ENSINO DE SISTEMAS SCADA, IHM, E PROTOCOLO MODBUS USANDO ARDUINO

DOI: 10.37702/2175-957X.COBENGE.2022.4013

Francisco Borges Carreiro - fborges@ifma.edu.br
Instituto Federal de Educação Ciência e Tecnologia do Maranhão

Felipe Cancio Costa - cancio.felipe@acad.ifma.edu.br
IFMA

Resumo: *Este trabalho apresentou a proposta e o desenvolvimento de hardware e software para o ensino de disciplinas como sistemas supervisório e IHM muito utilizadas nas matrizes curriculares de cursos técnicos em escolas técnicas e universidades. O hardware é composto de um Arduino Uno, botão de pressão, LEDs, e potenciômetro. O software correspondentes às telas foram desenvolvidas usando softwares na versão de demonstração do Elipse SCADA para o sistema supervisório e o EasyBuilder Pro para a IHM, enquanto que para a programação do Arduino, foi utilizada a IDE gratuita do Arduino e a biblioteca Modbusino também gratuita. Foi possível desenvolver telas que foram utilizadas para a visualização de dados nas aulas online durante a pandemia de COVID19, bem como capacitar alunos a desenvolverem seu próprio hardware e telas de supervisão tanto para sistemas SCADA como para IHM. Os resultados foram surpreendentes porque possibilitaram o ensino prático de disciplinas que normalmente só podem ser ensinadas em laboratório, além de apresentar um grau elevado de satisfação na aprendizagem destas disciplinas.*

Palavras-chave: *scada, hmi, modbus, Arduino Uno*

ENSINO DE SISTEMAS SCADA, IHM, E PROTOCOLO MODBUS USANDO ARDUINO

1 INTRODUÇÃO

Devido a impossibilidade de acesso aos laboratórios do Instituto Federal do Maranhão por causa das medidas sanitárias durante a pandemia de Covid19, o ensino prático de disciplinas de engenharia elétrica tais como Sistemas Supervisórios e Interface Homem Máquina (IHM), impuseram desafios para ministrá-las de forma prática. Um desafio foi como permitir que o aluno pudesse visualizar os resultados reais nas aulas *online* em vez de apenas simulações, e outro foi como capacitar o aluno a desenvolver seus próprios experimentos em casa. Uma solução foi capacitar o aluno a desenvolver seu próprio *hardware* de forma simples, com baixo custo e acessível à maioria dos alunos. Pensando nisso desenvolvemos um experimento simples com base em Arduíno e componentes discretos como botões, LEDs, potenciômetro, *protoboard* e *softwares* livre e/ou abertos que são acessíveis à maioria dos alunos.

Este trabalho apresenta e discute o *hardware* e o *software* desenvolvidos, bem como, os resultados das experiências desenvolvidas.

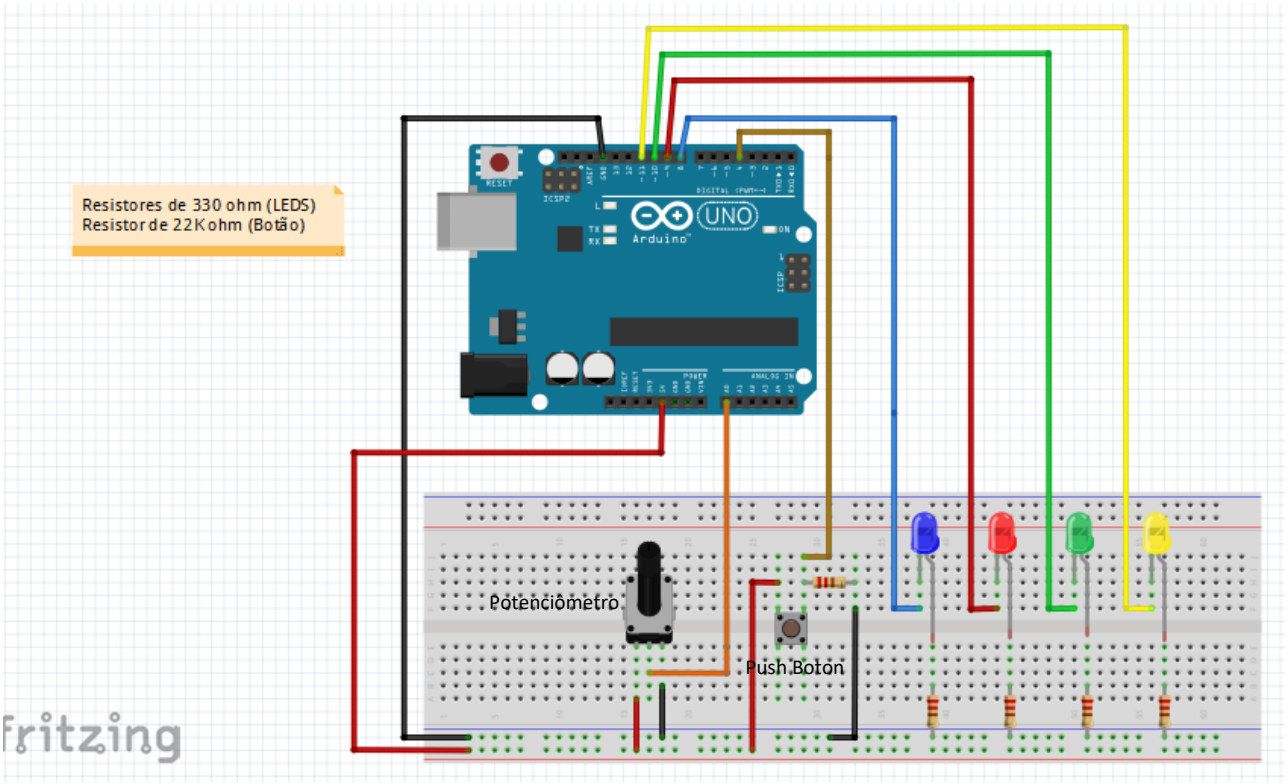
2 O HARDWARE

O *hardware* consiste de um circuito com um botão, resistores, LEDs e potenciômetro montados em uma placa de montagem (*protoboard*).

