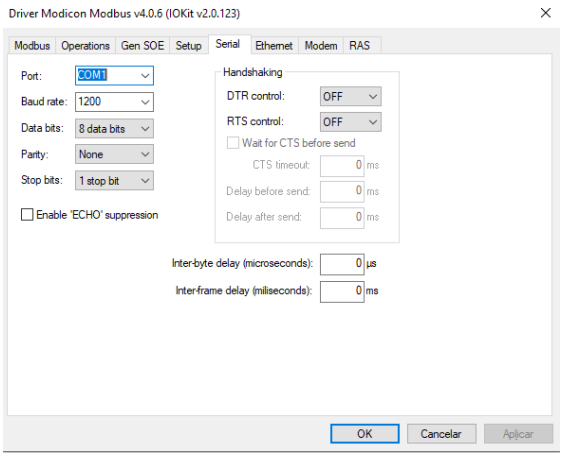

Fig. 16. Aba Serial dentro do Elipse E3.

Dentro do campo de criação de *tags* de comunicação referentes ao *driver*, existem quatro parâmetros que são responsáveis por identificar o código de operação, o endereço do escravo, e o endereço dos registros Modbus, sendo eles mostrados na Figura 17 [12].

- N1/B1: Endereço de Equipamento *Slave*/ID;
- N2/B2: Código de Operação;
- N3/B3: Não utilizado (Deixar em 0);
- N4/B4: Endereço do Registro Modbus ou bit.

Fig. 17. Aba de configurações *drivers* e OPC do Elipse E3.

Para este teste o simulador de rede Modbus da Elipse, Elipse Modbus Simulator foi utilizado para testar está comunicação. Para associar o Modbus, a biblioteca *pymodbus* e a função *ModbusTcpClient* foram requeridas neste teste.

Para se ter a associação deve se ativar o Simulator na máquina em que se deseja comunicar, pegando o IP (Figura 18). O IP é número exclusivo atribuído a cada computador e que tem a função de identificar um computador em uma rede. Através do código e com a biblioteca *pymodbus* importada para o Raspberry Pi se faz a comunicação, executando o código feito na IDE (*Integrated Development Environment*) do Raspberry Pi, a *Thonny*.

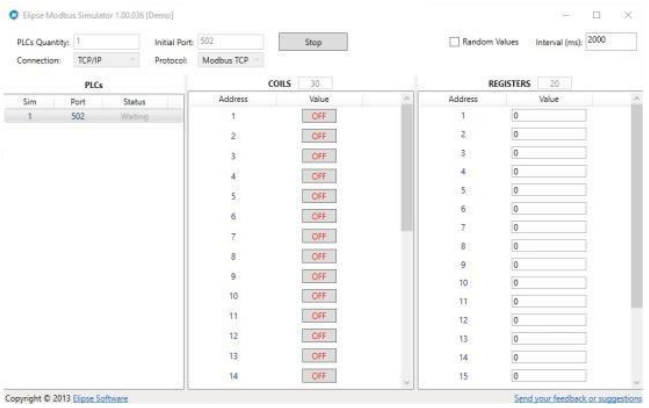

Fig 18. Ativação feita na máquina para comunicação com Modbus.

Feito isto, o desenvolvimento do código (Figura 19) para acertar a comunicação é uma conexão direta com a rede Modbus simulada pelo Elipse Modbus Simulator. O IP inserido representa o IP da máquina em que está rodando o simulador. O valor *True* retornado pela compilação do códigorepresenta uma conexão sucedida entre o Raspberry e o Simulator.

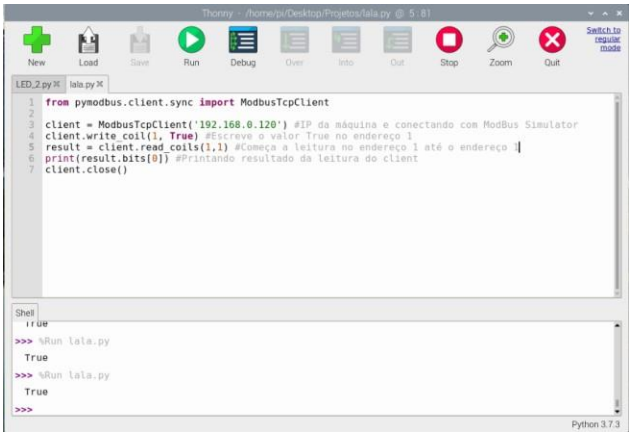

Fig. 19. Código de comunicação feita no IDE do Raspberry Pi.

IV. CONCLUSÃO

Abordar o conceito de redes industriais de uma forma mais palpável e simples, possibilita o entendimento e execução de todos os processos de forma mais claro, trazendo assim um maior conhecimento sobre o assunto em si e novas projeções para as crescentes inovações e mudanças no campo industrial.

Tendo como base toda a teoria exposta, foi aplicado uma comunicação entre o Raspeberry Pi e um CLP, através do meio físico RS-485, validando assim a comunicação entre ambos, utilizando de bibliotecas Python. Possibilitando uma gama de aplicações com este microprocessador, como a comunicação com o Elipse E3, que começou a ser parametrizado neste artigo.

Com o desenvolvimento da aplicação foram encontradas diversos contratempos, entre eles, ressalta a dificuldade de se encontar e moldar as bibliotecas Python de forma acertiva. Além disso, o fato de não de se ter um embasamento teórico sobre aplicações do Raspebrry Pi e bibliotecas que o tornam escravo, em uma comunicação serial, causa uma discrepância com o caminhar deste projeto.

