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

Trabalho Final de Curso II

JOÃO VICTOR DUTRA MORAIS LIÃO
LUCAS SILVA COUTINHO

PROPOSTA DE REDE INDUSTRIAL COM ARDUINO UTILIZANDO O PROTOCOLO MODBUS

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. Dr. Bruno Quirino de Oliveira – Orientador. PUC Goiás.

Prof. Dr. Antônio Marcos de Melo Medeiros – Banca. PUC Goiás.
Prof. Me. Luís Fernando Pagotti – Banca. PUC Goiás.

Goiânia 08 de Dezembro de 2021

PROPOSTA DE REDE INDUSTRIAL COM ARDUINO UTILIZANDO O PROTOCOLO MODBUS

João Victor Dutra Morais Lião, Lucas Silva Continho, Bruno Quirino de Oliveira, Antônio Marcos de Melo Medeiros e Luís Fernando Pagotti

***Resumo* — Neste artigo será relatado e demonstrado como a rede industrial pode ser relacionada com a comunicação *ModBus* e seu funcionamento para uma criação de uma rede de transmissão de dados. Para gerir e ter uma exatidão no projeto é necessário conhecer o significado e entender o funcionamento de cada etapa e dispositivo utilizado na aplicação, assim como, quais as funcionalidades que o sistema pode oferecer aos possíveis clientes além do sistema supervisório de controle através do Arduino. A partir desse protocolo de comunicação é possível obter resultados satisfatórios, o que torna essa aplicação atrativa, quando se busca maior eficiência e baixo custo.**

Palavras-chave — Redes Industriais, Sistemas Supervisórios, *ModBus*, Transmissão de dados

Abstract – In this article, it will be reported and demonstrated how the industrial network can be related to Modbus communication and its functioning to create a data transmission network. To manage and have an accuracy in the project, it is necessary to know the meaning and understand the functioning of each step and device used in the application, as well as what features the system can offer to a potential customer in addition to the supervisory control system through Arduino. From this communication protocol it is possible to obtain satisfactory results, which makes this application attractive, when seeking greater efficiency and low cost.

Keywords – Industrial Networks, Supervisory Systems, ModBus, data transmission

I. INTRODUÇÃO

Com o avanço tecnológico atual, é necessário estar antenado nas atualizações que o mercado oferece para que seu sistema de comunicação esteja sempre atualizado e pronto para oferecer a melhor conexão de dados possível. Sempre que as empresas/indústrias crescem, os processos se tornam cada vez mais complexos e com diversas variações e, para que não aja erros, é necessário ter um alto grau de controle e regulação do sistema para que dessa forma seja possível otimizar os serviçoes e as linhas de produçaõ com a finalidade de maximizar a produção e os lucros dos clientes [1].

Redes industriais são uma ótima ferramenta para alcançar esses objetivos. Uma maneira de definir as redes industriais seria que são uma forma de automação industrial, que utilizando protocolos de comunicação são usadas para supervisionar e gerenciar processos de uma empresa ou fábrica, através da troca de informações entre sensores, computadores e máquinas de uma forma mais veloz, eficaz e precisa [2].

A proposta deste artigo é descrever sobre as redes indústrias e protocolos de comunicação com o intuito de poder validar a comunicação serial Modbus com o meio físico RS485 utilizando o microcontrolador arduino e conversores necessários. Com o objetivo de futuramente conectar com o *software* Elipse E3 formando um sistema supervisório para realizar supervisão e gerenciamento dos processos, desde que seja realizado as alterações necessárias para cada caso específico, visto que, é um projeto variado e que possibilita diversas aplicabilidades.

II. FUNDAMENTAÇÃO TEÓRICA

A - REDES INDUSTRIAIS

O que difere a tecnologia das redes industriais das demais é o fato dela atuar de forma automática, dedicada ao contexto e ao ambiente industrial ao qual foi projetada para operar e ter êxito.

Para a implementação dessas redes, é necessário construir uma infraestrutura adequada, implementar um sistema de segurança e realizar conexões utilizando a Internet das Coisas (IoT) como um conjunto de armazenar dados [2].

Esse conceito é algo diretamente ligado ao conceito da indústria 4.0, tendo em vista que as redes industriais tornam os processos mais inteligentes e independentes, proporcionando uma ótima eficiência do processo de produção [2].