A função do *hardware* é fazer a aquisição de dados digitais e analógicos via um protocolo de comunicação industrial, o modbus RTU. Optou-se pelo modbus devido a maior facilidade de implementação para comunicar com *softwares* SCADA (Supervisory Control And Data Acquisition) e IHM comerciais físicas ou simuladas.

O *hardware* proposto é conforme mostra a Figura 1.

Figura 1 – Circuito para aquisição de dados e acionamentos



Fonte: Dos autores

Como mostra a Figura 1 o *hardware* contém componentes que permitem realizar:

- Aquisição de sinal analógico na faixa de 0 a 5V fornecida pelo potenciômetro;
- Aquisição de de sinal digital 0 ou 5V fornecida pelo *push boton*;
- Acionamento de digital de carga (LEDs Azul, Vermelho e Verde);
- Controle de carga com sinal PWM (LED amarelo).

O protocolo de comunicação industrial utilizado foi o Modbus RTU. Modbus é muito popular em ambientes industriais sendo aberto e livre de royalties. Modbus foi desenvolvido pela Modicon em 1979 para uso com seus Controladores Lógicos Programáveis (CLPs) mas se tornou um protocolo de comunicação padrão de fato sendo um meio comumente utilizado para conectar dispositivos eletrônicos industriais. Modbus é relativamente fácil de implantar e manter em comparação com outros padrões. Modbus é mantido pela organização modbus (modbus organization, 2022) e as especificações técnicas pode ser encontrada em (Modbus Application,2022).

Uma exigência para a comunicação neste projeto é que esta deve funcionar tanto no sistema SCADA para o desenvolvimento das telas do supervisório (Elipse SCADA), bem como o *software* EasyBuilder Pro (EasyBuilder Pro) usado para desenvolver as telas da IHM. A biblioteca Modbus RTU utilizada foi o Mobbusino porque esta foi a única que se conseguiu comunicar com ambos softwares acima citados. Modbusino (Raimbault S. 2022) é uma biblioteca do tipo Modbus escravo simplificada para Arduino mas é capaz de ler registros de retenção (*holding resgisters*) e escrever múltiplos registros atendendo plenamente o propósito deste trabalho.

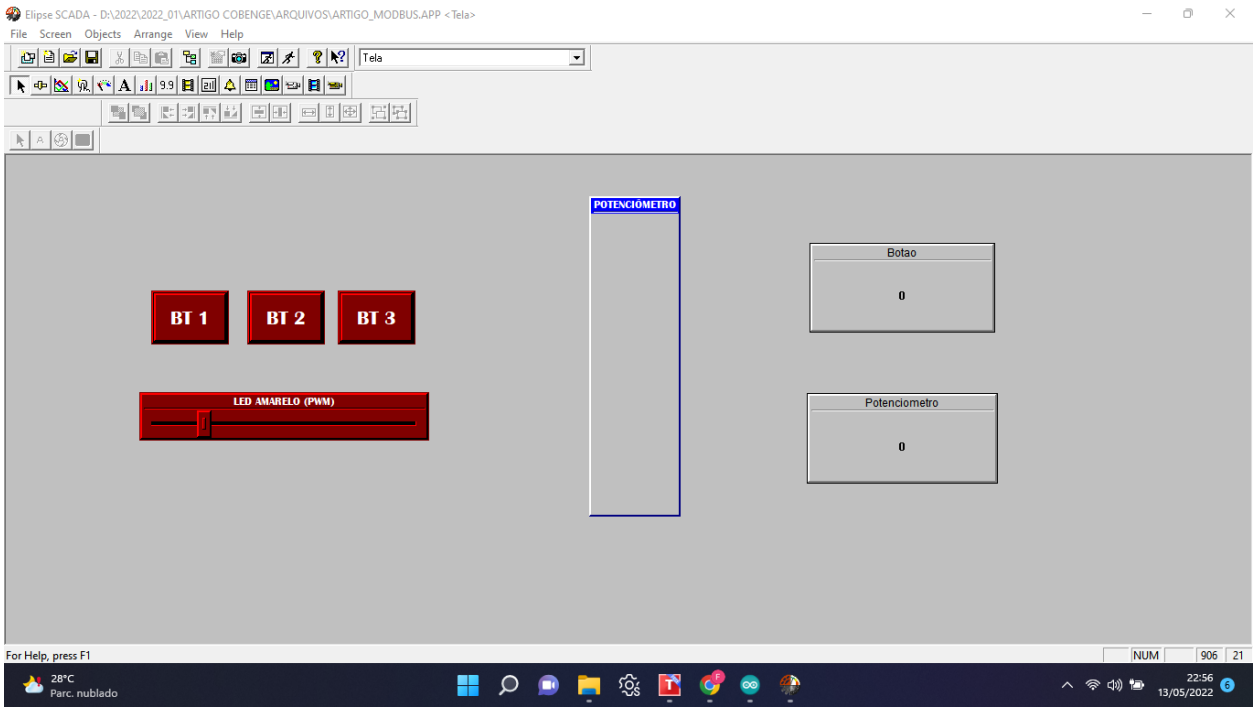
3 O SOFTWARE

O software pode ser apresentado em três etapas a saber: Supervisório Elipse SCADA, Arduino e EasyBuilder Pro para IHM.

