PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITÉCNICA

Trabalho Final de Curso II

Julio Cesar de Freitas Galvão

SISTEMA SUPERVISÓRIO IMPLEMENTADO COM IOT NA INDÚSTRIA UTILIZANDO ESP8266

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

BANCA EXAXMINADORA:

Prof. Dr. Antônio Marcos de Melo Medeiros – Orientador. Pontifícia Universidade Católica de Goiás.

Prof. Dr. Bruno Quirino de Oliveira – Banca Examinadora. Pontifícia Universidade Católica de Goiás.

Prof. Ma. Fabricia Neres Borges – Banca Examinadora. Pontifícia Universidade Católica de Goiás.

Goiânia, 10 de dezembro de 2022.

SISTEMA SUPERVISÓRIO IMPLEMENTADO COM IOT NA INDÚSTRIA UTILIZANDO ESP8266

Julio Cesar de Freitas Galvão, Antônio Marcos de Melo Medeiros, Bruno Quirino de Oliveira, Fabricia Neres Borges

Resumo **- A função deste projeto, é desenvolver um protótipo capaz de comunicar via internet dados entre o sistema supervisório e um controlador, como por exemplo um CLP ou inversor de frequência, aplicando assim o conceito de internet das coisas (Iot) na indústria. O protótipo em questão utiliza o micro controlador ESP8266 que fará essa interface entre o supervisório e o inversor, uma vez que o micro-controlador é capaz de gerar e conectar à internet, sendo conectado então a um inversor para este também ter acesso a rede também.**

Palavras-Chave **– ESP8266, Iot(Internet das coisas), inversor de frequência, supervisório, WI-FI.**

SUPERVISORY SYSTEM IMPLEMENTED WITH IOT IN INDUSTRY USING ESP8266

Abstract **- The function of this project is to develop a prototype capable of communicating internet life data between the supervisory system and a controller, such as a PLC or frequency inverter, thus applying the concept of internet of things (Iot) in the industry. The prototype in question uses the ESP8266 micro controller that will make this bridge between the supervisory and the inverter, since the micro controller is able to generate and connect to the internet, being then connected to an inverter so that it also has access to the network as well.**

Keywords **– ESP8266, Iot(Internet fo things), frequency inverter, supervisory and WI-FI.**

I. INTRODUÇÃO

Uma empresa seguindo as tendências da indústria 4.0 está totalmente ligada a geração e captação de dados, assim como a conectividade de todos os equipamentos presentes, de forma a se auto ajustar quando necessário, além de fornecer ferramentas de análise e implementação

Os objetivos deste projeto é impulsionar a implementação da indústria 4.0 trazendo conectividade para equipamentos legados e permitindo comunicação com a internet, reduzir custos na indústria dando uma maior vida útil aos equipamentos e reduzindo a quantidade de cabeamento e oferecer uma plataforma familiar ao meio industrial, tanto em meios físicos como na utilização dos protocolos de comunicação.

Uma vez que está sendo neste trabalho desenvolvido este protótipo, então será utilizado apenas um inversor de frequência ligado diretamente ao ESP. Uma vez que tanto o inversor como o CLP possuem comunicação RS485, permitindo a comunicação *master* e *slave*.

II. FUNDAMENTAÇÃO TEÓRICA

Para a implementação de qualquer indústria é necessário haver comunicação entre os equipamentos, ou seja, eles precisam “falar a mesma língua” e para isso foi criado os protocolos de comunicação, cada um contendo suas características e particularidades para o uso. Entretanto, eles também precisam e um meio físico para se comunicarem, isso irá determinar a a quantidade de cabos e quais serão necessários.

Já a indústria 4.0, possuem pilares para sua concepção, e eles são internet das coisas (Iot), big data, computação em nuvem, segurança cibernética, robôs, simulação, integração de sistemas, internet industrial, realidade aumentada, entre outras tecnologias conforme apresentado na imagem 1.

Figura 1: Pilares da indústria 4.0

A Internet das Coisas é uma rede global de objetos interligados por meio de endereços IP que permitem a coleta e troca de dados, com arquiteturas de recursos limitados, mas que aumentam o desempenho da rede de acordo com as aplicações e usuários. Considerando a heterogeneidade dos dispositivos e o tipo de informação a evolução da internet está voltada para a conectividade de ponta a ponta entre máquinas.

Comunicação RS485 - O padrão de comunicação RS-485 é um meio físico, ou seja, é um arranjo de circuito físico por

onde os dados trafegam entre os dispositivos industriais. Diversos protocolos utilizam esse padrão/meio físico para comunicação. Entre eles, destacamos o Protocolo Modbus. Esse padrão permite a comunicação com controladores lógicos programáveis em redes industriais, transmita dados por fios em até 1200 metros com até 32 dispositivos. [1]

Protocolo Modbus - Modbus é um protocolo de comunicação serial desenvolvido pela Modicon e publicado pela Modicon em 1979 para utilização em controladores lógicos programáveis (CLPs). Em termos simples, é um método usado para transmitir informação sobre redes seriais entre dispositivos eletrônicos, podendo ter um Master e até 247 Slaves. [2]

Comunicação RS232 - RS-232 (também conhecido por EIA RS-232C ou V. 24) é um padrão de protocolo para troca série de dados binários entre um DTE (terminal de dados, de Data Terminal equipment) e um DCE (comunicador de dados, de Data Communication equipment). São comumente usados nas portas seriais dos PCs.