Existem diversos tipos de redes industriais que podem ser divididos em: *sensorbus, devicebus, fieldbus* e Ethernet, sendo que essa última deu origem a diversos protocolos que podem ou não ser utilizado dependendo da aplicação. Os protocolos mais utilizados são: TCP/IP, *Modbus/* TCP, *profinet* e *ethernet/*IP [2]. O protocolo que será utilizado para essa aplicação de rede industrial é o *Modbus/* TCP.

B – PROTOCOLO *MODBUS* RS-485

O *Modbus* é um protocolo de comunicação de dados que é utilizado em sistemas de automação. Este protocolo foi criado em 1979 e continua sendo um dos mais utilizados em CLP's até hoje. Um dos modelos mais utilizados de comunicação é o RTU (*Remote Terminal Unit)* [3].

O modelo RTU do *Modbus* é um protocolo empregado para desenvolver comunicação do tipo mestre/escravo *(master/slave)* entre dispositivos inteligentes. Seguindo essa comunicação interface do tipo RS-485 pode ser utilizada como camada física dependendo das especificações do projeto.

O RS-485 é uma camada física do tipo *half-duplex*, onde dados podem ser transmitidos e recebidos um após o outro. Ela é capaz de interligar até 32 unidades de carga de (15kΩ) [4]. Esses meios físicos variam entre si em aspectos como a taxa de transmissão de dados e na maioria dos casos são utilizados cabos de par trançado para isso.

O padrão RS-485 (*Recommendad Standart*-485) permite trabalhar com taxas de comunicação que podem chegar a 12Mbps e em alguns casos até 50Mbps, vale lembrar que quanto maior o comprimento da rede menor será a velocidade de comunicação. A distância máxima da rede está em torno de 1200m e o número máximo de dispositivos no barramento da rede é de 32 [5].

O funcionamento desse protocolo consiste em: camada física, que é responsável pelo endereço do dispositivo escravo, bits de início e fim, código CRC (código para detecção de erros) e *time out*. Junto a essa função existe também a camada de dados, responsável pela função de rejeitar ou aceitar mensagens, reposicionar dados e manter a comunicação em estado ocupado [3].

Neste protocolo, a mensagem possui tamanho de 32 *bits* do tipo ponto flutuante ou inteiro. Para estabelecer comunicação, o dispositivo mestre envia uma pergunta ao dispositivo escravo, que responde de acordo com o que foi recebido. Os tipos e tamanhos de dados *Modbus* de um dispositivo estão ilustrados na Figura 1 [3].

Figura 1. Formato de mensagem do protocolo *Modbus* [3].

Levando em consideração o que foi descrito anteriormente, é possível afirmar que os dados do dispositivo escravo podem ser lidos por um mestre (computador), através do protocolo. Para estabelecer a comunicação o escravo deve ter um endereço próprio, que é acessado pelo mestre. A partir disso, dados referentes ao escravo podem ser lidos ou escritos. Que possuem também endereçamento próprio, com tamanhos variáveis de informações. A Figura 2 apresenta como é realizada a troca de informações entre os dispositivos mestre/escravo [3].

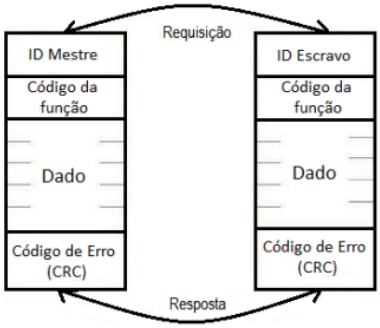

Figura 2. Exemplo de troca de mensagem entre mestre e escravo [3].

Analisando a descrição do protocolo de comunicação Modbus mencionado acima, pode ser percebido que suas implementações físicas e lógicas não oferecem muitas dificuldades, pois não necessitam de muitos recursos, assim, tornando o projeto mais eficiente e viável.

No protocolo *Modbus*, existe o código de função onde o mestre especifica o tipo de serviço ou função solicitada ao escravo. Cada função é utilizada para um tipo específico de dado, a Tabela 1 apresenta alguns exemplos de código de função *Modbus* [5].

Tabela 1. Código de Função [5].

Código da Função	Descrição
1	Leitura de bloco de bits do tipo coil (saída discreta).
2	Leituras de blocos de bits do tipo entradas discretas.
3	Leitura de bloco de registradores do tipo holding.
4	Leitura de blocos de registradores do tipo input.
5	Escrita em um único bit do tipo coil (saída discreta).