3.1 Supervisório Elipse SCADA

O sistema SCADA para o desenvolvimento das telas de supervisão foi o Elipse SCADA (Elipse, 2022). A tela principal desenvolvida é como mostra a Figura 2

Figura 2 – Tela do supervisório

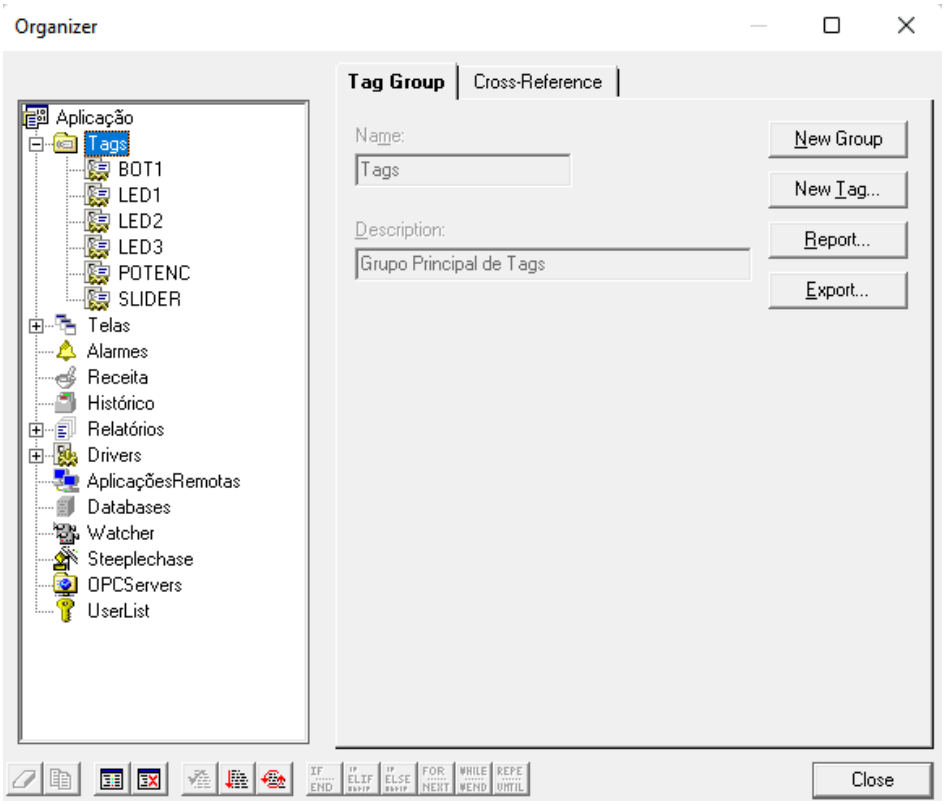


Fonte: Dos autores

Como se pode observar na Figura 2 existem três objetos do tipo botão nomeadamente BT1, BT2 e BT3 que são utilizados para acionar os LEDs azul, vermelho e verde no *hardware* mostrado na Figura 1. O objeto do tipo *slider* LED AMARELO (PWM) é utilizado para variar a luminosidade do LED amarelo. Dois *displays* foram utilizados, um denominado Botão que apresenta o valor 1 se o botão foi pressionado ou 0 em caso contrário, outro display denominado Potenciômetro que mostra o valor da tensão na entrada analógica A0 do Arduino no *hardware* da Figura 1. Também se pode observar na Figura 2 um objeto do tipo gráfico de barras denominado POTENCIOMETRO que mostra na forma de barra o valor da tensão analógica na entrada analógica A0.

Para os objetos foram criados e associadas Tags como mostra a Figura 3.

