UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Gustavo de Lima Machado

DESENVOLVIMENTO DE LABORATÓRIO REMOTO PARA BANCADA ELETROPNEUMÁTICA DIDÁTICA

Santa Maria, RS
2019

Gustavo de Lima Machado

DESENVOLVIMENTO DE LABORATÓRIO REMOTO PARA BANCADA ELETROPNEUMÁTICA DIDÁTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação, Área de Concentração em Redes de Comunicação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **em Engenharia de Controle e Automação**.

ORIENTADOR: Prof. Frederico Menine Schaf

Santa Maria, RS
2019

Endereço: Rua Augusta, n. 41, complemento 402
Fone (0xx) 51 98345 7609; End. Eletr.: gustavolmachado@outlook.com

Gustavo de Lima Machado

DESENVOLVIMENTO DE LABORATÓRIO REMOTO PARA BANCADA ELETROPNEUMÁTICA DIDÁTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação, Área de Concentração em Redes de Comunicação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **em Engenharia de Controle e Automação**.

Aprovado em 17 de dezembro de 2019:

Frederico Menine Schaf, Dr. (UFRGS)
(Presidente/Orientador)

Claiton Moro Franchi, Dr. (UEM)

Mateus Beck Rutzig, Dr. (UFRGS)

Santa Maria, RS
2019

RESUMO

DESENVOLVIMENTO DE LABORATÓRIO REMOTO PARA BANCADA ELETROPNEUMÁTICA DIDÁTICA

AUTOR: Gustavo de Lima Machado
ORIENTADOR: Frederico Menine Schaf

A evolução da tecnologia e o crescente uso da Internet por mais pessoas cria possibilidades para aplicações remotas, controladas via Web. Desta forma, laboratórios remotos se tornam uma realidade que auxilia acadêmicos na competitividade do mercado e abrem a possibilidade do acesso a experimentos por cada vez mais estudantes. Portanto, o desenvolvimento de dada aplicação possui inúmeras vantagens em diversos níveis e fomenta o desenvolvimento pessoal e profissional. A questão quanto a aplicabilidade de tal conceito está na comunicação da rede formada entre todos os dispositivos. Desde a planta que será controlada por um sistema SCADA, o servidor que envia dados ao cliente e o usuário, todos os dispositivos devem possuir uma maneira de se entenderem e garantirem o êxito na troca de dados. Este desenvolvimento Web da comunicação e criação de cliente e servidor é o que torna o laboratório remoto funcional e permite que experimentos práticos sejam realizados em qualquer lugar, a qualquer hora. Para tal projeto funcionar, o usuário deverá programar as ações de um CLP e enviar o arquivo de programação a uma bancada (planta do laboratório), o que é feito através de um website com um servidor e páginas criadas em HTML e JavaScript. Visando a aplicabilidade de tais conceitos e a facilidade para desenvolvedores, diversos instrumentos foram criados, um destes sendo um interpretador assíncrono de JavaScript chamado de “Node.js” que abre maneiras de transformar projetos complexos em simples, permitindo ações que antes demandavam muito tempo ou não eram possíveis com tal linguagem. A criação das páginas em HTML e JavaScript, as ações do servidor usando “Node.js” e a comunicação do servidor com o CLP utilizando uma biblioteca que controla o mouse e teclado do servidor combinados tornam o laboratório remoto funcional e auxiliam na formação de estudantes, tanto facilitando desenvolvimentos de trabalhos acadêmicos quanto instruindo-os a olhar para o futuro em novas tecnologias.

Palavras-chave: Laboratórios Remotos. SCADA. Comunicação. Desenvolvimento Web. JavaScript. Node.

ABSTRACT

REMOTE LABORATORY DEVELOPMENT FOR ELECTROPNEUMATIC DIDATIC BENCH

AUTHOR: Gustavo de Lima Machado
ADVISOR: Frederico Menine Schaf

The evolution of technology and the growing use of Internet by more people creates possibilities for remote applications, controlled through Web. Thus, remote laboratories become a reality that assists academics in the competitiveness of market and open the possibility of access to experiments by more and more students. Therefore the development of given application has countless advantages in several levels and promotes the personal and professional development. The question as to the applicability of this given concept is in the communication of the network formed among all the devices. From the SCADA system controlled plant, the server that sends data to the client and the user, all devices must have a way of understand each other and ensure the success in data exchange. This web development of the communication and creation of client and server is what makes the remote laboratory functional and allows that practical experiments to be performed anywhere, anytime. To make the project work, the user must program the actions of a PLC and send the programming archive to a bench (the laboratory plant), which is made through a website with a server and pages created with HTML and JavaScript. Aiming at applicability of these concepts and the facility for developers, several tools were created, one of those being an asynchronous interpreter of JavaScript called “Node.js” that opens ways of transformming complex projects into simple ones, allowing actions which before took a lot of time or weren’t possible with that language. The creation of pages with HTML and JavaScript, the server actions using “Node.js” and the communication of the server with the PLC using a library that control the mouse and keyboard of the server combined make the remote laboratory functional and support the graduation of students, both in making academic works and instructing them to look at the future on new technologies.

Keywords: Remote Laboratories. SCADA. Communication. Web Development. JavaScript. Node.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

$NUPEDEE$	Núcleo de Pesquisa e Desenvolvimento em Engenharia Elétrica
IoT	Internet of Things
CLP	Controlador Lógico Programável
$NEMA$	National Electrical Manufacturers Association
IEC	International Electrotechnical Commission
SFC	Sequential Function Chart
FBD	Function Block Diagram
OSI	Open System Interconnection
IP	Internet Protocol
$HTTP$	Hypertext Transfer Protocol
$HTTPS$	Hypertext Transfer Protocol Secure
$DHCP$	Dynamic Host Configuration Protocol
$SMTP$	Simple Mail Transfer Protocol
$IMAP$	Internet Message Access Protocol
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
$SCADA$	Supervisory Control And Data Acquisition
MTU	Master Terminal Unit
RTU	Remote Terminal Unit
$HTML$	Hypertext Markup Language
USB	Universal Serial Bus
$HDMI$	High-Definition Multimedia Interface
PHP	PHP: Hypertext Preprocessor
FS	File System
GUI	Graphical User Interface
$YAWCAM$	Yet Another WebCam Software
CSS	Cascading Style Sheets
NPM	Node Package Manager
API	Application Programming Interface

SUMÁRIO

1 INTRODUÇÃO

Em face da evolução tecnológica e crescente uso da Internet, há uma tendência de ter-se todo dispositivo conectado a rede. À essa tendência damos o nome de *Internet of Things* (IoT). IoT é um conceito de que todo dispositivo e equipamento eletrônico se conecte a Internet e sua informação seja compartilhada e controlada a distância através da rede. Tanto em nível profissional quanto pessoal, ter conhecimento básico do funcionamento e comunicação de tal conceito é fundamental para o crescimento.

Segundo estudo da Bsquare (2017), uma empresa especializada em IoT a nível industrial, 89% das indústrias já adotam o conceito, e 73% acreditam que as soluções de ter os aparelhos conectados a Internet foram muito efetivas. Destes dados, pode-se perceber o interesse de indústrias por esta nova tecnologia e sua importância no desenvolvimento de profissionais que possuam experiência na área.

Diretor geral e *head* de vendas da *Telit Communications* (outra empresa especializada em IoT), Simon (2018) afirma que “conhecer ao menos o mínimo sobre a IoT transcende a mera atualização profissional e torna-se um diferencial competitivo”. Esta afirmação apenas destaca o quanto este conceito é importante para o futuro de uma pessoa, e serve também para atentar aos estudantes que estão indo para o mercado. Afinal, aplicar esta ideia no espaço acadêmico acaba se tornando não apenas agregador, mas uma necessidade para competição no mercado.

A busca por este desenvolvimento tecnológico está em alta e no Brasil, investimentos em redes especializadas em IoT começaram a aparecer. Um exemplo de rede especializada foi criado no projeto Rede IoT Rio de Janeiro, que tem por objetivo “Criar redes de Internet das Coisas que promovam o desenvolvimento socioeconômico nas cidades do Brasil” (RIO, 2019), como forma de desenvolver ainda mais a área.

O próprio *Wi-Fi* como o conhecemos também evoluiu para que consiga acompanhar o desenvolvimento. Segundo matéria do site Loefflero (2019), enquanto o 5G se expande, a próxima versão do *Wi-Fi* - o chamado “WiFi 6” - tende a impactar profundamente e melhorar o acesso da tecnologia IoT, aumentando o número de nodos que um roteador consegue fornecer acesso e a velocidade geral da conexão.

Outro fator em ascenção é a experiência prática requerida por empresas em áreas técnicas, o que força estudantes a buscar instrumentos e experimentos que os auxiliem a melhorar o currículo. Essa alta demanda por instrumentos, geralmente industriais, acaba gerando custos para obtenção de equipamentos novos e alocação de espaço maior. Visto que equipamentos industriais são caros e exigem um espaço considerável, alternativas começam a entrar em questão.