Para a montagem de um meio físico dessa aplicação com um Arduino é necessário a utilização de um módulo conversor do protocolo para realizar a comunicação com o arduino esse hardware é o Max485 Modulo Conversor TTL (Lógica transístor-transistor) mostrado na Figura 3.

Figura 3. Max485 Modulo Conversor.

Este módulo RS-485 está baseado em um circuito integrado MAX485 que converte sinais TTL em sinais compatíveis com RS-485 *standart.* O módulo também funciona como receptor RS-485. A comunicação através desse modo pode ser de *half-duplex*.

Ele trabalha em uma única fonte de alimentação de 5 V e de corrente nominal de 300 mA além de poder atingir uma taxa de transmissão máxima de 2.5 Mbps.

C – ARDUINO UNO

O arduino é um microcontrolador, conforme mostrado na Figura 4, que é conectado a um computador através de um barramento serial universal (USB). Possui a finalidade de ser usado como uma interface onde o mesmo irá se comunicar com o computador através de linguagens de programação para que dessa forma possa controlar circuitos eletrônicos e comandos de dispositivos [6].

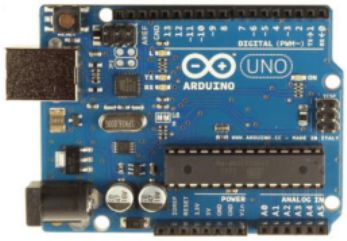

Figura 4. Arduino UNO [8].

O Arduíno é um dispositivo formado por dois componentes básicos: a placa arduino, que é o elemento de *hardware* utilizado para construir seus objetivos e a IDE (*Integrated Development Environment*), que é um programa de computador onde escreve os códigos (*sketch*) e que fará o upload para a placa arduino logo em seguida [7].

Os arduinos estão presentes em diversos tipos de equipamentos e aparelhos eletrônicos devido a seu baixo custo e a infinidade de opções para uso. No dia a dia é possível encontrar vários aparelhos que são usados com frequência que utilizam microcontroladores, por exemplo: televisões, brinquedos, carros, motos, celulares, micro-ondas, sensores. O arduino processa diversos sensores ao mesmo instante e com isso, é possível agregar diversos parâmetros de forma simultânea.

Existem diversos modelos de arduinos no mercado e quando as pessoas vão em busca do mesmo ficam muitas vezes perdidas sem saber qual modelo em específico comprar. Existem diversos modelos pelo simples fato do arduino poder controlar desde um simples processo como um controle remoto até processoas complexos como um braço robótico em uma linha de produção de uma fábrica. As principais diferenças entre os modelos são: tamanho, quantidade de memória e quantidade de portas. Os principais modelos de arduinos encontrados no mercado são: Arduino UNO, Arduino Leonardo, Arduino Mega 2560, Arduino Mega ADK, Arduino Due, Arduino Nano, Arduino Pro Mini e o Arduino Esplora [8].

Para realizar o projeto em questão, foi escolhido a utilização do Arduino UNO devido ao fato dele ser um dos mais comuns no mercado, seu baixo preço, atende a necessidade do projeto e possui 14 portas digitais, sendo que 6 podem ser utilizadas como saídas PWM (*Pulse Width Modulation*) e possui 6 portas analógicas. Ele opera com o microcontrolador Atmega328 que é montado sobre um *socket* na placa (fácil remoção e troca, caso necessário) o que é uma grande vantagem em relação aos demais arduinos, pois se caso queimar o microcontrolador basta trocar o mesmo sem danificar a placa, já os demais, utilizam esse microcontrolador soldado a placa, ou seja, se o mesmo vir a queimar terá dificuldades em trocar o componente e em alguns casos pode ser necessário trocar a placa ou comprar outro arduino [8].

Suas especificações são:

- Alimentação de 7-12Vdc
- Tensão de trabalho 5Vdc
- 32kb de memória.

O microcontrolador pode ser programado em qualquer computador que tenha suporte ao *software* do desenvolvedor que é fornecido de forma gratuita pelo site do mesmo e seu funcionamento consiste em: um sistema autônomo computadorizado em apenas um circuito integrado onde é utilizado a linguagem C e C++ e interface gráfica formado em Java [9].

A plataforma de desenvolvimento do Arduino é um *software* que contém um editor de texto para escrever o programa, uma área de mensagens, uma barra com botões para funções mais comuns, uma série de menus. Esse *software* se conecta a placa Arduino, através de um cabo USB, e transfere o programa para a placa. Os códigos desenvolvidos para o Arduino são chamados de *sketches* [9].