Com a validação da comunicação e aplicação prática utilizando o Modbus com o Raspberry Pi é possível vislumbrar trabalhos acadêmicos futuros, tendo em vista que se abre um leque de opções muito grande, como a comprovação da comunicação com o Elipse E3, desenvolver uma comunicação OPC (*Open Platform Communications*) utilizando o Raspberry Pi e buscar uma aplicação como por exemplo, controle de temperatura, nível, vazão etc.

V. REFERÊNCIAS

[1] DE SOUZA, Rafaela. **Redes Industriais: Conceito e Aplicação.** Páginas 141 – 149. Franca, 2016. Acesso: 20 de setembro de 2021.

[2] CÉSAR, Aldo. **Redes Industriais: o que são e para que servem na Indústria 4.0.** 2018. Disponível em: https://transformacaodigital.com/mercado/redes-industriais-o-que-sao-e-para-que-servem-na-industria-4-0/. Acesso em: 23 de setembro de 2021.

[3] CASSIOLATO, C. **Revista Saber Eletrônica.** Edição 461, 2012, páginas 24-32. Acesso: 24 de setembro de 2021.

[4] ANSCHAU, Martin. **Sistema de Verificação Automática de Testes no Desenvolvimento de Firmwares de Inversores de Frequência.** Blumenau, 2019. Acesso: 17 de outubro de 2021.

[5] NOERGAARD, T**. Embedded System Architecture: A Comprehensive Guide for Engineers and Programmers.** [S.l.]: Elsevier Inc., 2013. 35, 36. Acesso em: 23 de outubro de 2021.

[6] HEATH, S. **Embedded Systems Design, 2nd Edition.** [S.l.]: Elsevier Inc, 2002. 36. Acesso em: 17 de novembro de 2021.

[7] RASPBERRY. 2021. Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Acesso em: 12 de setembro de 2021.

[8] GARRETT, Felipe. **Raspbian é o sistema operacional ideal para começar com um Raspberry Pi.** 2016. Disponível em: https://www.techtudo.com.br/tudosobre/raspbian.html#:~:text=Raspbian%20%C3%A9%20o%20sistema%20operacional%20ideal%20para%20come%C3%A7ar%20com%20um%20Raspberry%20Pi&text=Raspbian%20OS%20%C3%A9%20uma%20distribui%C3%A7%C3%A3o,do%20computador%20da%20Raspberry%20Foundation. Acesso em: 25 de outubro de 2021.

[9] ELIPSE SOFTWARE – **ELIPSE E3.** Disponível em: https://www.elipse.com.br/produto/elipse-e3/. Acesso em: 25 de setembro de 2021.

[10] CALDIÉRI, M. R. **Implementação do MODBUS para aplicação em sistema de controle via rede sem fio.** Dissertação de mestrado em Engenharia Elétrica. Universidade Estadual Paulista "Júlio de Mesquita Filho". 68 páginas. 2016. Acesso em: 7 de outubro de 2021.

[11] NI – National Instruments – **O protocolo Modbus em detalhes.** 2021. Disponível em: https://www.ni.com/pt-br/innovations/whitepapers/14/the-modbus-protocol-in-depth.html. Acesso em: 09 de novembro de 2021.

[12] FREITAS, Carlos Márcio – **Protocolo Modbus: Fundamentos e Aplicações.** Goiânia, 07 de abril de 2014. Disponível em: https://www.embarcados.com.br/protocolomodbus/#:~:text=O%20Modbus%20%C3%A9%20um%20dos,RS%2D232%3B&text=Ethernet%20TCP%2FIP%20(MODBUS%20TCP). Acesso em: 10 de outubro de 2021.

[13] NOVUS – **USB I485**. Disponível em: https://www.novus.com.br/site/?Idioma=55&TroncoID=508083&SecaoID=739080&SubsecaoID=727271&Template=../catalogos/layout_produto.asp&ProdutoID=838073. Acesso em: 10 de novembro de 2021.

[14] MODICON – **Driver Modicon Modbus – Modbus.dll.** 2020. Disponível em: https://www.elipse.com.br/downloads/?cat=48&key=modbus&language=ptbr. Acesso em: 14 de novembro de 2021.

[15] PyModbus – **A Python Modbus Stack.** 2017. Disponível em: https://pymodbus.readthedocs.io/en/latest/readme.html. Acesso em: 22 de setembro de 2021.

[16] CS – Control Solutions Minnesota – **Modbus 101 – Introduction to Modbus.** 2020. Disponível em: https://www.csimn.com/CSI_pages/Modbus101.html. Acesso em 03 de setembro de 2021.

[17] HRYNIEWICZ, Jacek. **Setup of Modbus TCP on Raspberry Pi.** 2021. Disponível em: https://jacekhryniewicz.wixsite.com/website/raspberry-pi-withmodbus-tcp. Acesso em: 10 de outubro de 2021.

[18] Raspberry Pi – **pymodbus rtu RS-485 communication.** 2014. Disponível em: https://raspberrypi.stackexchange.com/questions/22712/pymodbusrtu-rs-485-communication. Acesso em: 14 de outubro de 2021.

[19] NUNES, Arthur Henrique; AMARAL, Israel Filipe. Learning.py – **Uma apostila de introdução à programação em Python.** Belo Horizonte, 2020. Acesso: 3 de outubro de 2021.

[20] CORDEIRO, José Augusto. **Estudo dos principais protocolos de redes industriais utilizadas no Brasil: AS-I, MODBUS E PROFIBUS.** Ouro Preto, 2019. Acesso: 5 de outubro de 2021.

[21] DE SOUZA, Rafaela. **Redes Industriais: Conceito e Aplicação.** Páginas 141 – 149. Franca, 2016. Acesso: 31 de outubro de 2021.