Uma destas alternativas é a integração da comunicação da Internet com os ex-

perimentos, criando o emprego de práticas laboratoriais remotas. Estes laboratórios remotos trazem grandes benefícios aos estudantes, uma vez que possuem flexibilidade espacial e temporal. Espacial pois podem ser acessados até mesmo fora da instituição de ensino e temporal devido a possibilidade de acesso a qualquer hora (SCHAF, 2011).

Deste modo, trazer tais conceitos, não apenas para experimentações mas para salas de aula e assim demonstrar o funcionamento de tal prática, abre possibilidades de aulas remotas. Também mostra aos estudantes aplicações de tais conhecimentos, comprovando a importância da comunicação através da Internet. Acaba se tornando um ganho para a instituição de ensino (que traz um auxílio ainda maior aos estudantes) e para todo o escopo industrial, com a vinda de acadêmicos com maior experiência em técnicas na área. Além da demonstração aos alunos do que pode ser feito com o trabalho em IoT, aumentando o interesse destes em práticas do futuro.

Desta maneira, o presente trabalho possui por objetivo o desenvolvimento de um laboratório remoto com uma bancada eletropneumática didática através do desenvolvimento de um *website* que pode ser acessado pelos estudantes e professores a distância. O acesso via Internet em navegadores torna possíveis aulas remotas e experimentações com a planta industrial simulada mesmo fora do ambiente acadêmico. Tal laboratório remoto será suprido de um controle de acesso, um método de programar a bancada e um sistema supervisório, tornando o laboratório remoto capaz de suprir todas as necessidades de um experimento prático, como se fosse local.

2 REFERENCIAL TEÓRICO

Para compreender o presente trabalho, alguns conceitos e definições devem ser apresentados previamente, dado que serão a base para o desenvolvimento da aplicação. Tais conceitos serão mostrados neste capítulo, onde serão apresentadas suas definições e explicado seu funcionamento resumidamente.

2.1 CONTROLADOR LÓGICO PROGRAMÁVEL

Em automação industrial, é muito comum o uso de Controladores Lógicos Programáveis (CLP) para garantir a execução de todo o processo mecânico de forma controlável e autônoma. Estes dispositivos funcionam como o "cérebro" da operação e executam códigos programados por seres humanos com a finalidade de controlar, monitorar e automatizar o processo industrial.

Oficialmente, a Associação Nacional de Fabricantes de Equipamentos Elétricos dos Estados Unidos da América (*National Electrical Manufacturers Association* NEMA), define o CLP como um aparelho eletrônico digital, que utiliza uma memória programável para armazenar internamente instruções e implementar funções específicas, tais como lógica, sequenciamento, temporização, contagem e aritmética, controlando, por meio de módulos de entradas e saídas diversas máquinas e processos (ZANCAN, 2011).

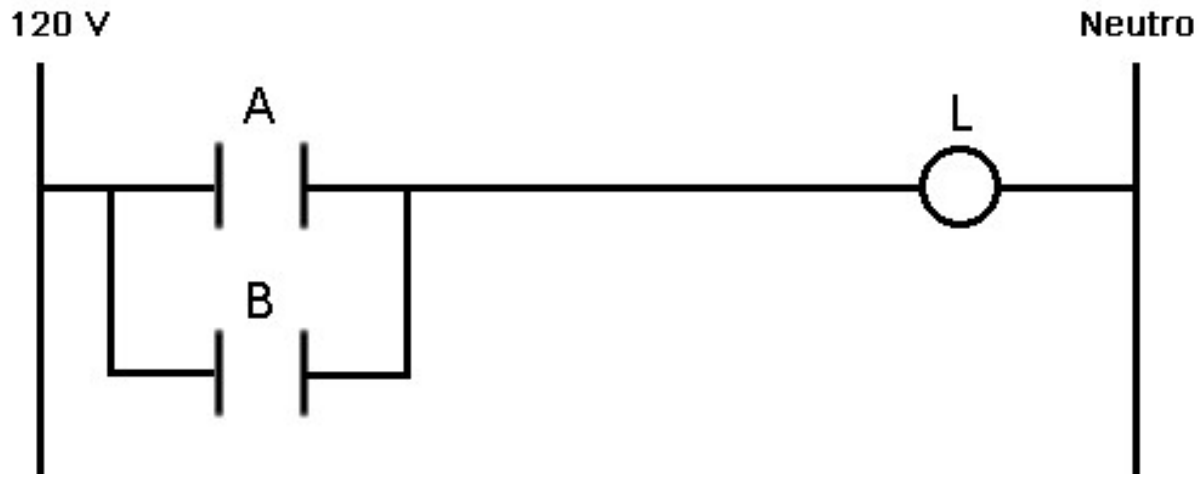
A comunicação de dados dos CLP's se dá através de canais seriais, embutidos de um protocolo, formando uma rede interligada de dispostivos em uma indústria. A rede formada por estas conexões se dá o nome de Rede Industrial e há vários protocolos que regem essa comunicação, dentre os quais alguns dos mais famosos são o Modbus, Profibus e Devicenet.

A programação desses dispositivos se dá através de *softwares* especificamente projetados muitas vezes para apenas um modelo. Tais ferramentas possuem o protocolo de comunicação para entrar em contato com o CLP e sua programação é realizada baseando-se nas regras impostas pela IEC 61131 (Comissão Eletrotécnica Internacional, norma 61131). Nesta norma são definidas cinco linguagens de programação para o CLP, sendo estas: texto estruturado (*Structured Text*), lista de instruções (*Instruction List*), SFC (*Sequential Function Chart*), FBD (*Function Block Diagram*) e *Ladder*.

2.1.1 Ladder

Sendo uma das maneiras de programar um CLP, a linguagem *Ladder* - também chamada de Diagrama Ladder - foi desenvolvida com o intuito de ter uma melhor documentação e construção de programação de dispositivos baseados em relés. Ferramentas como botões, válvulas e solenóides são facilmente representadas em *Ladder*. Seu nome vem do modo como é feita a disposição de contatos e bobinas, o que lembra uma escada. A Figura 2.1 mostra um exemplo simples de diagrama *ladder*.

Figura 2.1 – Exemplo de Diagrama Ladder.



Fonte: Adaptado de Silveira (2016)

É uma linguagem considerada de baixo nível utilizada na programação de CLP's, onde cada variável é relativa a um endereço de registrador no controlador ou da memória interna do mesmo. Em *Ladder*, as entradas (geralmente sensores) são chamadas de contatos e as saídas (atuadores), de bobinas. Os contatos possuem a representação gráfica como duas barras ("-| |-") e as bobinas como dois parênteses ("-( )-"), podendo ambos possuírem lógica direta ou inversa (inicialmente aberta ou fechada).

O *Ladder* também possui diversos blocos funcionais que podem disparar temporizadores ou outras lógicas booleanas e operações (como as lógicas "AND" e "OR").

2.2 PROTOCOLOS DE COMUNICAÇÃO

O compartilhamento de dados em uma rede é realizada através de uma comunicação previamente estabelecida. Esta comunicação é regida através de protocolos específicos, ou seja, uma série de normas e regras que os dispositivos devem cumprir no envio e recebimento de mensagens, de forma a não deixar a comunicação caótica

e incompreensível para os dispositivos envolvidos. Deve ser atentado o fato de que todos os dispositivos da rede devem possuir os mesmo protocolos, garantindo assim o diálogo entre eles.

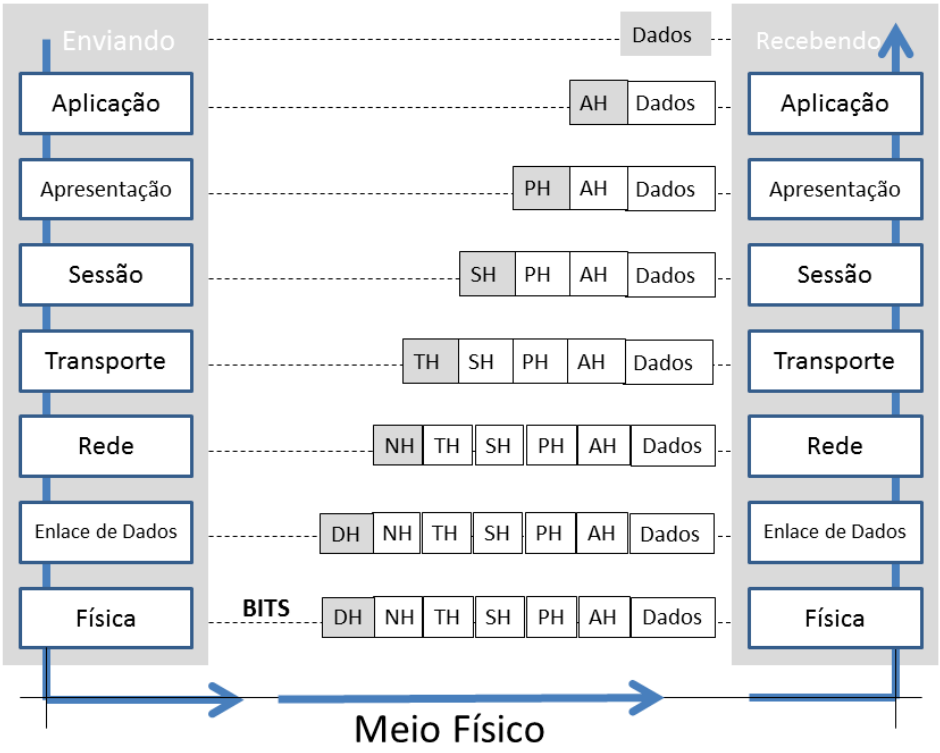
Um protocolo de comunicação é responsável por determinar, dentre outras características, o *baudrate* (taxa de transmissão de bits, usualmente em bits por segundo), o número de *data bits* (bits de dados), a paridade, delimitadores de início e fim e checagem de erros de todas as mensagens trocadas entre os dispositivos da rede. Geralmente, os pacotes de comunicação são divididos em *header*, conteúdo e *footer* (em respectivamente, início, meio e fim da mensagem).