D – ELIPSE E3

O *software* Elipse E3 foi o escolhido para o desenvolvimento do sistema supervisório para poder ser integrado na rede industrial proposta pelo simples motivo de que ele é uma poderosa plataforma para supervisão e controle de processos voltada à operação em rede e aplicações distribuídas. Ele é desenvolvido pela Elipse *Software*, uma empresa brasileira especializada no desenvolvimento de soluções para o gerenciamento de processos em tempo real [10].

A Figura 5 mostra a arquitetura do *software* de terceira geração Cliente/Servidor HMI/SCADA que é totalmente orientado para a operação em rede, ele oferece um avançado modelo de objetos, uma poderosa interface gráfica e uma nova e exclusiva arquitetura que permite não só desenvolvimento rápido de aplicações, mas também o máximo de conectividade a vários equipamentos e outras aplicações [10].

Além de tudo incorpora as mais recentes tecnologias de desenvolvimento de *software,* maximizando o desempenho e a produtividade, otimizando a qualidade de suas aplicações e seu processo de desenvolvimento, e também minimizar perdas e custos.

O Elipse E3 é formado por três programas principais do ponto de vista do usuário, sendo eles:

- *E3 Server*
- *E3 Viewer*
- *E3 Studio*

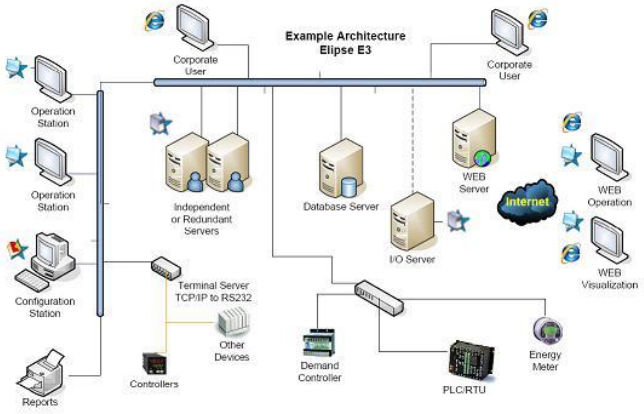

Figura 5. Exemplo de Arquitetura do E3 [10].

A versão utilizada no projeto é a *demo,* onde o máximo de TAG's de comunicação são 20, somente o primeiro objeto de cada seção da galeria está disponível e a aplicação é executada até no máximo por uma hora.

III. PROJETO PROPOSTO

A primeira parte do projeto a ser realizada foi o teste de transmissão e recepção de dados através de duas placas arduino uno, onde foram desenvolvidos dois códigos com as funções específicas para ser implementado em cada um. A Figura 6 apresenta o esquema de *hardware* utilizado para essa aplicação.

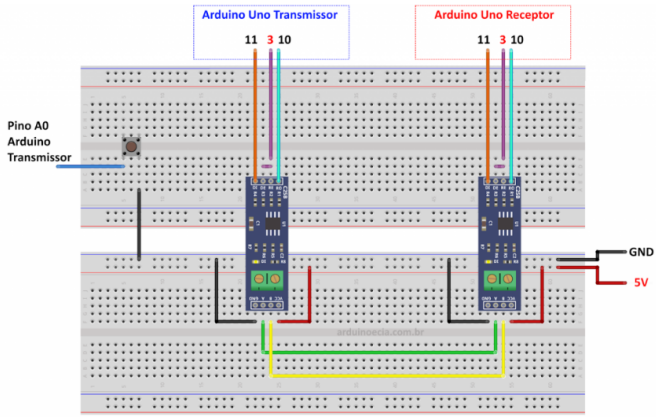

Figura 6. Esquema de *harware* para implementação de transmissor e receptor.

O primeiro arduino (mestre) tem como funcionamento a transmissão de dados do sistema, enquanto o segundo funciona como receptor dos dados (escravo). Essa aplicação é possível através de uma comunicação serial *Modbus*, onde os dados a serem enviados ao escravo são pulsos elétricos. A Figura 7 mostra o esquema das ligações dos microcontroladores de forma teórica e esquemática, enquanto, a Figura 8 demonstra o projeto prático propriamente dito e como o mesmo ficou.