Protocolo ASCII - O código ASCII serve para representar textos em computadores e equipamentos de comunicação. O ASCII tem esse papel, por não existir um método que armazene diretamente os dados na memória em forma de caracteres (letras ou textos). Por isso, cada caractere possui o seu equivalente em código numérico que é o ASCII. [3]

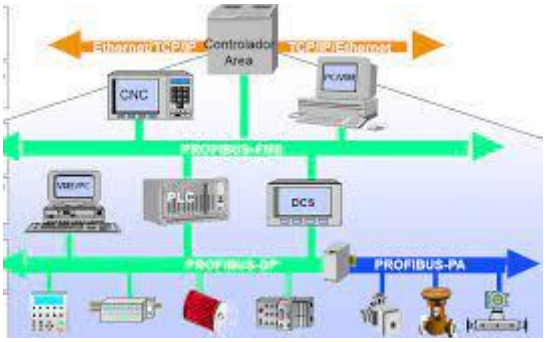
III. MATERIAIS E MÉTODOS

A visão de comunicar tudo em tempo real, é possível através da incorporação de inteligência em dispositivos eletrônicos e tomar decisões, trocar informações, controlar processos, invocar ações e habilitar serviços em tempo real. O ESP8266 faz exatamente isso, usando conexão via wifi com uma banda de 2.4GHz e controlando ou apenas comunicando com o maquinário industrial que ainda tenha somente conexão cabeada.

Dentre as bases em questão citadas anteriormente, as três que serão aprofundadas e desenvolvidas para a realização do protótipo de sistema supervisório implementado com Iot na indústria será o próprio Iot, Big data e Processamento em nuvem.

O atual modelo de controle de uma planta industrial consiste em um sistema supervisório ligado diretamente ao controlador lógico programável (CLP) que por sua vez se conecta a inversores. Toda a comunicação é feita através de cabos, entretanto acarretam altos custos para fábricas de grande porte, sendo assim a sua implementação se torna inviável. Na figura a seguir está de forma genérica o caminho da informação em uma indústria.

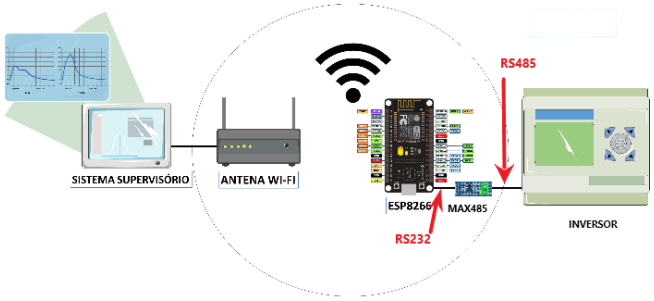
Figura 2: Ilustração simplificada do caminho percorrido pelas informações em uma indústria comum



O sistema proposto neste estudo, promover a comunicação ethernet dos sistemas supervisório com a rede de controle, uma vez que essa conexão é feita de forma cabeada em sistemas legados, então para se enquadrar na indústria 4.0 é necessário haver a conexão e comunicação desses sistemas por meio do wifi entre o sistema supervisório e os controladores.

Na figura 3 mostra uma ilustração de como essa comunicação será feita.

Figura 3: Comunicação remota entre o sistema supervisório e o CLP, sendo feita através de antenas controladas por ESP8266



Essa comunicação remota será feita com o micro controlador ESP8266 que possui WI-FI integrado, em relação ao Arduino, a instalação é acessível comparado aos demais e de fácil programação, uma vez que é possível usar o mesmo compilador do Arduino visto que o mesmo é mais difundido.

Como mostrado anteriormente na figura 3, o protótipo irá se desenvolver em 4 bases que são, supervisório, servidor na rede, comunicação do ESP e comunicação com inversor.

Para se iniciar a programação no ESP8266, é necessário primeiro entender sobre todos os protocolos e linguagens de comunicação utilizados. A começar com o inversor de frequência, que utiliza o meio físico RS485 e o protocolo MODBUS para realizar os comandos. O ESP por sua vez, irá usar o meio físico RS232 e o protocolo usado nesse meio é o ASCII. Já na página web será usado a linguagem HTML para a construção do website e inserção de comandos, porém os comandos serão escritos em ASCII e quando enviados para o inversor passarão por um conversor antes que converterá tudo em MODBUS.