2.3 MODELO DE REFERÊNCIA OSI

O modelo de referência de Interconexões de Sistemas Abertos (do inglês, OSI - *Open System Interconnection*) traz uma padronização às comunicações de redes industriais a nível internacional. Esta padronização é feita através de uma divisão em sete camadas, separando os objetivos para estabelecer a comunicação e deixando a informação de uma camada encapsulada dentro do campo de dados da anterior. Os objetivos de cada camada são os seguintes:

- Camada Física: responsável pela transmissão de dados brutos através de um canal de comunicação em bits. Protocolos usuais desta camada são o RS-232 e o RS-485;
- Camada de Enlace de Dados: proporciona meios funcionais e procedurais para a transferência de dados, além de detectar e corrigir erros vindos da camada física;
- Camada de Rede: responsável pelo endereçamento lógico das mensagens e roteamento (define um caminho entre nós para a mensagem);
- Camada de Transporte: estabelece conexões de fim a fim, ou seja, entre um ponto e outro de maneira virtual (também chamada de conexão *peer-to-peer*), também estabelecendo confiabilidade;
- Camada de Sessão: gerenciamento de sessões entre aplicações e usuários em diferentes máquinas, proporcionando controle de diálogo, sincronização, etc.;
- Camada de Apresentação: responsável pela representação de dados, ou seja, conversão de informações de máquina em informações de usuário;
- Camada de Aplicação: proporciona acesso a rede à aplicação, é onde está contida a interface com o usuário.

Figura 2.2 – Esquema de camadas do modelo OSI, mostrando encapsulamento.



Fonte: Farias e Medeiros (2013)

A Figura 2.2 apresenta a disposição de camadas do modelo OSI. Cada camada tem responsabilidade por informações essenciais a aplicação e deve ser regida por seus próprios protocolos a fim de estabelecer a comunicação.

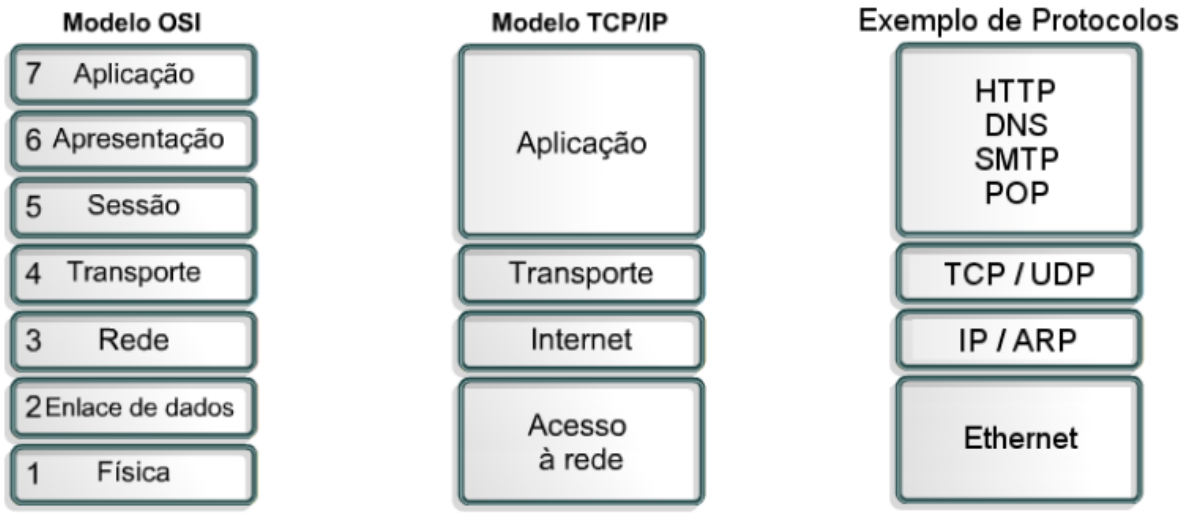
2.3.1 Protocolos de Rede

Necessários em toda a máquina que é conectada a rede, os protocolos de rede são o conjunto de regras que regem a comunicação entre computadores conectados à Internet. Ao utilizar deles, baixar arquivos, enviar mensagens e acessar *websites* se tornam tarefas possíveis. Há vários protocolos de rede, cada um relacionado a uma tarefa específica com a finalidade de estabelecer a comunicação, alguns dos principais abordados no presente trabalho são:

- Arquitetura TCP/IP: é o acrônimo de dois protocolos: TCP (*Transmission Control Protocol*, ou em português, Protocolo de Controle e Transmissão) e IP (*Internet Protocol*, Protocolo de Internet). Esta arquitetura é composta por um conjunto de protocolos que conseguem cobrir o funcionamento de todas as camadas do

modelo OSI, como pode ser observado na Figura 2.3, junto aos protocolos que “encapsulam” a arquitetura TCP/IP.

Figura 2.3 – Modelo OSI comparado a arquitetura TCP/IP.



Fonte: Material didático prof. Frederico Schaf

- Protocolo HTTP e HTTPS: os protocolos HTTP (*Hypertext Transfer Protocol*, ou Protocolo de Transferência de Hipertexto) e HTTPS (*Hypertext Transfer Protocol Secure*, Protocolo de Transferência de Hipertexto Seguro) funcionam de maneira muito similar e executam as mesmas funções, com a diferença de um oferecer mais segurança que o outro. Eles são usados para a navegação básica em *sites* na Internet. O cliente envia uma solicitação para visualizar uma página, por exemplo, e o servidor envia uma resposta (a página em si), caso haja permissão para visualizar.

  O protocolo HTTPS está em ascenção em sua utilização devido a segurança extra que proporciona, principalmente em aplicações que dependem da privacidade de dados, como sistemas de pagamento. Ele é utilizado quando o *site* acessado possui um certificado SSL (*Secure Sockets Layer*, Camada de Sockets de Segurança), que cria uma criptografia para impedir ameaças e ataques.

- Protocolo DHCP: acrônimo para Protocolo de Configuração Dinâmica de Endereços de Rede (*Dynamic Host Configuration Protocol*). É um protocolo que possui a função de obter um endereço IP para o dispositivo conectado à Internet. Endereços IP são a maneira de identificar usuários da rede, ou seja, um identificador de um dispositivo quando este é conectado, compreendendo informações como protocolo, rede, domínio e país. Vale ser mencionado que um endereço IP é único e no momento que uma máquina o utiliza, ele fica indisponível para

ser utilizado por outra, até o momento em que o computador seja desligado ou desconectado.

Endereços IP ainda podem assumir uma forma “nominal”, englobando tanto seu valor como identificador quanto seu canal de comunicação (a porta com o qual vai se conectar).

É possível definir o IP de uma máquina de 3 maneiras: automaticamente, dinamicamente ou ainda manualmente, sendo este último formando um endereço único e estático. A maneira manual é utilizada quando é necessário um IP fixo para a máquina, geralmente quando se pretende utilizar como um servidor.

- Protocolo SMTP: o Protocolo de Transferência de Correio Simples (*Simple Mail Transfer Protocol*) é voltado para o envio de *e-mails*. O cliente envia a mensagem determinando os destinatários, é autenticada por um servidor, onde os usuários recebem. Apesar de simples, é limitado, uma vez que aceita somente formato de texto.

- Protocolo IMAP: similar ao SMTP, o IMAP (*Internet Message Access Protocol*, ou Protocolo de Acesso à Mensagem de Internet) também possui funções relacionadas ao envio e recebimento de *emails*. A diferença está no gerenciamento do servidor, uma vez que o SMTP é carregado de todas as mensagens que precisa por cada usuário, e o IMAP permite que o usuário acesse e gerencie seus arquivos e mensagens diretamente no próprio servidor.