Para realizar a montagem do projeto descrito na Figura 8 é necessário: dois Arduino UNO, dois conversores MAX485, uma placa *Protoboard*, um botão físico, um resistor de 1kΩ, um conputador que suporte o *software* e cabos para os Arduinos e para ligações na placa *Protoboard.*

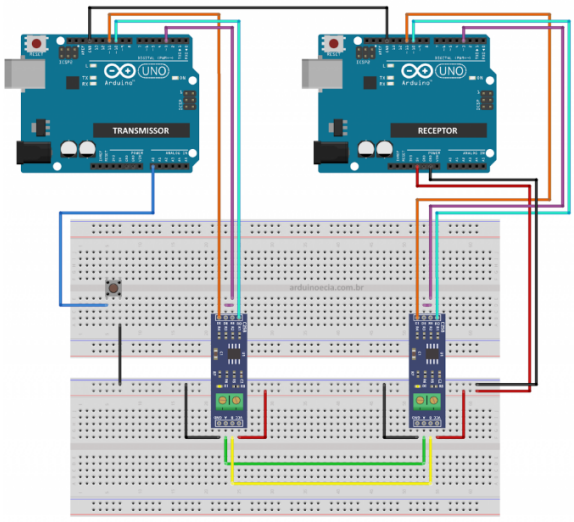

Figura 7. Esquema de *harware* para implementação de transmissor e receptor com arduino inserido.

Antes de chegar no modelo final descrito na Figura 8, foram encontrados algumas dificuldades e obstáculos no projeto prático. Uma das primeiras dificuldades foi conectar corretamente os dois arduinos ao computador, visto que, as portas que estavam encontrando (de forma automática) não estavam vinculando de maneira correta.

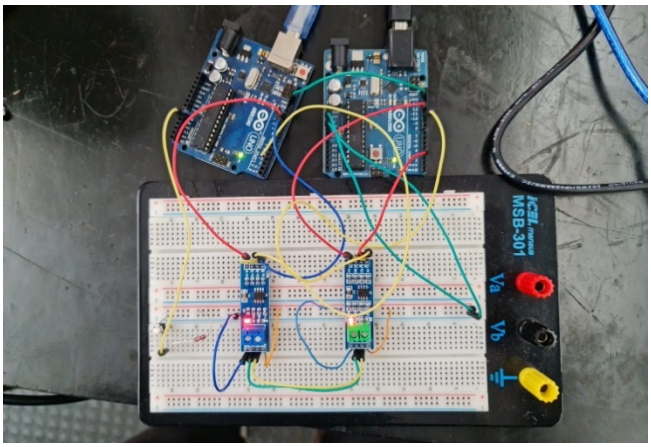

Figura 8. Esquema prático do projeto proposto.

O outro obstáculo encontrado consistiu no fato do sistema estar flutuando e ao pressionar o botão o pulso elétrico não acionava a luz como deveria. Para resolver esse problema foi necessário incrementar ao sistema um resistor de 1kΩ em série com o botão de acionamento, conforme Figura 9. Nessa imagem a fonte de tensão de 5 V ocupa o lugar do Arduino e na prática, o que alimenta o sistema é a saída de 5 Vcc.

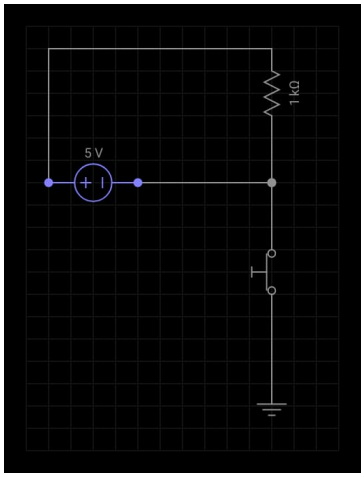

Figura 9. Resistor em série com botão.

A aplicação somente é possível através dos códigos específicos para cada um dos microcontroladores onde o Anexo 1, apresenta a implementação para o arduino mestre que funciona como transmissor do sistema proposto, onde no código é criado uma serial por *software* nos pinos 10 e 11 do Arduino, liberando a serial por *hardware.* Dessa forma, é possível acompanhar pelo serial monitor o que está sendo enviado pela placa. Além disso, o led 13 *onboard* do Arduino é utilizado para mostrar quando o dado está sendo enviado, e antes de enviar foi habilitado o modo de transmissão, então envia a *string* e depois desabilita o modo de transmissão.