A vantagem de usar MODBUS é por que possui velocidades de transmissão de 30-35 Mbps para distancias de até 10 metros, e 100 kbps para distancias de até 1200 metros, feitas em cabo de par trancado para evitar interferências no

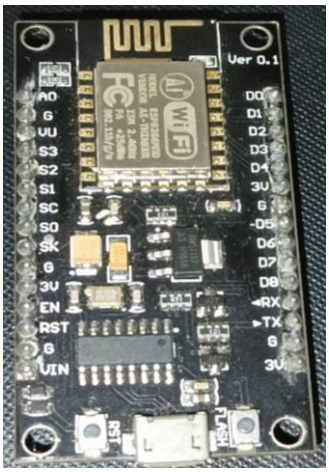
sinal transmitido, tornando seu uso na indústria algo indispensável. [4]

A. ESP8266-12E

O ESP8266 é um micro controlador com microchip WI-FI embutido, produzido pela Espressif Systems em Shanghai, China. Sua popularidade surgiu devido as suas características e principalmente ao seu baixo custo, obtendo um ótimo custo-benefício. A sua programação utiliza a mesma linguagem LUA e C#, inclusive é possível utilizar também o Arduino IDE, tornando seu uso além de barato, prático com todas essas compatibilidades. [5]

A seguir a figura 4 mostra a placa que será usada em detalhes as funções de cada pino. A pinagem da placa possui algumas diferenças em relação ao arduino, em especial o número da porta não representa o número do GPIO.

Figura 4: Imagem real da placa eletrônica com o modulo ESP 12E



Algumas das especificações da placa ESP 12E são as seguintes:

- Wireless padrão de 802.11 b/g/n;
- Antena embutida;
- Entradas e saídas de 3,3V;
- Frequência de 2.4 GHz;

Todas essas informações são capazes de serem vistas diretamente no site da Espressif (https://www.espressif.com/en).

B. Inversor de frequência CFW500(WEG)

Os inversores de frequência são equipamentos que facilitam o controle de motores elétricos, uma vez que os inversores podem ser programados usando diferentes protocolos de comunicação, além de converterem um sinal senoidal de amplitude fixa em um sinal modulado por largura de pulso (PWM), para ter um controle mais assertivo da rotação do motor. [6]

O inversor de modelo CFW500 da marca WEG possui uma faixa de potências de 0,25 a 175 cv, corrente de saída de 1,0 a 211 A, tensão de alimentação monofásica ou trifásica, porta RS485 incorporada e diversas outras características, porém as mais importantes para esse projeto são somente essas. [7]

Figura 5 - Inversor de frequência CFW500

C. Elipse E3

O sistema supervisório usado é o Elipse E3 Studio, uma ferramenta SCADA que será usada para fazer o monitoramento das atividades que serão executas no inversor de frequência.

O Elipse E3 é uma plataforma SCADA, que oferece escalabilidade e evolução constante para diversos tipos de aplicações, desde simples interfaces IHM até complexos centros operacionais em tempo real. Desenvolvido para atender aos requisitos de conectividade atuais e futuros, ideal para atender projeto, não importa a extensão ou necessidades [8]

D. Conversores

Foram adquiridos conversores para a realização do protótipo, uma vez que a comunicação com o inversor será feita utilizando o protocolo MODBUS e transmissão pela rede RS485, entretanto o ESP se comunicará utilizando RS232.

O passo seguinte é a conexão do ESP com o inversor, para isso é necessário um conversor RS232 para RS485, foi utilizado dois conversores, que atendem a mesma proposta, porém o Max485 (HW-97) é mais dedicado para usar com Arduino por usar tensão 5V na entrada, enquanto o conversor rs485 para ttl (XY-017) é mais usado no ESP, possuindo uma tensão de entrada de 3,3V.

Figura 6: Conversor RS485 para RS232 (HW-97)

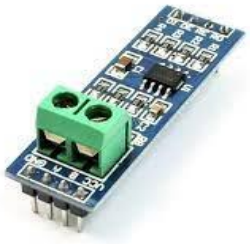
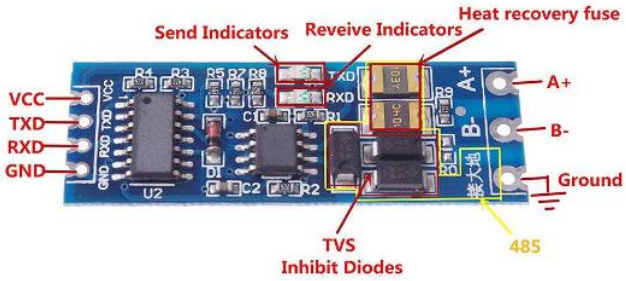


Figura 7 - Módulo Conversor Rs485 para Ttl (XY-017)



Com isso, a ligação que cada conversor faria com o ESP8266 estão demostrados nas figuras 8 e 9 a seguir respectivamente:

Figura 8 - Ligação ESP com conversor MAX485 (HW-97)