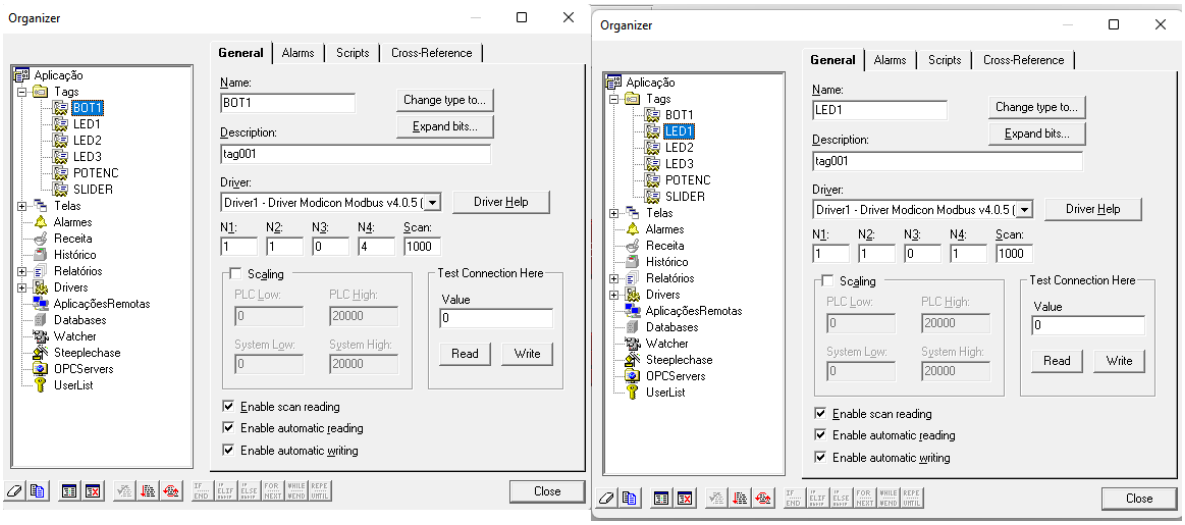
Figura 3 – TAGs



Fonte: Dos autores

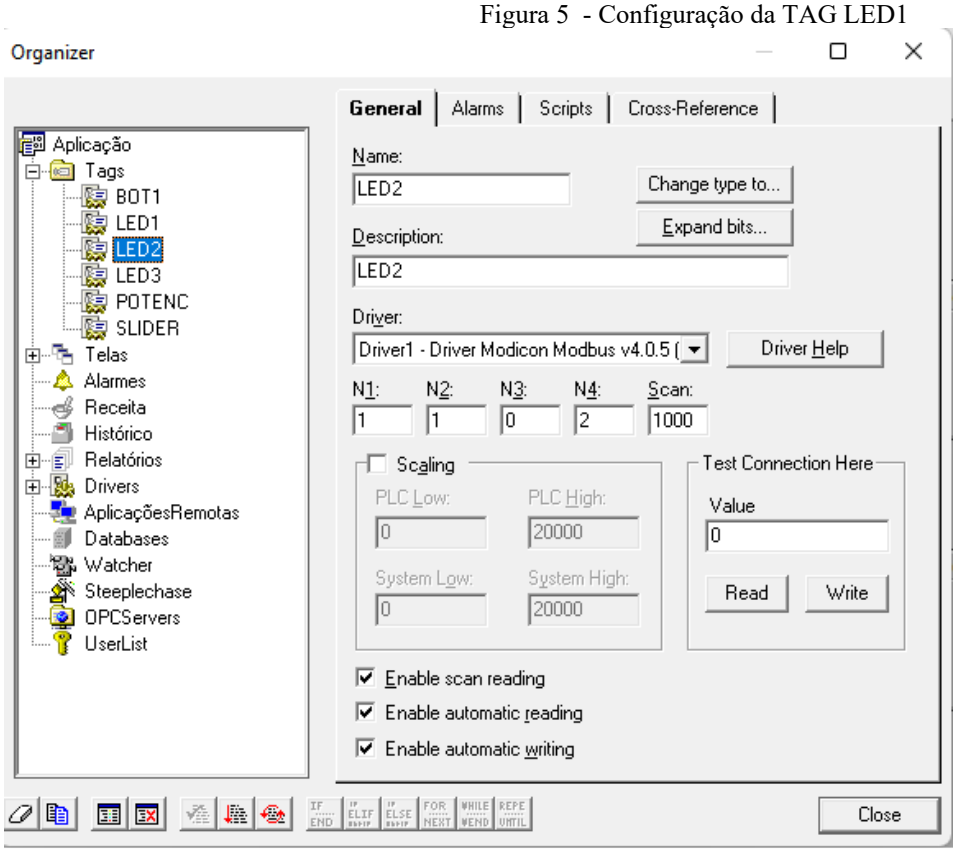
As TAGs são todas do tipo PLC e estão associadas ao *driver* Modbus e configuradas da forma a seguir. A Figura 4 mostra as configurações para as tags BOT1 e LED1, enquanto que a Figura 5 mostra a configuração para a tag LED2.