Já no Anexo 2 é mostrado o código de implementação para o Arduino escravo da rede onde ele funciona como receptor do sistema. Onde ele possui as mesmas configurações de comunicação do tranmissor (Rx e Tx nos pinos 10 e 11 do Arduino). A *string* é recebida pela serial do RS485 e vai sendo "montada" caracter a caracter e armazenada na variável *inputString*. Quando o programa recebe o valor de final de *string (n),* o valor de *inputString* é enviado para a serial.

A biblioteca utilizada para o desenvolvimento desses códigos foi a *SoftwareSerial.h* que utiliza comunicação serial para a linha de transmissão do projeto proposto. Ao final dos testes foi obtido um resultado satisfatório para o que foi proposto, conforme mostrado na Figura 10 que evidencia o sucesso da comunicação entre os arduinos.

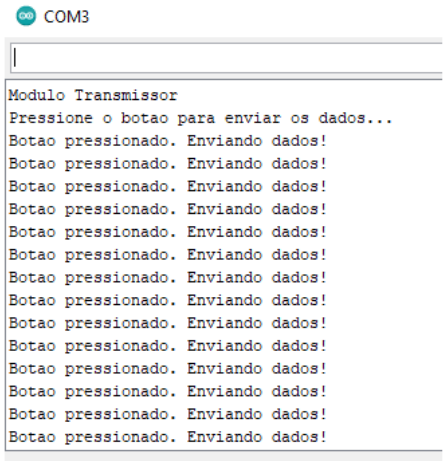

Figura 10. Mostragem de funcionamento do código de transmissão de dados.

A Figura 11 apresenta o fluxograma do sistema que foi montado para especificar os processos passo a passo dessa montagem.

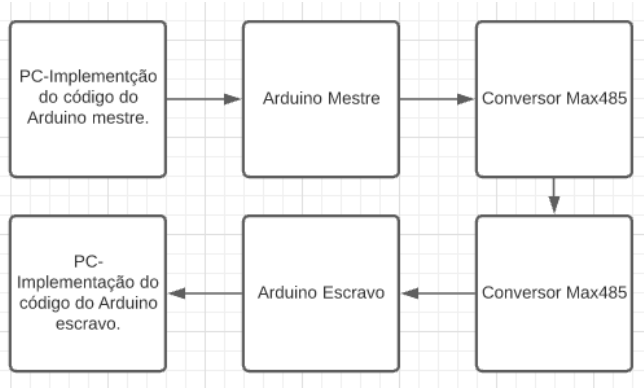

Figura 11. Fluxograma do sistema proposto.

Validando esta comunicação é possível inserir o sistema supervisório, que será o mestre da rede e os microcontroladores que serão os escravos. A Figura 12

apresenta um esquema que representa essa aplicação.

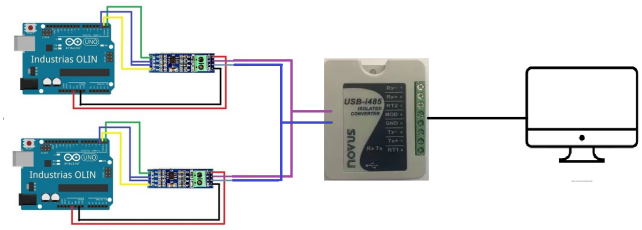

Figura 12. Exemplo de rede industrial com sistema supervisório.

A configuração dos dispositivos escravos da rede deve seguir as orientações da biblioteca *SoftwareSerial.h*, conforme detalhado no Quadro 2. Para se realizar a comunicação entre o Elipse E3 e o Arduíno, é necessário o emprego de uma biblioteca que faça que o Elipse possa atuar em modo mestre, e para isso é necessário a utilização de um *driver .dll.* Ele possibilita a comunicação em modo Modbus no Elipse E3.

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. É necessário também a configuração dos parâmetros da rede serial como por exemplo, valores de *baud rate*, a porta, o *data bits*, *stop bits*, e *bit* de paridade.

Após configurado o driver são criadas *tags* de comunicação. Para a criação delas existem quatro parâmetros que são responsáveis por identificar o código de operação, o endereço do escravo, e o endereço dos registros Modbus, conforme apresentado a seguir:

- 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*.

IV. CONCLUSÃO

Existe uma infinidade de possibilidades para utilização de sistema supervisório baseado na comunicação *modbus*. Porém, foi verificado e constatado que a comunicação entre os equipamentos é uma das partes mais importantes do projeto, pois qualquer erro de comunicação entre as partes pode resultar em severos danos ao sistema.