2.3.2 Sockets

Soquetes (ou *Sockets*, do inglês) pertencem a camada de transporte do modelo OSI e servem para a troca de informações entre *hosts* (as máquinas na rede). *Sockets* permitem haver comunicação ponto a ponto entre um cliente e um servidor, onde o servidor responde requisições do cliente. São endereçados através de um endereço IP e uma porta, necessários para um fim de linha de comunicação e estabelecer a própria.

Existem dois tipos básicos de protocolo para *sockets*: TCP e UDP. *Sockets* que utilizam UDP (*User Datagram Protocol*) não garantem confiabilidade e apenas lançam a mensagem com a requisição do cliente, também não garantem ordem de entrega das mensagens. Já o protocolo TCP (*Transmission Control Protocol*), onde o foco é confiabilidade e controle de fluxo, há mensagens de confirmação, indicando se a informação chegou ou não ao seu destino, bem como maneiras de controlar o estado da conexão e garantir a ordem de entrega das mensagens.

Outro protocolo pouco conhecido para *sockets* é o Multicast. Ele funciona de maneira similar, com ramificações visando garantia de confiabilidade (Multicast Confiável) ou sem garantia (Multicast não-confiável). A maior diferença do protocolo Multicast é o destino das mensagens do *socket*: ao invés de ser uma conexão ponto a ponto, a mensagem é enviada a um grupo de processos através de um único pacote IP.

2.4 SERVIDORES

Em uma arquitetura de comunicação denominada “cliente-servidor”, há duas máquinas: o cliente, que faz requisições de arquivos e funções, e o servidor, cuja função principal é prover o cliente dos serviços requeridos. Um servidor pode servir vários ou apenas um cliente, dependendo de sua configuração.

Portanto, um servidor é um *software* ou computador que provém serviços a uma rede de outras máquinas (comumente a própria Internet, mas pode ser local). Tais serviços podem ser de várias naturezas, como correios eletrônicos, *websites*, arquivos para *download*, dentre outros.

Um servidor que provém serviços pela Internet pode ser acessado através do endereço IP da máquina a qual está inserido e da porta de comunicação onde estão os serviços. Desta maneira, o cliente tem acesso a suas funcionalidades. Caso o endereço IP possua uma forma nominal, esta também pode ser utilizada para acessá-lo.

2.5 BANCOS DE DADOS

Como o nome sugere, bancos de dados são conjuntos/coleções de dados guardados em software e relacionados entre si de forma a atribuir algum sentido. Um exemplo é uma lista telefônica, que pode ser considerada um banco de dados que mistura nomes, números de telefone e endereços.

Essas ferramentas são operadas por um SGBD (Sistema de Gerenciamento de Banco de Dados), sendo este um *software* capaz de manipular as informações do banco de dados e comunicar-se com o usuário.

Um dos usos mais comuns de tal tecnologia é guardar informações de cada usuário de um *website*. Dados de *login* e senha são colocados nestes bancos de dados e requisitados toda vez que o usuário tenta estabelecer uma sessão no navegador, onde o restante de suas informações são utilizadas.

2.6 CÂMERAS IP

Câmeras IP são equipamentos que transmitem vídeo e áudio para a internet ou para uma máquina em específico, tornando possível o monitoramento a distância. São utilizadas principalmente no âmbito da segurança doméstica, com a instalação de câmeras ao redor de uma residência, mas suas aplicações podem estar em diversas áreas, como por exemplo observação remota de processos em uma indústria ou observar recém-nascidos na maternidade.

O dispositivo é conectado diretamente ao roteador ou ao *modem*, onde então ganha um endereço IP exclusivo, semelhante a um computador na rede. Deste momento em diante, basta acessar o endereço IP da câmera em um navegador e pode-se monitorar. Grande parte das câmeras IP já disponibilizam o vídeo em *streaming* em uma página HTML.

2.7 SISTEMAS SCADA

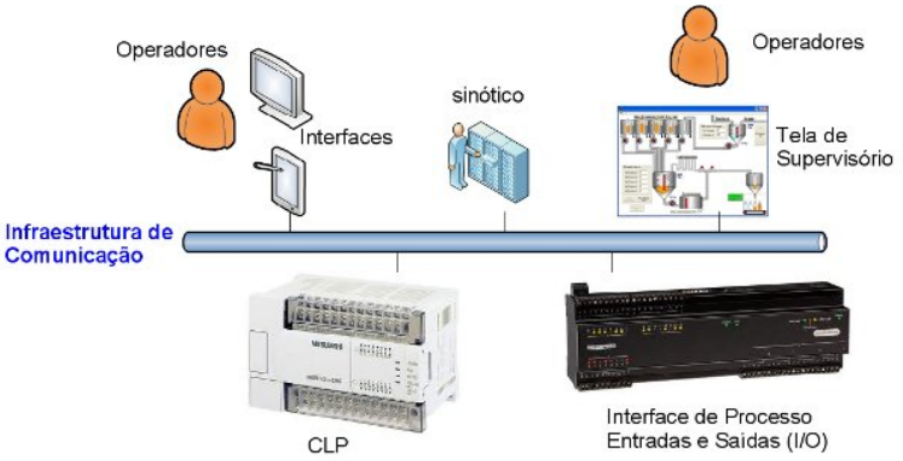
Sistemas de controle supervisório e aquisição de dados (ou SCADA - *Supervisory Control and Data Acquisition*), são sistemas que utilizam qualquer meio para monitorar, controlar e supervisionar todas as variáveis de atuadores e sensores em um processo, bem como operá-lo através de uma interface. É qualquer dispositivo ou *software* que permita coletar e controlar dados de um processo. Um sistema SCADA geralmente possui:

- IHM - Interface Homem-Máquina: a interface onde o usuário acessa o sistema supervisório, geralmente composta por diversas telas com funções específicas;
- Unidade Terminal Mestre (MTU - Master Terminal Unit): a unidade do sistema onde o SCADA está centralizado, fazendo requisições de estado;
- Unidade(s) Terminal(is) Remota(s) (RTU - *Remote Terminal Unit*): unidades remotas posicionadas em locais onde há a necessidade de controle da planta;
- CLP;
- Infraestrutura de comunicação: a comunicação de um sistema SCADA é essencial para seu bom funcionamento. Os dispositivos devem ser capazes de se entender e conversar entre si. Para tanto, *drivers* e circuitos de condicionamento de sinais podem ser necessários.

Sistemas supervisórios também são capazes de gerar relatórios sobre operações realizadas e/ou problemas que ocorreram na planta. Os relatórios mais comuns são de alarmes, de acesso ao SCADA e do estado de variáveis no tempo.

A Figura 2.4 mostra os elementos principais de um sistema SCADA.

Figura 2.4 – Elementos de Sistema SCADA.



Fonte: Material Didático Prof. Frederico Schaf

3 MATERIAIS E MÉTODOS

Neste capítulo serão abordados os *softwares* e materiais utilizados no desenvolvimento do laboratório remoto. Serão explicados como funcionam e a maneira que são utilizados no projeto. Também será explicada a metodologia geral de funcionamento do laboratório e a maneira que este irá cumprir o objetivo do projeto. As tecnologias utilizadas neste trabalho foram escolhidas com base em pesquisa e empirismo e podem não ser as opções mais adequadas ao desenvolvimento.

3.1 BANCADA ELETROPNEUMÁTICA

O laboratório a ser programado e testado remotamente trata-se de uma bancada eletropneumática disponível no Núcleo de Pesquisa e Desenvolvimento em Engenharia Elétrica (NUPEDEE) da Universidade Federal de Santa Maria (UFSM). A bancada em questão pode ser visualizada na Figura 3.1.

Figura 3.1 – Bancada Eletropneumática Didática.

Fonte: Autor

O processo realizado pela bancada trata-se de uma simulação de planta industrial, com usinagem e seleção de peças através de uma linha de produção controlada por um CLP. A peça em questão do processo é um cilindro, que pode ser furado ou não (sólido), para que haja diferença de peso. Os atuadores e sensores são digitais, possuindo valores que podem variar entre 0 ou 1. A planta da bancada possui sensores de estado para cada atuador (desligado/ligado, avanço/recuo), além de sensores que auxiliam o processo em si, sendo estes últimos os seguintes:

- 1 sensor capacitivo;
- 1 sensor óptico;
- 2 sensores indutivos;
- 1 sensor de peso (balança) com referência ajustável para comutação de *status*.