Figura 4 - Configurações das TAGs BOT1 e LED1



Fonte: Dos autores

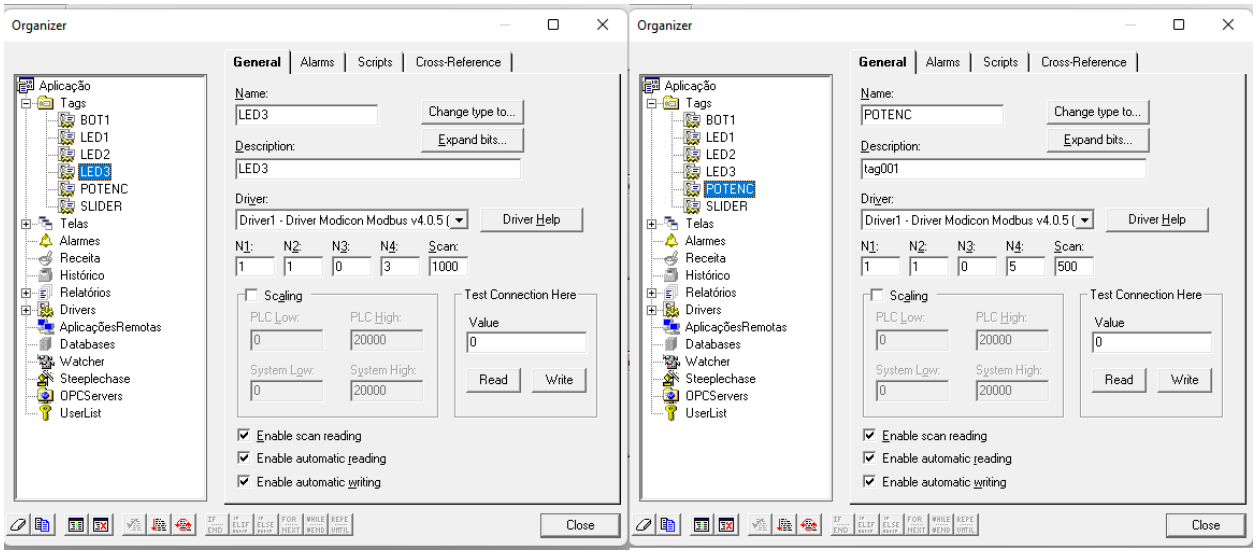
Figura 5 - Configuração da TAG LED1



Fonte: Dos autores

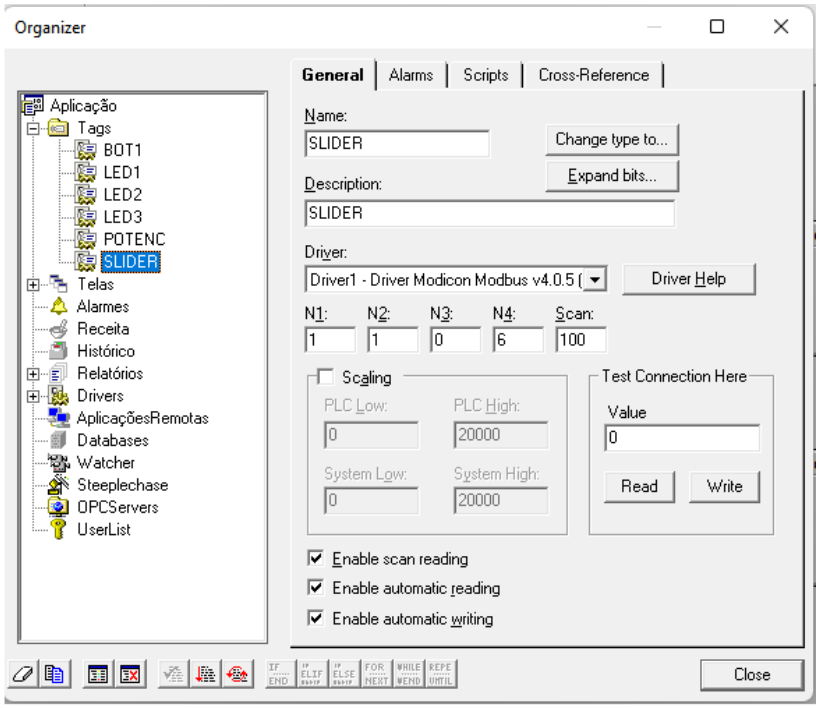
A Figura 6 mostra as configurações para as TAGs LED3 e POTENC e a Figura 7 mostra a configuração para a tag SLIDER.

Figura 6 - Configurações TAGs LED3 e POTENC



Fonte: Dos autores

Figura 7- Configuração TAG SLIDER



Fonte: Dos autores

A Tabela 1 resume as configurações utilizadas nos parâmetros N1, N2, N3 e N4 e SCAN aqui ressaltadas porque contribuem para o entendimento do *software* no Arduino.

Tabela 1 - Resumo das configurações das TAGs

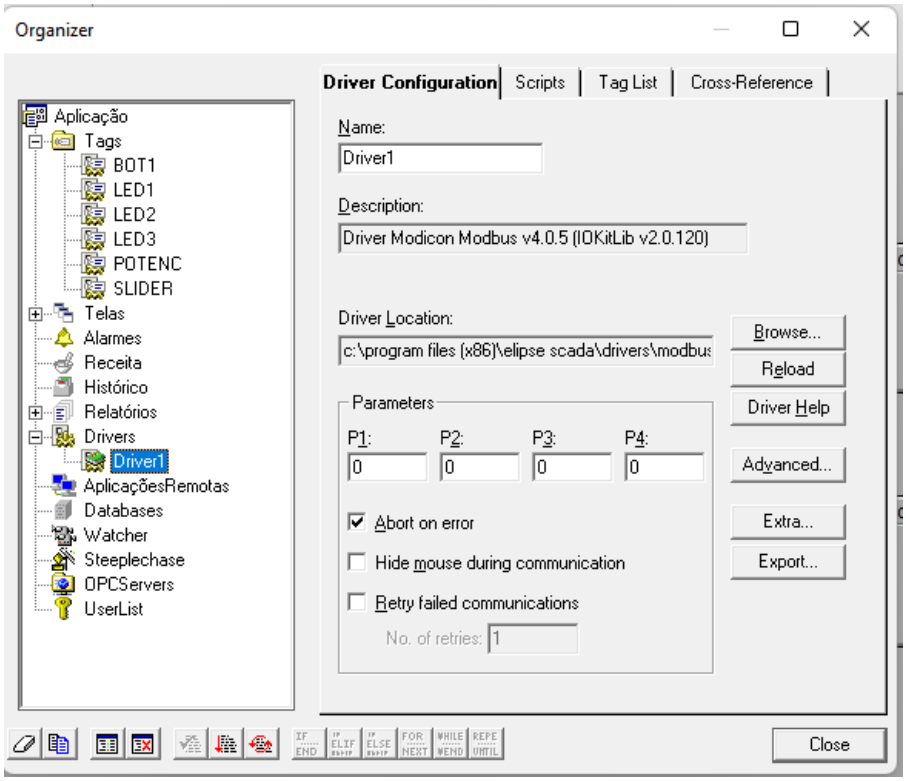
TAG	N1	N2	N3	N4	SCAN(ms)
BOT1	1	1	0	4	1000
LED1	1	1	0	1	1000
LED2	1	1	0	2	1000
LED3	1	1	0	3	1000
POTENC	1	1	0	5	500
SLIDER	1	1	0	6	100

Fonte: Dos autores

De acordo com o manual do driver Driver Modicon Modbus N1 é o número do escravo, N2 é o número da operação configurada no *driver*, N3 não é usado deixar em 0 (zero), N4 é o endereço do registro Modbus ou bit, e SCAN é o período em milissegundo que o elipse faz a a leitura das TAGS. O Elipse SCADA é o metre, o Arduino é o escravo e tem endereço 1.

O driver Modbus, assim como outros, pode ser baixado gratuitamente do site Elise software e instalado. A configuração do driver se faz inicialmente como mostra a Figura 8.