Com os testes desenvolvidos ao decorrer do projeto foi possível perceber que os processos de comunicação encontram várias dificuldades para serem regularizados. Para contornar essas dificuldades utilizamos dois diferentes códigos para regularizar a comunicação serial *Modbus*.

Outra grande dificuldade ocorreu no momento de realizar a montagem do projeto prático onde foi visto que o sistema estava oscilando e não executava os comandos de maneira satisfatória. A solução desse problema foi sanada com a inserção de um resistor de 1kΩ no sistema juntamente com um botão de acionamento.

Com essa implementação foi possível obter uma rede de transmissão funcional onde os dados em *bits* são transmitidos através de pulsos elétricos para um receptor que capta esses dados e faz com que toda vez que o botão for acionado manualmente uma luz irá piscar na placa arduino. Comprovando assim a eficácia da comunicação do sistema.

Após o êxito obtido com a comunicação do sistema, o próximo passo foi incrementar um sistema supervisório via o Elipse E3. Todavia, ao simular os testes, foi verificado uma incompatibilidade no sistema que impossibilitou a continuidade do projeto e das idéias propostas devido a um erro de comunicação, bibliotecas incompletas e ligação entre os equipamentos. Infelizmente não foi possível achar uma solução viável por falta de prazo de entrega do trabalho.

Sugestões de trabalhos futuros: executar uma implementação utilizando a comunicação com o Elipse E3 e com uma aplicação, como por exemplo, de um sistema de alarmes onde o transmissor iria enviar os dados e o receptor faria a leitura deles para apresentar ao usuário.

V. REFERÊNCIAS

[1] ANTONELI, Gustavo Henrique Dos A, ALVES, Yan C. Controle de Irrigação Usando Inversor de Frequência em Rede ModBus. – Goiânia, 2020.

[2] CÉSAR, Aldo. Redes Industriais: o que são e para que servem na indústria 4.0. **Transformação Digital**, 2021. Disponível em: <https://transformacaodigital.com/mercado/redes-industriais-o-que-sao-e-para-que-servem-na-industria-4-0/>. Acesso em 16 de maio de 2021.

[3] ANSCHAU, Martin H. Sistema de Verificação Automática de Testes no Desenvolvimento de Firmwares de Inversores de Frequência, Blumenau 2019.

[4] PERRIM, B. The Art and Science of RS-485. – Circuit Cellar Magazine, Jul 1999.

[5] FREITAS, Carlos Márcio. Protocolo Modbus: Fundamentos e Aplicações. **Embarcados,** 2014. Disponível em: https://www.embarcados.com.br/protocolo-modbus/. Acesso em 02 de novembro de 2021.

[6] MARTINAZZO, C. A. Arduino: Uma tecnologia no ensino de física. Erechim, 2014.

[7] CAMPOS, R. A. F. Automação residencial utilizando arduino e aplicação web. Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação. Brasília, 2014. 85 p. Disponível em: http://repositorio.uniceub.br/bitstream/235/5461/1/Monografia_Roberto.pdf.

[8] TIPOS DE ARDUINO – QUAL ARDUINO ESCOLHER. **Athos Eletronics,** 2020. Disponível em: <https://athoselectronics.com/qual-arduino-comprar/> Acesso em 16 de maio de 2021.

[9] CAETANO, P. Linguagem Arduino. Disponível em:<https://professorcaetano.wordpress.com/microcontrolador/linguagem-arduino>. Acesso em 16 de maio de 2021.

[10] SALVADOR, Marcelo B. Como Funciona o Elipse E3. **Elipse Knowledgebase,** 2019. Disponível em: <https://kb.elipse.com.br/como-o-elipse-e3-funciona/>. Acesso em 20 de maio de 2021.

[11] ANTONELI, Gustavo Henrique Dos A, ALVES, Yan C. Controle de Irrigação Usando Inversor de Frequência em Rede ModBus. – Goiânia, 2020.

[12] MCROBERTS, M. Arduino Básico. São Paulo: Novatec, 2018

[13] MORAES, C. C. CASTRUCCI PL. Engenharia de Automação Industrial. – Rio de Janeiro: LTC, 2007.

[14] RUIZ, J. Communication protocol for a router-based building automation and control network. U.S.. Patent n. 5,916,306, 29 jun. 1999

[15] VALLE, A. Plataformas de e-commerce open source. 2017. Disponível em: <http://www.guiadeecommerce.com.br/plataformas-de-ecommerce-open-source/> Acesso em 16 de maio de 2021.