O sensor capacitivo detecta a peça na esteira e faz o atuador do módulo de transferência avançar, ao passo que quando o sensor óptico detecta a peça, o mesmo atuador recua. Os sensores indutivos possuem a função de parar os motores de giro da mesa e do braço de descarga. A balança pode ser definida como o sensor de peso e possui uma referência ajustável - valores acima da referência possuem resposta 1, valores abaixo, resposta 0. Os atuadores são agrupados e rotulados na bancada por módulos de acordo com a sua respectiva função. Tais módulos presentes na bancada são citados abaixo:

- Módulo esteira;

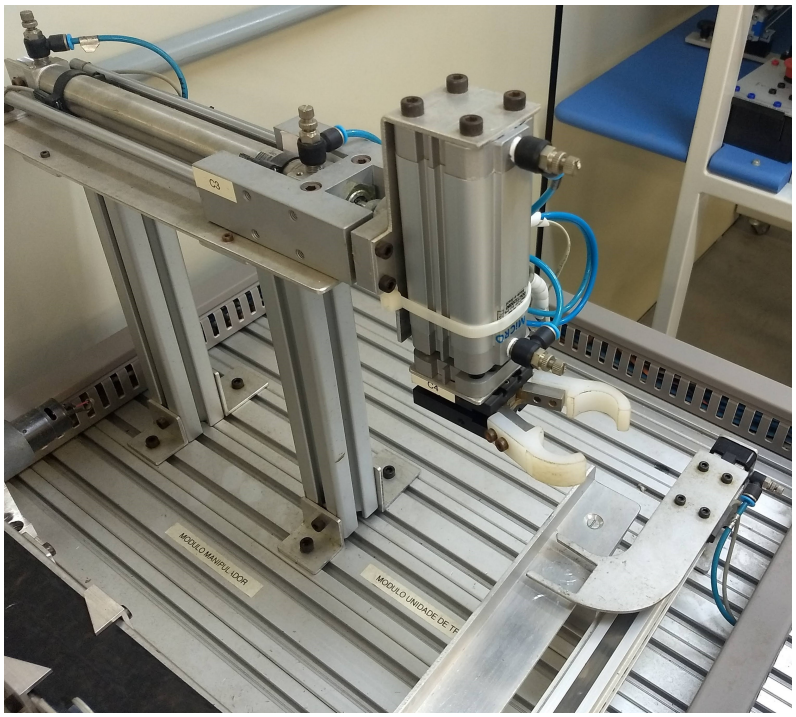
Figura 3.2 – Módulo esteira.

Fonte: Autor

- Módulo de transferência;

- Módulo de manipulação cartesiana;

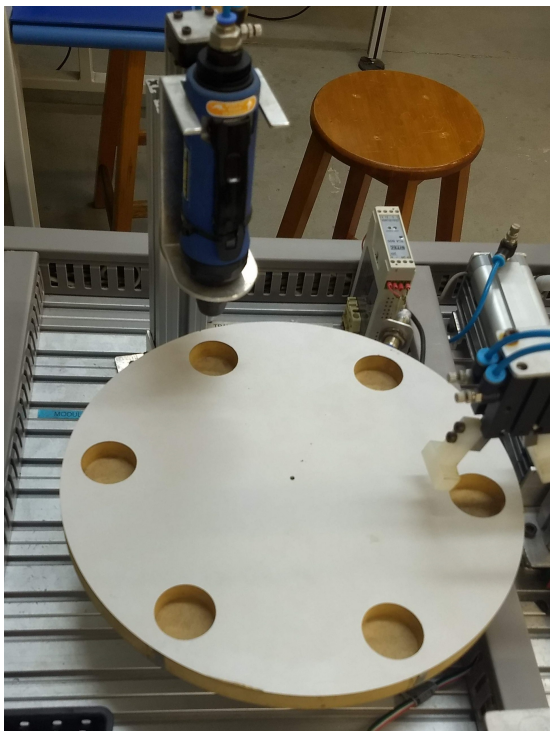
Figura 3.3 – Módulo de transferência e manipulação cartesiana.



Fonte: Autor

- Módulo mesa;

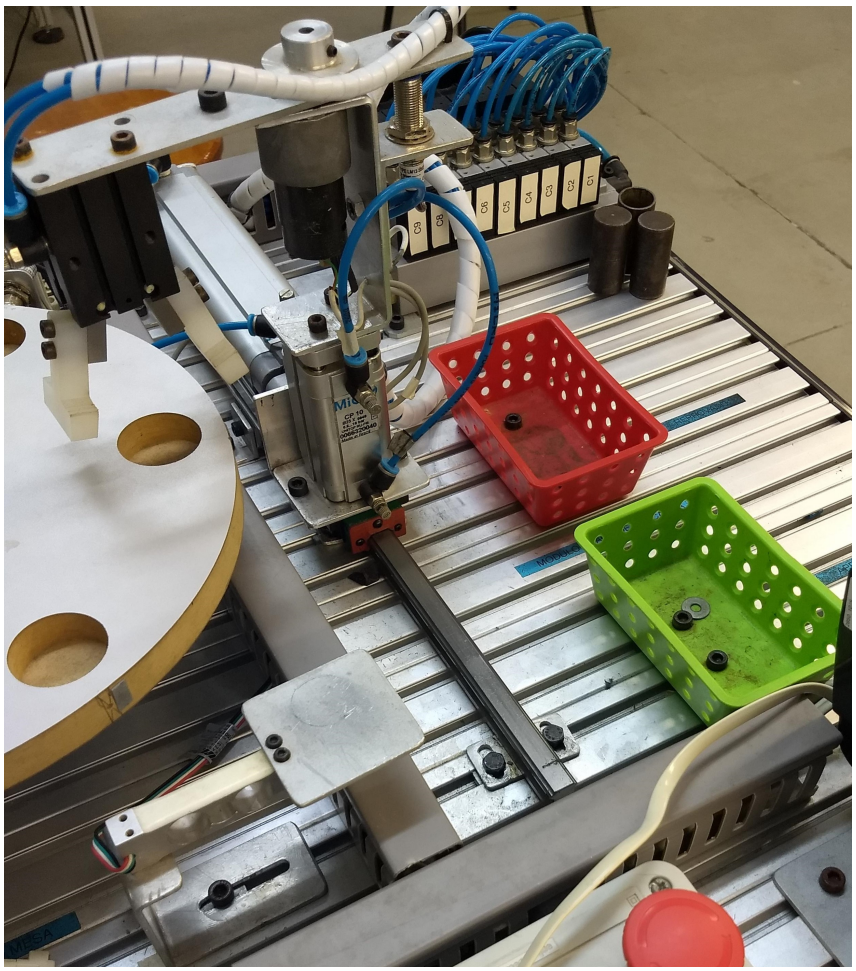
Figura 3.4 – Módulo mesa.



Fonte: Autor

- Módulo de descarga de peça.

Figura 3.5 – Módulo de descarga de peças.



Fonte: Autor

O módulo esteira é responsável por trazer a peça cilíndrica para o módulo de transferência, passando pelos sensores de capacitivo e óptico. O módulo de transferência então leva a peça até o módulo de manipulação cartesiana, que através da ação de uma garra, leva o cilindro até o módulo mesa. A mesa possui a função de levar a peça até a furadeira, onde ocorre o processo de usinagem, e posteriormente ao módulo de descarga. O módulo de descarga deve encaminhar a peça a balança e definir o destino final, o cesto verde ou vermelho dependendo da condição dela.

3.1.1 ATOS Expert BF

O CLP que controla a planta é o Atos Expert BF da fabricante Schneider Eletric. O modelo possui uma interface homem-máquina integrada com tela LCD para controle local e capacidade para 14 entradas e 10 saídas digitais, além de 2 entradas e saídas analógicas. A comunicação serial pode ser realizada através do protocolo APR03 em

um canal de comunicação RS-232, ou Modbus, em um canal de comunicação RS-485. A configuração do processo a ser executado é realizada através do *software* Atos A1 Soft.

A Figura 3.6 mostra o modelo do CLP Atos Expert BF e suas interfaces de comunicação, informando os protocolos utilizados em cada.

Figura 3.6 – Modelo do CLP utilizado mostrando as interfaces de comunicação.

Fonte: Autor

Com o objetivo de suprir todos os sensores e atuadores da planta na bancada, foram adicionados extensores aos módulos de I/O, devido a quantidade inicial de 14 entradas ser baixa, bem como a quantidade inicial de saídas. Estes extensores estão vinculados ao canal de comunicação RS-485, sendo internos a planta (bancada); o canal RS-232 é externo à planta e é utilizado para a programação via Atos A1 Soft.

Através da documentação de instalação da bancada, é possível verificar os endereços de cada entrada e saída do CLP, que podem ser visualizados na tabela abaixo, onde endereços físicos estão em decimais, mostrando o mapeamento de memória do CLP e endereços lógicos em hexadecimais, para uso na programação de comunicação.

Tabela 3.1 – Endereços de entradas e saídas da bancada.