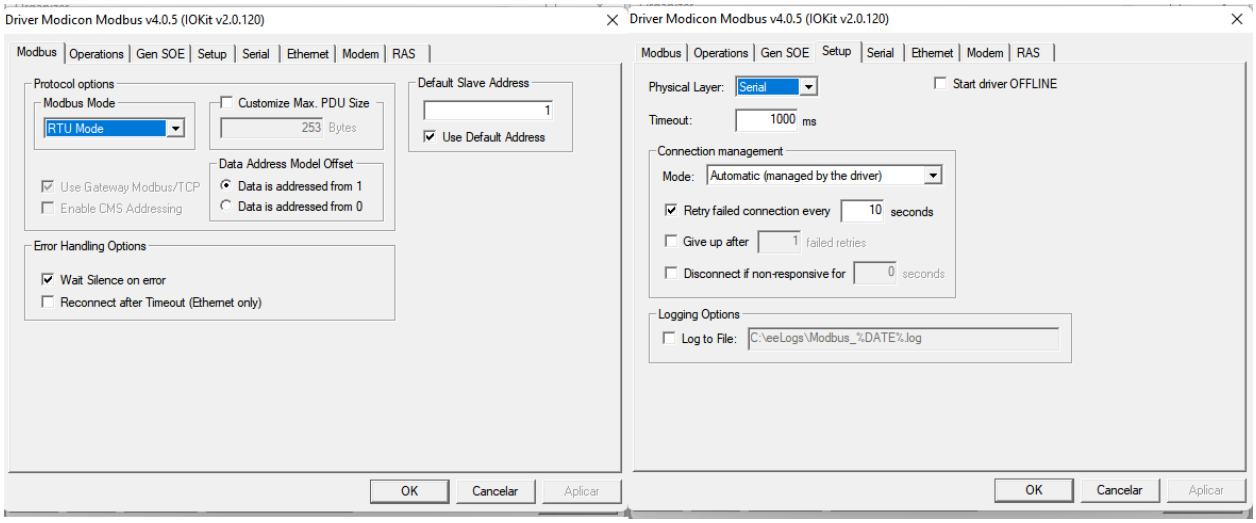
Figura 8 - Configuração driver Modbus



Fonte: Dos autores

Em Extra se tem acesso ao restante da configuração como mostra as Figura 9 e 10.

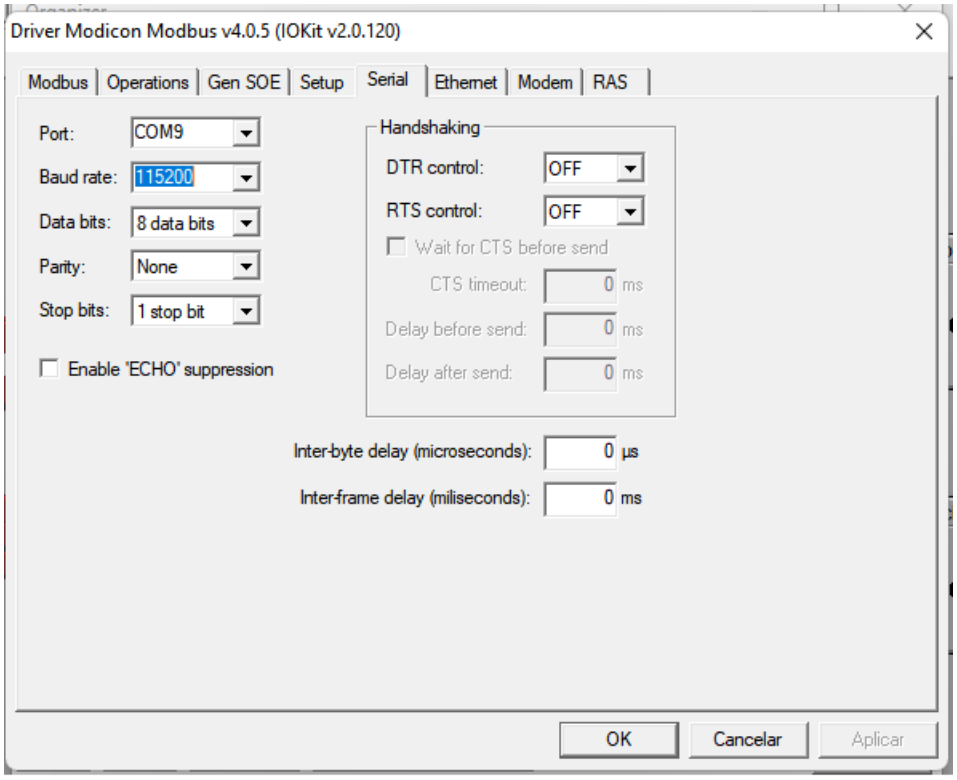
Figura 9 - Configurações adicionais do driver Modbus



Fonte: Dos autores

Na aba Modbus foi escolhido o Modbus RTU deixou-se offset de 1 e na aba setup foi selecionado e comunicação serial. A Figura 10 mostra a configuração para a porta serial utilizada pelo Arduino e suas configurações de baud rate, dados, paridade e stop bit.

Figura 10 - Configurações da porta serial



3.2 Código no Arduino

O código da aplicação desenvolvida no Arduino é simples e está todo comentado como pode ser visualizado e descrito a seguir.

```
#include <Modbusino.h> // Inclue a biblioteca Modbusino
#define LED1 8   // Define o pino para o led Azul
#define LED2 9   // Define o pino para o led Vermelho
#define LED3 10 // Define o pino para o led Verde
#define LED4 11 // Define o pino para o led  Amarelo (PWM)
#define B1 4       // Define o pino para o botão
#define P1 A0     // Define o pino para o potenciômetro

// Inicializa o escravo modbus com ID 01
ModbusinoSlave modbusino_slave(1);

uint16_t tab_reg[6];  // Aloca 6 Holding Registers

void setup() {
   // Configura baud rate de 115200
   modbusino_slave.setup(115200);

   // Configura pino de entrada
   pinMode(B1, INPUT);

   // Configura pino  pinos como saída
   pinMode(LED1, OUTPUT);
   pinMode(LED2, OUTPUT);
```

```
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
}
void loop() {

tab_reg[0] ? digitalWrite(LED1, HIGH):digitalWrite(LED1, LOW); // Acionamentodo LED AZUL
tab_reg[1] ? digitalWrite(LED2, HIGH):digitalWrite(LED2, LOW);//Acinamento do LED VERMLEHO
tab_reg[2] ? digitalWrite(LED3, HIGH) : digitalWrite(LED3, LOW); // Acionamento do LED VERDE
tab_reg[3] = digitalRead(B1) ? 1 : 0; // Atribue o estado do botão a tab_reg[3]
tab_reg[4] = (word)analogRead(P1);//Atribueo valor de leitura do potenciômetro a tab_reg[4]
analogWrite(LED4, tab_reg[5]); //Aciona o LED AMARELO com sinal PWM vindo do slider
modbusino_slave.loop(tab_reg, 6); // Habilita o loop de leitura dos registros modbus
delay(1); // Delay de 1ms
}
```