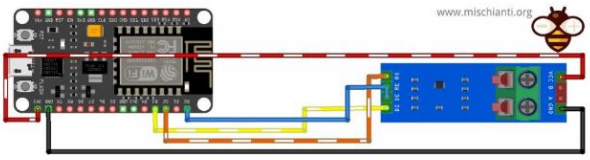
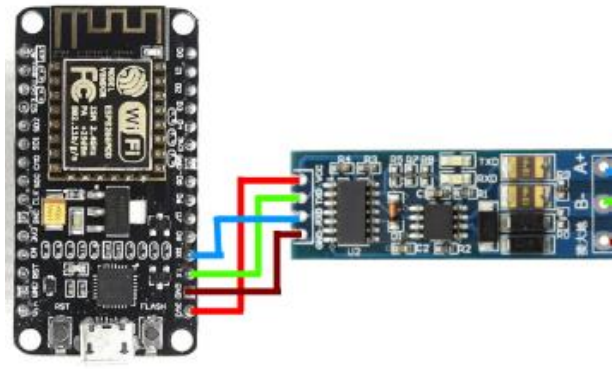


Figura 9 - Ligação do ESP com o conversor rs485 para ttl (XY-017)



Dessa forma, o ESP8266 que terá a função de mestre, criando a página web. Página essa que será programada em HTML e terá botões de controle que enviaram sinal seguindo o protocolo ASCII na qual será convertida para modbus e "entendida" pelo inversor. O inversor por sua vez obedecerá aos comandos e responderá também quando solicitado, enviando código em modbus que será convertido em ASCII e recebido pelo ESP.

IV. DESENVOLVIMENTO

.

Para testar a funcionalidade do servidor, foi criado um pequeno programa que está no apêndice (Programa 1), onde será criado um website para o acionamento do LED presente no próprio ESP. Após a criação dessa página será gerada um endereço de IP que será informado no monitor serial para acessar o site digitando pelo navegador, o que nesse caso, a rede criada é "192.168.0.25".

Em seguida, foi criado um sistema supervisório para comunicar com a página criada e inclusive ser capaz de controlar através do sistema supervisório. Dessa forma, o sistema supervisório se torna uma ferramenta poderosa e o ponto chave na conclusão do projeto, uma vez que todos os dados do sistema já serão direcionados ao sistema supervisório, e agora com a capacidade de além de supervisionar, controlar todo o processo.

Uma das funções já descritas do ESP8266 é de se conectar a uma rede já existente ou gerar a própria rede onde é capaz de acessar a própria página que foi gerada. Dessa forma, para implementar essa função foi utilizado como teste o ESP-01 e utilizado o código que está no apêndice (Programa 2). Esse modelo foi escolhido em especial pela comodidade em já ter as bibliotecas para a compilação do código. Diferente de quando criamos uma página web de uma rede já existente onde o número de IP é gerado de acordo com a rede em que foi criada, essa aplicação irá gerar um número de IP já definido anteriormente na programação.

Para a realização das funções serão necessários parâmetros como frequência de saída do motor, corrente do motor, tensão de saída, aceleração, entre outros. Uma vez que a tabela ASCII manda em hexadecimal os valores que serão identificado pelo inversor.

Como já sabemos qual os códigos ASCII que serão enviados, é necessário criar agora um código em HTML onde terá os botões de controle e ao pressiona-los, enviarão a mensagem em ASCII para o inversor.

A criação do código para ser compilado no ESP8266 é o ponto central deste projeto, porque é através dele que será feito o website, que será feita a comunicação com o inversor de frequência, que será gerado a rede wifi (apesar de também poder usar uma rede já existente) e será o ESP que será usado como mestre ao enviar as informações para os escravos. Tudo isso respeitando as linguagens de programação de cada sistema (C# e HTML) e respeitando também os meios físicos (RS485 e RS232) e protocolos usados em cada um (MODBUS e ASCII).

V. RESULTADOS

Criação da própria rede - Ao compilar do código e colocando em modo de execução, uma nova rede aparece para se conectar.

Ao acessar a rede gerada, basta digitar no navegador p endereço “192.168.4.1” que foi gerada pelo próprio controlador. Encontrando assim a seguinte página na web.

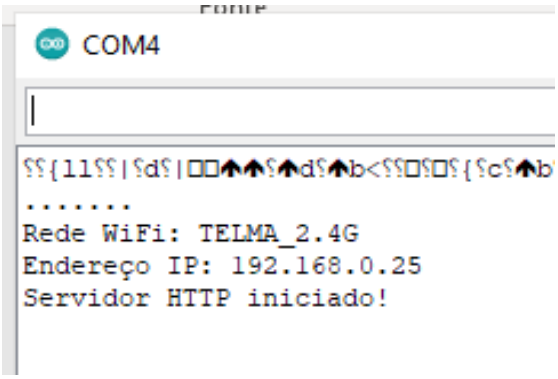
Figura 10 - Página web criada pelo ESP-01

Com isso, a página gerada pelo ESP é capaz de ser acessado tanto por computadores como por smartphones mesmo que estes dispositivos não estejam conectados à internet.

Página - Com a página criada, é possível controlar remotamente de inúmeras formas, tudo baseado na linguagem HTML para gerar os comandos necessários.

Para consultar as informações de rede e o endereço de IP gerado, basta abrir o monitor serial no software arduino e será encontrado as informações de acordo com a imagem a seguir.