Atuador / Sensor	Endereço (físico)	Endereço (lógico)
Sensor óptico	0	00 00
Sensor capacitivo	1	00 01
Sensor indutivo da mesa	2	00 02
Sensor indutivo do módulo descarga	3	00 03
Saída da balança	10	00 0A
Motor da esteira	1030	04 06
Motor da mesa	1031	04 07
Pistão de transferência	1024	04 00
Movimento vertical do manipulador cartesiano	1025	04 01
Movimento horizontal do manipulador cartesiano	1026	04 02
Garra do manipulador cartesiano	1027	04 03
Movimento vertical da furadeira	1028	04 04
Motor da furadeira	6447	19 2F
Movimento vertical do braço de descarga	1029	04 05
Movimento horizontal do braço de descarga	6449	19 30
Garra do braço de descarga	6448	19 31
Motor de giro do braço de descarga (anti-horário)	1032	04 08
Motor de giro do braço de descarga (horário)	1033	04 09

Fonte: Autor

3.1.2 Driver de comunicação serial Prolific

Em computadores mais novos, não há portas de comunicação direta com RS-232, sendo todas USB (“Universal Serial Bus”, Barramento Serial Universal) ou HDMI (“High-Definition Multimedia Interface”, Interface Multimídia de Alta Resolução). Ou seja, o computador não consegue se comunicar com a bancada diretamente, sendo necessário o uso de um *driver* e um conversor RS-232/USB para “transformar” dados do protocolo da porta USB ao RS-232 e vice-versa.

O *driver* em questão é o PL2303 Prolific, que foi instalado na máquina conectada ao CLP. Entretanto, as versões mais atuais de tal *driver* acabam não funcionando para o conversor utilizado, de forma que foi necessário colocar em uma versão anterior de 2008 ao invés da mais nova, de 2018.

O cabo conversor RS-232/USB utilizado pode ser visto na Figura 3.7.

Figura 3.7 – Cabo conversor USB/RS-232.

Fonte: Mercadão da Informática

3.1.3 Protocolo APR03

Como citado no início desta seção, o canal de comunicação externo (RS-232) utiliza um protocolo chamado APR03. Este protocolo é de propriedade da fabricante do CLP, a Schneider Eletric. É um protocolo assíncrono e *half-duplex* (ou seja, seu canal possui apenas uma via de comunicação, possibilitando que por ele esteja passando uma requisição ou uma resposta e nunca ambos ao mesmo tempo), sendo que o *baudrate* pode variar de 1200 a 57600 bps. As mensagens são padronizadas como “8N1”, ou seja, 8 bits de dados, nenhuma paridade e um bit de parada (*stop bit*), além disso, cada mensagem possui delimitadores de início e fim, sendo estes respectivamente “5A” e “5B” em hexadecimal.

O protocolo APR03 é baseado na topologia de mestre escravo, onde o próprio controlador é um dos 31 escravos possíveis. Os escravos podem somente transmitir quando recebem um comando do mestre.

O formato geral das mensagens no protocolo é composto de um *byte* denominado “f/e” e um *byte* de paridade. O *byte* f/e é constituído de 3 *bits* relativos a função da mensagem e 5 *bits* para endereçamento do nodo da rede. No caso da bancada, o APR03 considera como mestre o computador conectado ao CLP, e o CLP como o único escravo da rede, atribuindo a ele o endereço “00001” - os atuadores e sensores são considerados apenas na comunicação interna da bancada, onde o APR03 não atua. O endereço do mestre é atribuído como “00000”. A paridade é a paridade longitudinal (XOR) entre todos os *bytes* entre os delimitadores.

Resumidamente, para o canal de comunicação externa (RS-232), onde está o protocolo APR03, o CLP é escravo da comunicação. Já para a comunicação interna,

realizada por um canal de comunicação com RS-485 e protocolo Modbus, o CLP é mestre.

O APR03 define oito tipos de mensagem possíveis, sendo que para a presente aplicação, somente 4 são utilizadas, cada uma com sua estrutura própria, apresentada a seguir:

- PBLOC: requisição do mestre, indica um pedido de leitura de estado. O *byte* f/e para esta mensagem é “111 00001”, seguido do endereço do registrador requisitado (2 *bytes*), número de dados requeridos (1 *byte*) e a paridade longitudinal (1 *byte*);

Figura 3.8 – Estrutura de mensagem PBLOC.

Byte f/e	Conteúdo			Paridade
111eeeee	MSB ADD	LSB ADD	N bytes	P
1 byte	1 byte	1 byte	1 byte	1 byte

Fonte: (SCHNEIDER ELETRIC, 2010a)

Tabela 3.2 – Exemplos de mensagens PBLOC da bancada em *Buffer*.

Atuador / Sensor	Mensagem
Sensor óptico	5A E1 00 00 01 E0 5B
Sensor capacitivo	5A E1 00 01 01 E1 5B
Saída da balança	5A E1 00 0A 01 EA 5B
Motor da esteira	5A E1 04 06 01 E2 5B
Motor da mesa	5A E1 04 07 01 E3 5B
Pistão de transferência	5A E1 04 00 01 E4 5B

Fonte: Autor

- BLOC: envio de bloco do escravo, indica a resposta de um PBLOC, ou seja, o valor de leitura do estado atual de um registrador/atuador. O *byte* f/e para esta mensagem é “100 00001”, seguido do endereço do registrador requisitado (2 *bytes*), quantidade de registradores requisitados (1 *byte*), valores de estado de cada um e a paridade longitudinal (1 *byte*);

Figura 3.9 – Estrutura de mensagem BLOC.

Byte f/e	Conteúdo				Paridade
100eeeee	MSB ADD	LSB ADD	N bytes	dado 1 dado 2 ...dado n	P
1 byte	1 byte	1 byte	1 byte	n bytes	1 byte

Fonte: (SCHNEIDER ELETRIC, 2010a)

Tabela 3.3 – Exemplos de mensagens BLOC da bancada em *Buffer*.

Atuador / Sensor	Mensagem (estado 0)	mensagem (estado 1)
Sensor óptico	5A 80 00 00 01 FF 7E 5B	5A 80 00 00 01 FE 7F 5B
Sensor capacitivo	5A 80 00 01 01 FF 7F 5B	5A 80 00 01 01 FE 7E 5B
Saída da balança	5A 80 00 0A 01 FF 74 5B	5A 80 00 0A 01 FE 73 5B
Motor da esteira	5A 80 04 06 01 FF 7C 5B	5A 80 04 06 01 FE 7D 5B
Pistão de transferência	5A 80 04 00 01 FF 7A 5B	5A 80 04 00 01 FE 7B 5B

Fonte: Autor

- BYT: envio de *byte* do mestre, indica um dado a ser escrito. O *byte* f/e para esta mensagem é “011 00001”, seguido do endereço do registrador requisitado (2 *bytes*), dado a ser escrito e a paridade longitudinal (1 *byte*);

Figura 3.10 – Estrutura de mensagem BYT.

Byte f/e	Conteúdo			Paridade
011eeeee	MSB ADD	LSB ADD	DATA	P
1 byte	1 byte	1 byte	1 byte	1 byte

Fonte: (SCHNEIDER ELETRIC, 2010a)

Tabela 3.4 – Exemplos de mensagens BYT da bancada em *Buffer*.

Atuador	Mensagem (estado 0)	Mensagem (estado 1)
Motor da esteira	5A 61 04 06 FF 9C 5B	5A 61 04 06 FE 9D 5B
Pistão de transferência	5A 61 04 00 FF 9A 5B	5A 61 04 00 FE 9B 5B
Garra do manipulador cartesiano	5A 61 04 03 FF 99 5B	5A 61 04 03 FE 98 5B

Fonte: Autor

- ACK: confirmação de mudança de estado, enviado pelo escravo após uma escrita. O *byte* f/e desta mensagem é “110 00001”, seguido pela paridade longitu-

dinal (1 *byte*). A mensagem ACK possui sempre o mesmo formato, sendo este “5A C0 C0 5B”

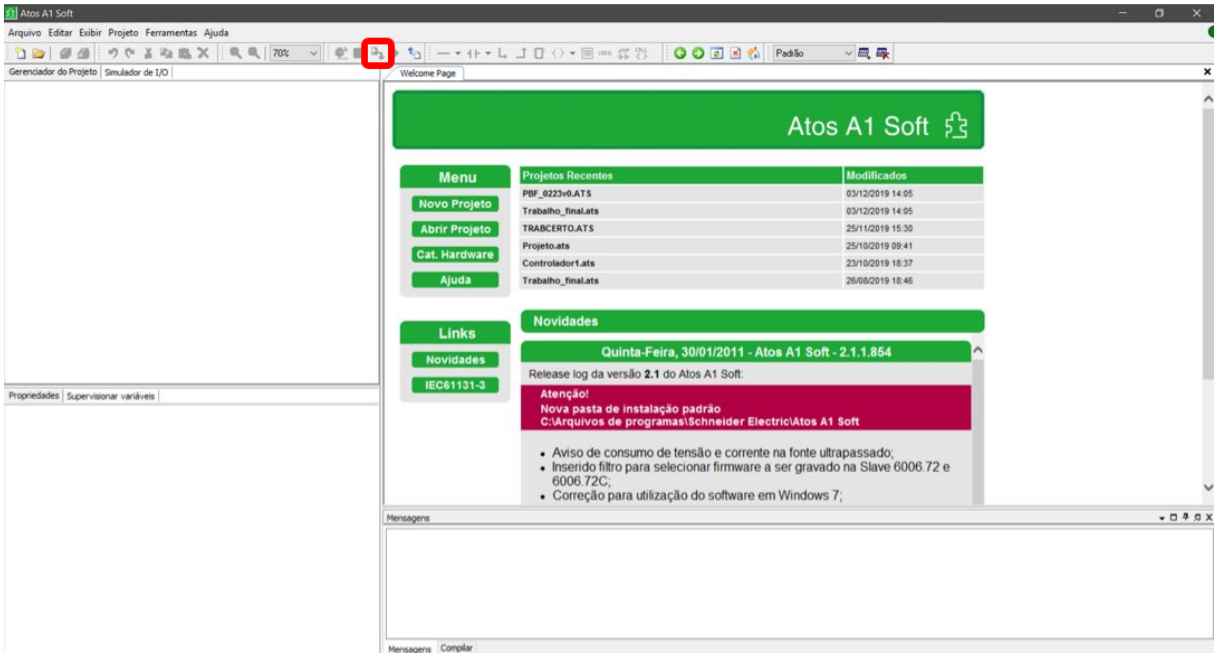
A comunicação no protocolo APR03 se dá na seguinte maneira: o mestre sempre envia um PBLOC, tanto para escrita quanto pra leitura; o escravo responde com um BLOC, expressando o valor dos registradores. Caso a requisição for uma leitura, a comunicação termina aqui, mas no caso de escrita, o mestre envia um BYT, com o novo valor do registrador e é respondido com um ACK.