O código é fácil de ser compreendido e está devidamente comentado dispensando maior detalhamento. Vale ressaltar que foi utilizado o *offset* de 1 na aba modbus referente à configuração do driver (ver Figura 9), assim tab_reg[0] por exemplo, utiliza o valor 1 no parâmetro N4 na tag LED1 em vez de 0. Este mesmo procedimento vale para outras tags devido ao offset de 1.

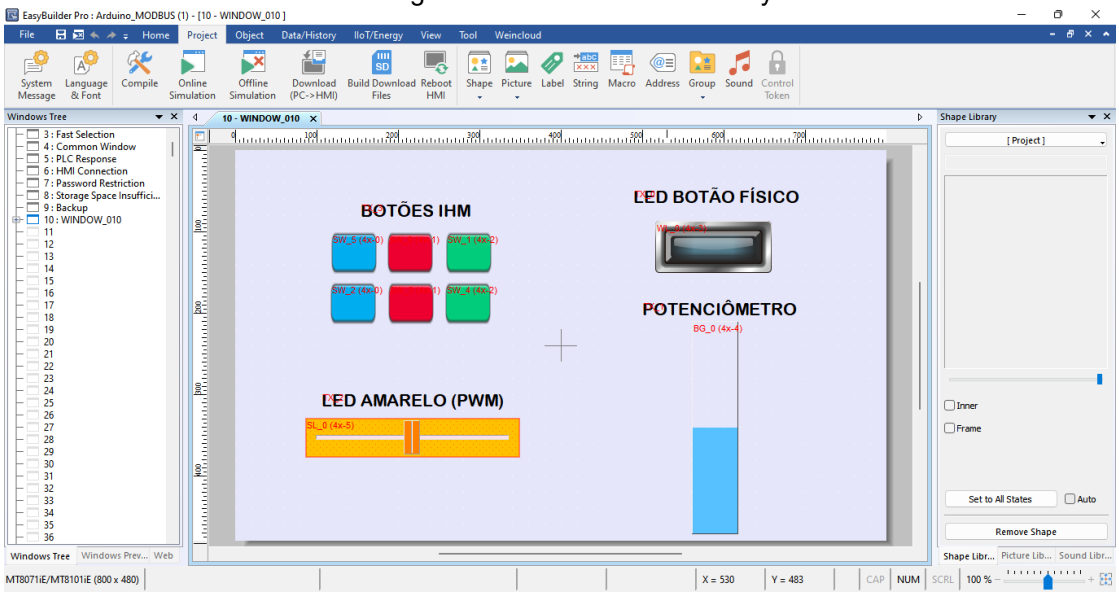
3.3 O software no EasyBuilder Pro

O Easybuider Pro é desenvolvido pela Weintek e pode ser baixado também gratuitamente. No Brasil a WEG utiliza este *software* para programação de suas linha comercial de IHM. Easybuider Pro possui um simulador que permite criar telas e comunicá-las, via diversos protocolos industriais, com *hardware* como o mostrado na Figura 1 utilizando o protocolo Modbus. O Easybuilder Pro permite usar o computador como uma IHM e testar o funcionamento das telas sem uma IHM real. Assim foi possível utilizar este potente *software* para o ensino de IHM de forma eficiente e sem custo algum. Outra vantagem de se utilizar a simulação é que uma IHM comercial é um item muito caro e pode chegar a mais de 30 mil reais o que é inacessível para estudantes e até mesmo para universidades públicas e escolas técnicas.

A criação de telas segue as etapas de desenvolvimento como criar projeto, selecionar e configurar objetos gráficos, definir interfaces de comunicação e a simulação. São etapas que exigiriam muitos passos e não seria diatático fazê-las por este meio. Existem documentações como por exemplo em (EasyBuilder Pro docs) sobre manuais de utilização, canal sobre programação de IHM (PLC & Drivers), bem como um canal da própria Weintek (Weintek USA, Inc) com um video sobre Conexão básica Arduino + HMI via protocolo Modbus mas que usa biblioteca Modbus diferente da utilizada neste projeto.

A tela principal da IHM desenvolvida é como mostra a Figura 11.