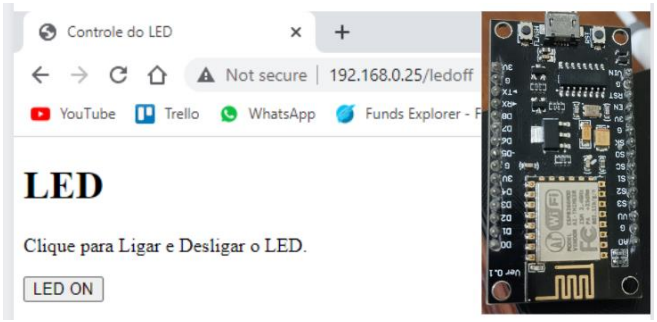
Figura 11: Imagem do servidor criado baseado na rede WIFI conectado



Após digitar o IP no navegador, irá abrir a seguinte pagina para controlar o LED.

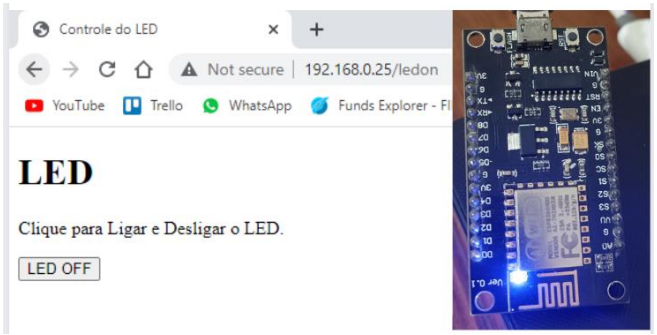
Figura 9: Página web criada pelo ESP

Figura 12: Página criada para controlar o LED do ESP, apresentando apagado



Em seguida, as diferenças na URL quando o sinal de ligar é enviado.

Figura 13: Página enviando informação para acender o LED

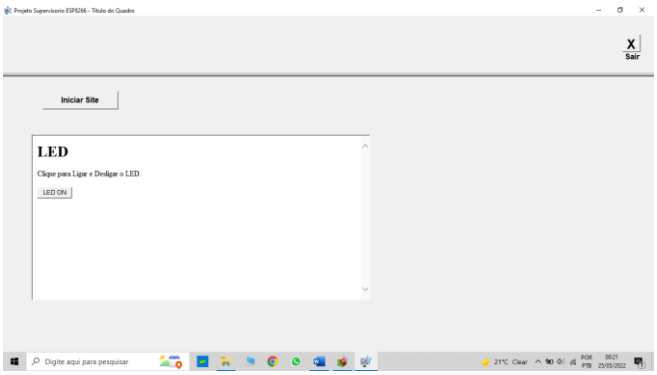


Nas imagens anteriores foi mostrado o funcionamento na prática do servidor utilizado na criação de uma página web no controle o LED do ESP.

Entretanto, não foi possível conectar em redes que possuem mecanismos de autenticação mais sofisticados. Esses comandos em sí são possíveis, entretanto é preciso modificar algumas funções no programa e compila-lo novamente no ESP8266 através do IDE do Arduino.

Supervisório - Foi criado somente uma tela do supervisório, e nela foi utilizada a função “ActiveX” e foi criada uma janela do tamanho escolhido de acordo com a imagem a seguir.

Figura 14: Página web aberta no sistema supervisório



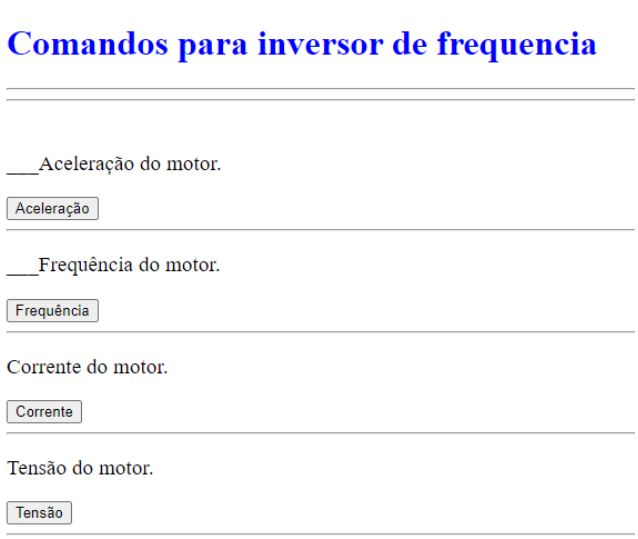
Com isso o sistema criado permite carregar a página que ESP abriu, sendo possível utilizar os comandos gerados pela página e ainda permanecer na tela de supervisão, acompanhando os indicadores e sensores que foram configurados para serem monitorados

ASCII para MOBUS - O parâmetro referente a frequência do motor em ASCII é 0x500x300x300x300x35 já em MODBUS é P0005. Já para mostrar a corrente do motor é enviado o código 0x500x300x300x300x33, que será convertido para P0003. A tensão o motor é o código 0x500x300x300x300x37, que será identificado pelo inversor como P0007. E a aceleração que é dada por 0x500x310x320x35, que é representado por P125.