3.2 ATOS A1

O Atos A1 Soft é um *software* de programação para os CLP’s da linha Atos, desenvolvido e disponibilizado para *download* pela Schneider Eletric. Nele, além da programação em *Ladder*, é possível simular sem enviar ao CLP e ainda, após o envio, monitorar as variáveis de interesse do usuário. Também é possível coletar informações do CLP e “puxar” a programação atual nele, para fazer modificações ou verificar erros.

A programação do *software* gera um arquivo de extensão “.ats” e diversos binários. É possível utilizar as duas opções para encaminhar para o CLP: enviar o arquivo fonte “.ats” diretamente ou enviar um arquivo binário que é criado após a compilação do *Ladder*. A Figura 3.11 mostra a interface do Atos A1, com o botão de enviar arquivo para o CLP destacado.

Figura 3.11 – Interface do Atos A1.



Fonte: Autor

3.3 LINGUAGEM DE PROGRAMAÇÃO

A fim de criar a aplicação, diversas linguagens devem ser utilizadas, observando a especialidade de cada uma. É necessário haver uma linguagem voltada ao cliente (*client-side*) e uma voltada às funções do servidor da aplicação (chamada de *server-side*).

3.3.1 *HTML* e *CSS*

O HTML (*Hypertext Markup Language*) é a linguagem de programação padrão no *design* de *websites*. Documentos com formato ".html" podem ser interpretados em navegadores através do protocolo HTTP ou HTTPS. Com o HTML é possível criar páginas na Internet, ligações para outras páginas, formulários para envio de mensagens e dados, além de acoplar outros tipos de arquivo, como códigos em JavaScript (para processar informações da página), arquivos multimídia (imagens e videos) e códigos com formato ".css".

A linguagem CSS (*Cascading Style Sheets*, do inglês Folhas de Estilo em Cascata) é utilizada em conjunto com o HTML para a criação de páginas na *Web*. A função do CSS é simplesmente melhorar a organização e estilo do HTML, sendo localizado em um arquivo separado ou no próprio código com o fim de realizar um *design* melhorado na página. Como fazer o *design* de páginas muito complexas pode ser demorado, *frameworks* - um conjunto de códigos e funções que podem ser utilizados de maneira genérica - foram criados para facilitar o desenvolvimento dessas páginas. Um desses *frameworks* é o Bootstrap.

O Bootstrap é considerado uma "biblioteca de componentes *Front-End*" voltada ao desenvolvimento de páginas responsivas e sistemas móveis (BOOTSTRAP, 2019). Basicamente, é uma coleção de arquivos em CSS e Javascript que possuem a capacidade de tornar fácil a melhoria de *designs* em páginas e ainda promover diversas funções extras, como escalar uma página para a resolução em dispositivos móveis e realizar animações de botões, por exemplo.

3.3.2 Linguagem *Server-side*

Enquanto o HTML e o CSS servem como linguagens *client-side*, há duas linguagens amplamente usadas no lado do servidor, sendos estas o PHP e o JavaScript. O PHP (acrônimo para *PHP: Hypertext Preprocessor*, antigamente denominado *Personal Home Page*) é uma linguagem voltada ao lado do servidor capaz de gerar as

páginas do lado do cliente e processar o conteúdo dentro da própria página, com o fim de tornar o conteúdo dinâmico. O JavaScript, por outro lado, foi desenvolvido inicialmente para o *client-side*, sendo "embutido" em páginas HTML para o controle de funções e movimentações. Posteriormente, com o fim de padronizar uma linguagem para ambos os lados, foram desenvolvidas extensões *server-side* da linguagem, adicionando uma ampla dinâmica e facilitando o desenvolvimento de novas aplicações, dado que era necessário aprender apenas o JavaScript. Das extensões *server-side* criadas, a que mais obteve êxito em desenvolver novas funcionalidades foi o chamado Node.js, que devido à sua aplicabilidade foi optado para o presente trabalho.

3.3.3 Node.js

O Node.js é um interpretador de JavaScript (ou seja, usa o próprio JavaScript com código e sintaxe), porém possibilita o uso de certas API's (*Application Programming Interfaces*) chamadas de módulos ou pacotes. Estes pacotes são gerenciados pelo NPM (*Node Package Manager*) e são desenvolvidos pela comunidade. Os módulos são instalados com o comando "npm install [nome do pacote]" e podem ser chamados diretamente no algoritmo de JavaScript através de "require('nome do pacote')".

Para executar um código usando o interpretador Node.js, é necessário chamar o comando "node" e o arquivo de *script* ".js" a ser executado. Isto pode ser feito no terminal de comando da máquina, através de "node [nome do arquivo].js".

Outras características pertinentes ao Node.js são relativas a forma como ele trabalha. Além de ser assíncrono, ele é baseado em eventos e utiliza uma única *thread* para realizar suas funções, deixando as requisições em fila para execução. Isto torna o *software* baseado em Node.js mais leve, mas o impede de realizar funções relativas a algoritmos complexos que consumem muita memória, como edição de imagens, por exemplo.

Diversos módulos foram utilizados no projeto, alguns até sobrepondo as funções de outros. Os principais módulos usados e sua determinada função para a aplicação são:

- Http: responsável por indicar uma porta que fica escutando requisições. Ao usar a função "http.listen(100)", o servidor está ligado na porta informada (no caso, a porta 100);
- Express: responsável pelas funções de renderizações de páginas, ou seja, indica qual a resposta para uma determinada requisição do cliente, tanto para páginas quanto para métodos;

- Sockets.io: responsável por estabelecer uma conexão com *sockets*, tornando possível uma personalização maior da comunicação;
- Serialport: o JavaScript básico não permite conexões com o *hardware*, incluindo as portas seriais do computador. O módulo Serialport então, torna possível esta comunicação, tanto no envio quanto no recebimento de mensagens via porta serial;
- FS: também chamado de *File System*, o módulo fs permite o servidor acessar pastas e arquivos do computador o qual está inserido, tanto para leitura quanto para escrita;
- Nodemailer: responsável pelo envio automático de *e-mails*. Por padrão, utiliza o protocolo SMTP, mas pode ser configurado para o protocolo IMAP;
- Child-Process: dentre diversas funções especiais, o módulo *Child-Process* torna possível a execução de arquivos de extensão ".exe" via código JavaScript;
- Mongoose: o módulo Mongoose foi criado para facilitar a conexão e comunicação com o banco de dados MongoDB, possibilitando escrever as *queries* (inquisições, linhas de códigos em bancos de dados que realizam funções de chamadas de dados ou alteração destes) direto no código;
- Passport: responsável pela autenticação de usuários através da confirmação de *login* vinculada a um banco de dados. O módulo passport pode ainda ser usado, através de extensões, para utilizar bancos de dados de outras plataformas de redes sociais, como o *Facebook* ou *Twitter*.

3.3.4 Python

O Python, licenciado pela Python Software Foundation, é uma linguagem de alto nível de fácil aprendizado e *Open-Source*, ou seja, é aberta a desenvolvimento em comunidade e o uso é irrestrito. Devido a sua facilidade de aprendizado e uso, é amplamente usada atualmente para as mais diversas funções, como desenvolvimento *Web*, acesso a banco de dados, programação em rede e desenvolvimento de GUI's (*Graphical User Interfaces*, ou Interface Gráfica do Usuário).