[16] DIAS, S. Aplicações móveis na monitorização de dados biométricos na DPOC. Disponível em: https://ria.ua.pt/bitstream/10773/13607/1/Tese.pdf

[17] FREQUENTLY ASKED QUESTIONS ABOUT RS-485. **Control Solutions Minnesota**, 2020. Disponível em: <https://www.csimn.com/CSI_pages/RS-485-FAQ.html>. Acesso em 16 de maio de 2021.

[18] READ DATA FROM CONTROLLER VIA RS485 MODBUS RTU ARDUINO. **Stack Exchange,** 2021. Disponível em: <https://arduino.stackexchange.com/questions/75383/read-data-from-controller-via-rs485-modbus-rtu-arduino>. Acessado em 16 de maio de 2021.

ANEXO 1

```
//Programa: Comunicação RS485 com Arduino - Transmissor
#include <SoftwareSerial.h>
//Pinos de comunicacao serial do modulo RS485
#define Pino_RS485_RX    10
#define Pino_RS485_TX    11
//Pino de controle transmissao/recepcao
#define SSerialTxControl 3
#define RS485Transmit    HIGH
#define RS485Receive     LOW
//Define led 13 para mostrar atividade na comunicacao
#define Pin13LED         13
//Cria a serial por sofware para conexao com modulo RS485
SoftwareSerial RS485Serial(Pino_RS485_RX, Pino_RS485_TX);
void setup()
{
  //Inicializa a serial do Arduino
  Serial.begin(9600);
  Serial.println("Modulo Transmissor");
  Serial.println("Pressione o botao para enviar os dados...");
  pinMode(Pin13LED, OUTPUT);
  pinMode(SSerialTxControl, OUTPUT);
  //Inicializa a serial do modulo RS485
  RS485Serial.begin(4800);
  //Seta o pino A0 como entrada e habilita o pull up
  pinMode(A0, INPUT_PULLUP);
}
void loop()
{
  //Verifica se o botao foi pressionado
  int  valor = digitalRead(A0);
  if (valor == 0)
  {
    Serial.println("Botao pressionado. Enviando dados!");
    //Habilita o modulo para transmissao
    digitalWrite(SSerialTxControl, RS485Transmit);
    //Envia a string
    RS485Serial.println("Botao acionado");
    //Liga o led 13 para mostrar que ha conexao
    digitalWrite(Pin13LED, HIGH);
    delay(10);
    digitalWrite(Pin13LED, LOW);
    //Desabilita o modulo para transmissao
    digitalWrite(SSerialTxControl, RS485Receive);
    while (digitalRead(A0) == 0)
    {
      delay(50);
    }
  }
}
```

ANEXO 2

```
//Programa: Comunicação RS485 com Arduino - Receptor
#include <SoftwareSerial.h>
//Pinos de comunicacao serial do modulo RS485
#define Pino_RS485_RX    10
#define Pino_RS485_TX    11
//Pino de controle transmissao/recepcao
#define SSerialTxControl 3
#define RS485Transmit    HIGH
#define RS485Receive     LOW
//Define led 13 para mostrar atividade na comunicacao
#define Pin13LED         13
//Cria a serial por sofware para conexao com modulo RS4850
SoftwareSerial RS485Serial(Pino_RS485_RX, Pino_RS485_TX);
//Armazena os dados que chegam pela serial
String inputString = "";
//Variavel de string completa
boolean stringComplete = false;
void setup()
{
 //Inicializa a serial do Arduino
 Serial.begin(9600);
 Serial.println("Modulo Receptor");
 Serial.println("Aguardando dados...");
 pinMode(Pin13LED, OUTPUT);
 pinMode(SSerialTxControl, OUTPUT);
 //Coloca o modulo RS485 em modo de recepcao
 digitalWrite(SSerialTxControl, RS485Receive);
 //Inicializa a serial do modulo RS485
 RS485Serial.begin(4800);
}
void loop()
{
 //Recebe os dados do RS485 via porta serial
 if (RS485Serial.available())
 {
  while (RS485Serial.available())
  {
   //Recebe os dados e monta a string
   char inChar = (char)RS485Serial.read();
   inputString += inChar;
   if (inChar == 'n')
   {
    //Mostra no Serial Monitor a string recebida
    Serial.print(inputString);
    stringComplete = true;
    inputString = "";
   }
  }
 }
}
```