Código HTML - Para a criação do código, foi usado o bloco de notas porem a sua extensão foi mudada para ".htm", dessa forma ao executar o código ele irá abrir uma página web seguindo a programação descrita.

O código em questão está representado no apêndice (Programa 3), onde é de programado de forma simples o layout da página e alguns botões que poderão ser utilizados. A página criada está representada na figura a seguir.

Figura 15 - Imagem do código HTML criado



Código fonte - Por conta da integração com todos os sistemas do protótipo, essa programação se torna mais complexa. Foi usado o código que está no apêndice (Programa 4), onde o ESP32 funcionará como um gateway, recebendo requisições Modbus e redirecionando a ao slave (server), na rede RS485. Esse código é de autoria do grupo "Drive automação e engenharia" e seria usado como base para o desenvolvimento do protótipo e melhorado para mandar/receber as informações via wifi, tornando o equipamento compatível com o modelo IOT.

Entretanto, dentre os testes realizados ouve problemas com as bibliotecas usadas, em especial com a "eModbus-master" não conseguindo ser compilado para o controlador e consequentemente não sendo possível enviar as informações utilizando o protocolo MODBUS.

VI. CONCLUSÕES

A utilização do ESP8266 para criação do website se mostrou bem eficiente, uma vez que o código já foi compilado no controlador, ele só precisa ser energizado que já consegue se conectar automaticamente da rede e criar o site, demorando apenas alguns segundos para isso. Uma dificuldade foi na utilização de redes que precisam de algum tipo de login, como as redes da PUC GO, porém com mais testes e modificando um pouco o código funcionaria da mesma maneira.

Com a inclusão da página web no sistema supervisório, é possível ter controle das funções na página e ainda configurar o supervisório da forma que melhor atende as necessidades do projeto, uma vez que é possível colocar a tela do website do tamanho desejado, colocando outras informações ao lado se necessário.

Com a criação da própria rede, foi contornado a dificuldade em se conectar em redes fechadas além e ser uma facilidade para usar a comunicação wifi mesmo em lugares que não possuem rede wifi, como lugares mais afastados.

A criação do website foi feita com o intuito de ser aberta no computador em uma tela horizontalmente retangular, principalmente levando em consideração que será aberta na tela do sistema supervisório. Dessa forma, é ainda possível abrir a página utilizando um smartphone, entretanto o layout não foi pensado para abrir em uma tela vertical, ocasionando a desorganização das informações na tela.

Mesmo com testes sendo feitos anteriormente utilizando o código descrito no apêndice (Programa 4), não foi possível colocá-lo em pratica uma vez que por conta da limitação de tempo para a realização do protótipo, a quantidade de teste realizada não foi a necessária para identificar o erro cometido e o conserto necessário.

Dessa forma, a utilização do ESP8266 para controle remoto se mostrou eficiente, uma vez que a criação de um servidor possibilita o fácil controle e acesso as mais diferentes funções desejadas, além de permitir o envio de dados para o monitoramento e controle dos processos. Como aprimorações futuras poderá ter a possibilidade de utilizar aplicativos de

celular para o controle de equipamentos e controle na palma da mão.

VII. BIBLIOGRAFIA

[1] MOREIRA, Diego. COMUNICAÇÃO RS485 ARDUINO: APRENDA A COMUNICAR 3 ARDUINOS ATRAVÉS DA COMUNICAÇÃO SERIAL:. 2021. Usinainfo eletronica e robotica. Disponível em: https://www.usinainfo.com.br/blog/comunicacao-rs485-arduino-aprenda-a-comunicar-3-arduinos-atraves-da-comunicacao-serial/. Acesso em: 13 dez. 2022.

[2] SILVEIRA, Cristiano Bertulucci. Saiba Tudo Sobre o Protocolo Modbus:. 2016. Cytissystems. Disponível em: https://www.citisystems.com.br/modbus/#:~:text=Modbus%20%C3%A9%20um%20protocolo%20de,redes%20seriais%20entre%20dispositivos%20eletr%C3%B4nicos.. Acesso em: 13 dez. 2022

[3] SILVA, Antonio. PARA QUE SERVE O CÓDIGO ASCII?: .. 2019. HUICODE ACADEMY. Disponível em: https://www.huicode.com.br/2019/06/para-que-serve-o-codigo-ascii.html. Acesso em: 13 dez. 2022

[4] (OLGA WEIS, 2020, Brasil. Tudo sobre Pinagem e Sinais RS485. Brasil, 2020. 8 p.)

[5] (INTRODUÇÃO ao ESP8266. Direção de Fernando K.. S.I., 2017. (21 min.), color. Acedido em 02 de Maio de 2022 em: https://www.youtube.com/watch?v=QXB5WHH9GOE.

[6] PEDRA, Camilo. O que é um inversor de frequência? Como é o seu funcionamento? 2020. Elaborado por Schneider Electric. Acedido em 11 de Setembro de 2022 em: https://blog.se.com/br/automacao-industrial/2020/10/22/o-que-e-um-inversor-de-frequencia-e-como-escolher-a-melhor-opcao/.