Figura 11 - Tela da IHM no EasyBuilder Pro

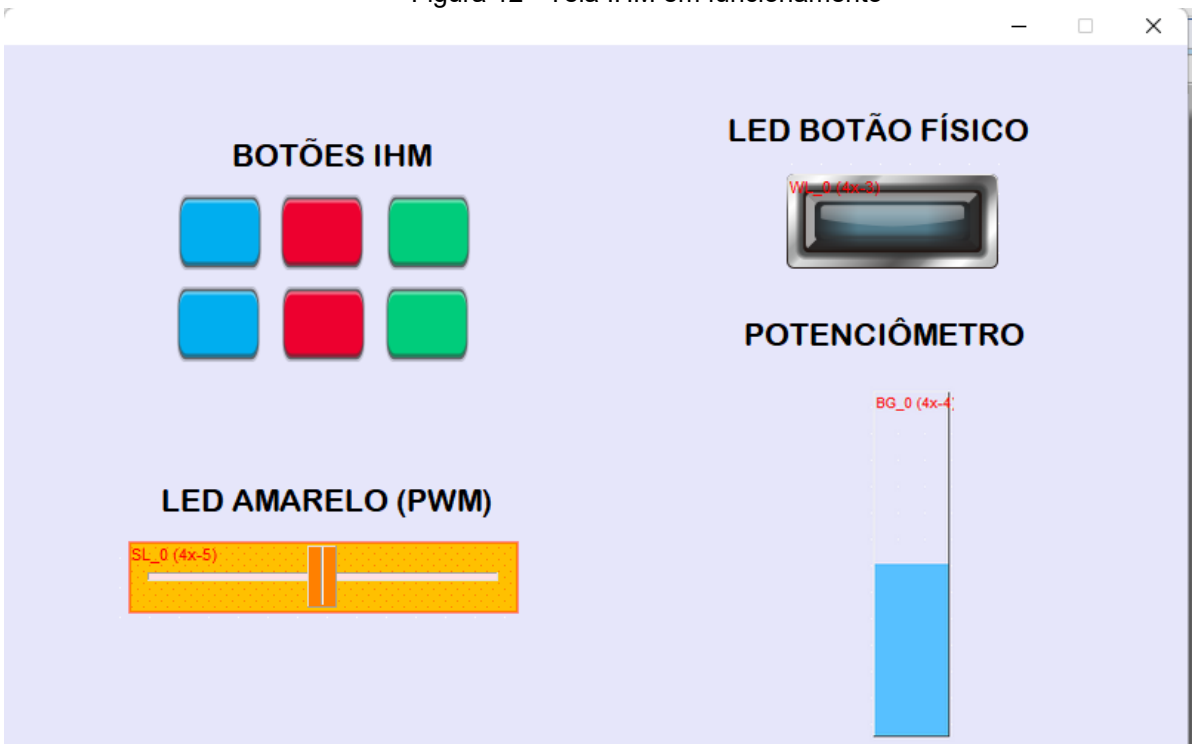


Fonte: Dos autores

Como se pode observar na Figura 11 há muita semelhança entre a tela desenvolvida no Elipse SCADA e no EasyBuilder Pro, assim feitas de propósito para que utilizassem o mesmo *hardware* e critérios de desenvolvimento semelhantes. Foram colocados seis botões para os LEDs, sendo dois por LED um para ligar e outro desligar o mesmo LED visto que não foi possível configurar a função *togle* usando um só botão. Existem um objeto LED BOTÃO FÍSICO que muda de cor na IHM quando o botão físico é pressionado no *hardware*. Também há um gráfico de barras com o nome POTENCIOMETRO que exibe o valor do sinal de tensão analógico na entrada A0 do Arduino e por último um objeto do tipo *slider* com o nome LED AMARELO (PWM) que envia um sinal de PWM via o protocolo Modebus e que é aplicado ao LED amarelo no hardware da Figura 1 para controlar a intensidade do LED.

A tela de IHM em funcionamento é como mostra a Figura 12.

Figura 12 - Tela IHM em funcionamento



Fonte: Dos autores

A Figura 12 mostra como fica a IHM simulada apenas para mostrar a semelhança com a tela de uma IHM real visto que não seria possível mostrar numa figura só as mudanças de estados dos objetos.

4 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento e os resultados obtidos para o ensino de disciplina práticas como Sistemas Supervisórios e IHM no Instituto Federal do Maranhão. Como resultados pudemos mostrar resultados da interação entre hardware e telas de um sistema supervisório e IHM com recursos físicos normalmente disponíveis com os alunos e software gratuitos como Arduino e versão demo (Elipse SCADA) nas aulas *online*. Também foi realizada a montagem e testes por diversos alunos utilizando seu próprio hardware seguindo os procedimentos aqui descritos. Isto possibilitou um grau muito maior na capacidade de desenvolvimento nas habilidades de montagens eletrônicas e de programação bem como alto grau de satisfação por tarte dos alunos que foram muito prejudicados pela impossibilidade de acessar aos laboratórios devido às medidas restritivas por causa da pandemia de Covid19. Como resultado final e muito importante foi observar o grau de satisfação esboçado por diversos alunos em face à realização de suas próprias montagens.

REFERÊNCIAS

EasyBuilder Pro. Disponível em https://www.weintek.com/globalw/Download/Download.aspx. Consultado em em 13 de Maio de 2022.

Elipse SCADA. Elispse Software. Disponível em www.elipse.com.br. Consultado em em 13 de Maio de 2022.

modbus organization, 2022. Disponível em https://modbus.org/ consultado em 13 de Maio de 2022.

Modbus Application,2022. MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b. Disponível em https://modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. consultado em 13 de Maio de 2022.

PLC & Drivers. Disponível em https://www.youtube.com/channel/UCROztAbrU6i836LV7BLMZFQ.Consultado em em 13 de Maio de 2022.
Raimbault S., 2022. Disponível em https://github.com/kmihaylov/modbusino. Consultado em em 13 de Maio de 2022.

TEACHING SCADA SYSTEMS, HMI, AND MODBUS PROTOCOL USING ARDUINO

Abstract: *This work presented the proposal and the development of hardware and software for the teaching of subjects such as SCADA systems and HMI (Human Machine Interface) that are widely used in technical schools and universities. The hardware consists of an Arduino Uno, push button, LEDs, and potentiometer. The software corresponding to the screens were developed using software in the demo version of Elipse SCADA for the supervisory system and EasyBuilder Pro for the HMI, while for Arduino programming, the free Arduino IDE and the Modbusino library were used, also free. It was possible to develop screens that were used to visualize data in online classes during the COVID19 pandemic, as well as to enable students to develop their own hardware and supervision screens for both SCADA and HMI systems. The results were surprising because they made possible the practical teaching of subjects that normally can only be taught in the laboratory, in addition to presenting a high degree of satisfaction in learning these subjects.*
Keywords: *scada, hmi, modbus, Arduino Uno*