[7] WEG SA (Santa Catarina) (org.). Inversor de Frequência CFW500: sobre o produto. Sobre o produto. 2018. Elaborado por WEG SA. Acedido em 11 de Setembro de 2022. em: https://www.weg.net/catalog/weg/BR/pt/Automa%C3%A7%C3%A3o-e-Controle-Industrial/Drives/Inversores-de-Frequ%C3%AAncia/Drives-para-OEMs-e-Uso-Geral/Inversor-de-Frequ%C3%AAncia-CFW500/Inversor-de-Frequ%C3%AAncia-CFW500/p/MKT_WDC_BRAZIL_PRODUCT_INVERTER_CFW500.

[8] SOFTWARE, Elipse. ELIPSE E3. 2015. Acedido em 04 de Junho de 2022 em: https://www.elipse.com.br/en/produto/elipse-e3/.

VIII. APÊNDICES

1. Programa 1

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>


// Configração do WiFi
const char* ssid = "TELMA_2.4G";    // Rede Wifi
const char* password = "01051313";  // Senha Wifi

// Variáveis de Server e Status do LED
ESP8266WebServer server(80);
bool LEDstatus = LOW;

void setup() {
  // Inicia Serial e LED
  Serial.begin(115200);
  pinMode(2, OUTPUT);

  // Inicia Conexão WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Aguarda Conexão e Informa IP
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Rede WiFi: ");
  Serial.println(ssid);
  Serial.print("Endereço IP: ");
  Serial.println(WiFi.localIP());
  delay(100);

  // Configura Handles do Server e Inicia Server
  server.on("/", handle_OnConnect);
  server.on("/ledon", handle_ledon);
  server.on("/ledoff", handle_ledoff);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("Servidor HTTP iniciado!");

}

void loop() {
  server.handleClient();    // Faz o Handle
  if (LEDstatus)            // Checa se LED deve
acender
    digitalWrite(2, HIGH);
  else
    digitalWrite(2, LOW);
}

// FUNÇÕES HANDLE PARA HTML SERVER

void handle_OnConnect() {
  LEDstatus = LOW;
  server.send(200, "text/html", SendHTML(false));
}

void handle_ledon() {
  LEDstatus = LOW;
  server.send(200, "text/html", SendHTML(true));
}

void handle_ledoff() {
  LEDstatus = HIGH;
  server.send(200, "text/html", SendHTML(false));
}

void handle_NotFound() {
  server.send(404, "text/plain", "Not found");
}

String SendHTML(uint8_t led) {
  String ptr = "<!DOCTYPE html>\n";
  ptr += "<html>\n";
  ptr += "<head>\n";
  ptr += "<title>Controle do LED</title>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<h1>LED</h1>\n";
  ptr += "<p>Clique para Ligar e Desligar o
LED.</p>\n";
  ptr += "<form method=\"get\">\n";
  if (led)
    ptr += "<input type=\"button\" value=\"LED
OFF\"
onclick=\"window.location.href='/ledoff'\">\n";
  else
    ptr += "<input type=\"button\" value=\"LED
ON\"
onclick=\"window.location.href='/ledon'\">\n";
  ptr += "</form>\n";
  ptr += "</body>\n";
  ptr += "</html>\n";
  return ptr;
}
```

2. Programa 2

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>

const char* ssid = "TCCInver";
const char* password = "8266";

String site = {""};

ESP8266WebServer server(80); // server: http://192.168.4.1

void handleRoot()
{
  site = "<html\n>";
```

```
  site += "<head><title>Teste de rede</title></head>\n";
  site += "<body style=\"color: blue\">\n";
  site += "<center><h1>Teste de rede usando
ESP8266</h1></center>\n";
  site += "</body>\n";
  site += "</html>";

  server.send(200, "text/html", site);

  site = "";
}

void setup() {
  WiFi.mode(WIFI_AP); // aceita WIFI_AP / WIFI_AP_STA
/ WIFI_STA
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  server.on("/", handleRoot);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

- **Programa 3**

```
<!DOCTYPE html>
<html>
<head>
<title>Comandos para inversor</title>
</head>
<body>

<h1 style="color:blue;"> Comandos para inversor de
frequencia</h1>
<hr>
<hr>
<br>

<p style="font-size:120%;">___Aceleração do motor.</p>
<input type="button"
onclick=\"window.location.href='/Monitor de velocidade'\"
value="Aceleração">
<hr>

<p style="font-size:120%;">___Frequência do motor.</p>
<input type="button"
onclick=\"window.location.href='/Monitor de velocidade'\"
value="Frequência">
<hr>


<p style="font-size:120%;">Corrente do motor.</p>
<input type="button"
onclick=\"window.location.href='/Monitor de velocidade'\"
value="Corrente">
<hr>

<p style="font-size:120%;">Tensão do motor.</p>
<input type="button"
onclick=\"window.location.href='/Monitor de velocidade'\"
value="Tensão">
<hr>

</body>
</html>
```

- **Programa 4**

```
#include <Arduino.h>
#include <WiFi.h>
#include <ModbusClientTCP.h>  // Inclui a biblioteca
ModbusClient TCP
#include <HardwareSerial.h>   // biblioteca serial
#include <ModbusBridgeWiFi.h> // Inclui a biblioteca
Modbus bridge
#include <ModbusClientRTU.h>  // Inclui a biblioteca
Modbus RTU Client


int8_t LED_AM = 23;          // LED amarelo, para
sinalização da WiFi
uint16_t PORT = 502;         // Porta de comunicação Modbus
TCP

char ssid[] = "xxxx";       // Nome da rede
char pass[] = "xxxx";       // Password


ModbusClientRTU MBRTU(Serial2);   // Cria uma instância
ModbusRTU Cliente
ModbusBridgeWiFi MBGateway(5000); // Cria o servidor e
define o tempo maximo de espera por respostas

IPAddress local_IP(10, 0, 0, 9);  // Seta o Ip local (fixo)
IPAddress gateway(10, 0, 0, 1);

// Ajustes opcionais
IPAddress subnet(255, 255, 255, 0); //  Mascara de rede
IPAddress primaryDNS(8, 8, 8, 8);   // DNS primario
IPAddress secondaryDNS(8, 8, 4, 4); // DNS secundario

/*--------------------------------------------------------------
    Setup inicial
--------------------------------------------------------------*/
void setup(void)
{
  Serial.begin(115200);         // Inicia a serial principal
(monitor serial)
  while(!Serial) {}
  Serial.println("__ OK __");   // Aguarda serial estar OK

  pinMode(LED_AM, OUTPUT);      // Define o pino como
saida, para sinalização com o LED
  Serial2.begin(19200, SERIAL_8E1); // reconfigura a serial
para o padrão do inversor: 19200bps, paridade par, 1 bit stop

  if(!WiFi.config(local_IP, gateway, subnet, primaryDNS,
secondaryDNS))  // Cria/configura o endereço IP
```

```
  {                                                 // fixo e
demais configurações.
    Serial.println("Falha na configuração da WiFi");   //
Envia mensagem pela serial

    while(1) digitalWrite(LED_AM, (millis() / 250) % 2); //
loop infinito, pisca LED @ 4Hz.
  }

  WiFi.begin(ssid, pass);              // Inicia conexão a rede
WiFi
  while (WiFi.status() != WL_CONNECTED)  // Aguarda
conexão ser estabelecida
  {
    Serial.print(". ");
    delay(500);
  }

  IPAddress wIP = WiFi.localIP(); // Carrega IP local
  Serial.printf("IP address: %u.%u.%u.%u\n", wIP[0],
wIP[1], wIP[2], wIP[3]); // Envia pela serial

  digitalWrite(LED_AM, HIGH);   // Liga LED indicando
que a conexão com a WiFi foi estabilecida

  MBRTU.setTimeout(2000); // Seta o tempo maximo para
resposta da mensagem RTU em 2000ms
  MBRTU.begin();         // Inicia a tarefa ModbusRTU em
"segundo plano"


  //--- Define e inicia o modo WiFi Gateway ---
  // ServerID 1: Servidor WiFi (ethernet TCP) 1, com
servidor remoto (Modbus RTU) 5, acessado através do
  //           ModbusRTU Cliente (MBRTU). Todos os
códigos de função aceitos.
  MBGateway.attachServer(1, 5, ANY_FUNCTION_CODE,
&MBRTU);
  MBGateway.denyFunctionCode(1,
WRITE_HOLD_REGISTER);      // Nega o código de
função 0x06.
  MBGateway.denyFunctionCode(1,
WRITE_MULT_REGISTERS);     // Nega o código de
função 0x10.

  // ServerID 2: Servidor WiFi (ethernet TCP), com servidor
remoto (Modbus RTU) 5, acessado através do
  //           ModbusRTU Cliente (MBRTU). Aceita somente
o código de função 0x06.
  MBGateway.attachServer(2, 5,
WRITE_HOLD_REGISTER, &MBRTU);
  MBGateway.addFunctionCode(2,
WRITE_MULT_REGISTERS);        // Adiciona o código de
função 0x10.

  // ServerID 3: Servidor WiFi (ethernet TCP) 3, com
servidor remoto (Modbus RTU) 6, acessado através do
  //           ModbusRTU Cliente (MBRTU). Todos os
códigos de função aceitos.
  MBGateway.attachServer(3, 6, ANY_FUNCTION_CODE,
&MBRTU);

  MBGateway.listServer();  // Checagem: Envia todas as
confinações de servidores pela serial

  MBGateway.start(PORT, 4, 1000); // Inicia o gateway.
Porta 502, permite conexão de até 4 clientes.
                                  // Tempo de inatividade maximo do
cliente: 1000ms

  Serial.printf("Use o IP mostrado e a porta %d para enviar
requisições\n", PORT);

}


/*--------------------------------------------------------------
   Loop principal
--------------------------------------------------------------*/
void loop()
{
  while(1) digitalWrite(LED_AM, (millis() / 1000) % 2); //
loop infinito, piscando LED.
}
```