Semelhante ao Node.js, o Python também possui "extensões" para ampliar seu uso, através de bibliotecas. Para o presente trabalho, duas de suas bibliotecas são utilizadas:

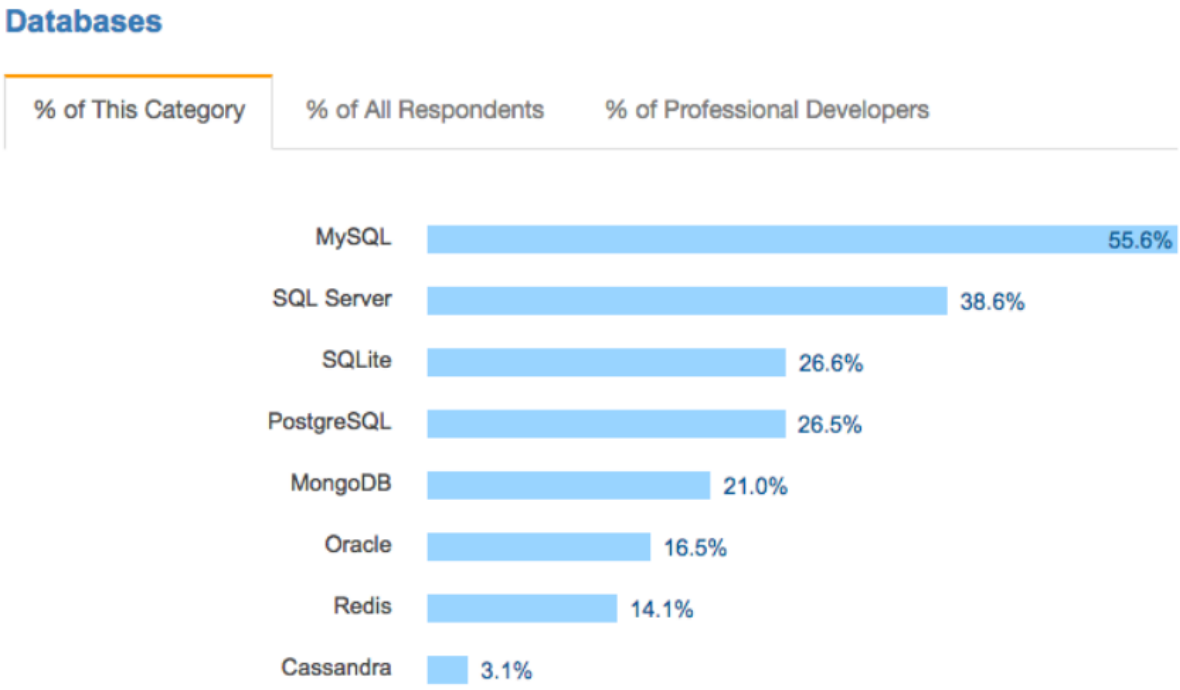
- Pyautogui: permite criar um código capaz de realizar funções do *mouse*, teclado e monitor, tais como cliques, digitações e capturas de tela, de maneira personalizada;
- Pyinstaller: biblioteca responsável por transformar um arquivo Python (de extensão ".py") em um arquivo executável, passível de ser usado por outras aplicações.

3.4 MONGODB

Escrito na linguagem C++, MongoDB é um banco de dados de código aberto, gratuito e de alta performance que armazena arquivos semelhantes ao formato ".json", o que possibilita que ele veja uma inserção nova como um objeto com seus devidos atributos, diferente de bancos de dados tradicionais que usam o sistema linha-coluna. Uma de suas principais vantagens é em relação a interoperabilidade, principamente com o JavaScript, sendo este capaz de ser usado em consultas personalizadas. O JavaScript realiza um comando e este é enviado direto ao banco de dados para ser executado.

MongoDB é um banco de dados popular para desenvolvimento de aplicações *online*, com baixa latência, alta vazão e disponibilidade (UMBLER, 2017). Entre os bancos não-relacionais (também chamados de NoSQL), o mongo DB é o mais utilizado, como pode ser verificado na Figura 3.12. Devido a sua interoperabilidade com JavaScript e facilidade de uso com Node.js através do módulo Mongoose, o MongoDB foi optado na realização do projeto.

Figura 3.12 – Comparação de uso de bancos dados em aplicações online.



Fonte: (UMBLER, 2017)

Apesar da escolha do projeto, é importante salientar que bancos não-relacionais podem apresentar problemas de atomicidade (entrega de informações parciais ao invés de completas) caso haja muitas requisições simultâneas. Apesar de não fornecer muitas vantagens neste trabalho, um banco relacional pode ser utilizado caso tal problema comece a ocorrer.

3.5 YAWCAM

Para a monitoração do laboratório remoto, foi optado pela visualização através de uma *stream* de vídeo com câmeras IP. Uma maneira de atingir este objetivo é transformar uma *webcam* comum em uma câmera IP, o que pode ser feito utilizando o *software* Yawcam (*Yet Another WebCAM software)*. Ele possui uma interface de simples configuração, onde é possível selecionar a porta para cada tipo de uso que suporta.

O endereço IP que o *software* utiliza para a câmera é o próprio da máquina, onde é apenas configurável a porta em que será transmitida a *stream*. Com a porta informada, o *software* abre um servidor, com uma página HTML transmitindo o vídeo. Ainda pode ser configurado o número máximo de espectadores da transmissão, ou seja, se o número for configurado em 1, caso uma segunda pessoa acesse a mesma

página, não poderá ver o vídeo.

O modelo de *webcam* utilizado pode ser visualizado na Figura 3.13.

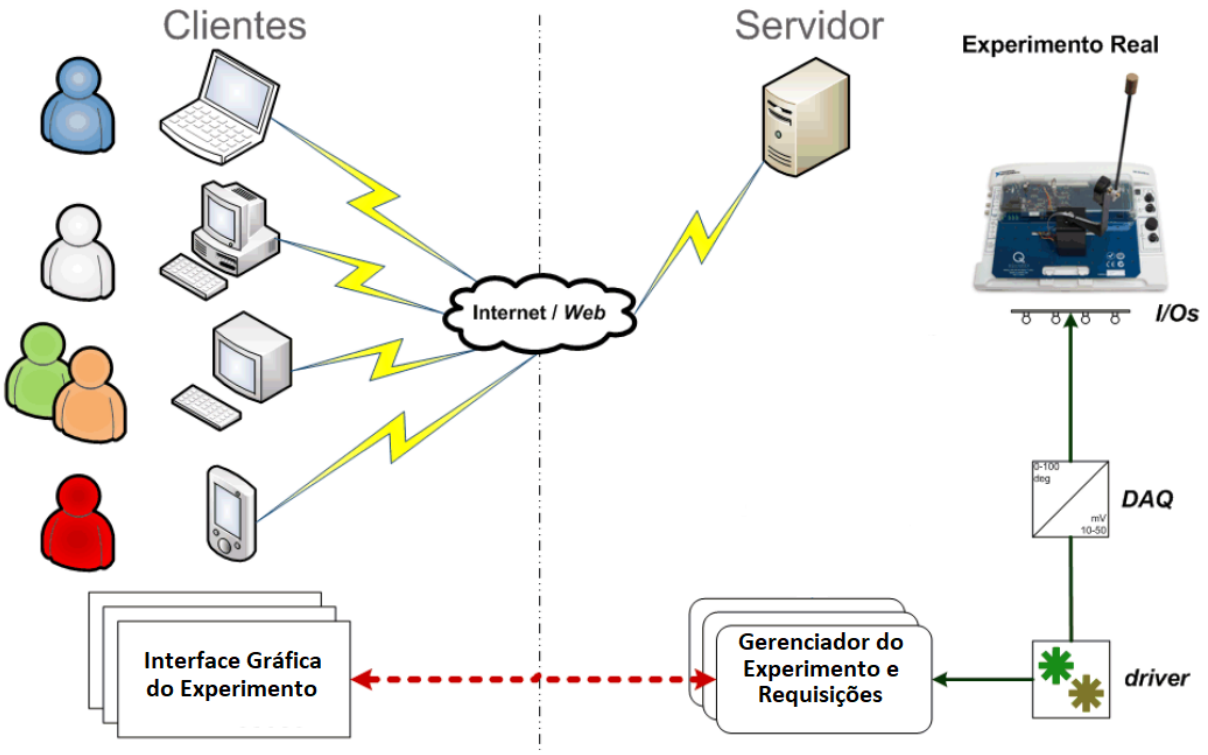
Figura 3.13 – Webcam utilizada, modelo da empresa Genius.

Fonte: Autor

3.6 LABORATÓRIO REMOTO

O projeto do laboratório remoto funciona como um *website* através de páginas abertas em um navegador. Cada página se comporta como um cliente, enquanto o servidor é executado no terminal de comando do computador em que está inserido. As páginas se comunicam com o servidor através de *sockets* e o servidor se comunica com a bancada através da porta serial.

Figura 3.14 – Diagrama de comunicação do laboratório remoto.



Fonte: Adaptado de Schaf (2011)

A Figura 3.14 mostra o funcionamento de um laboratório remoto genérico, onde pode ser visualizada cada etapa da comunicação completa. Os clientes possuem uma interface gráfica do experimento e enviam sinais pela Internet ao servidor. O servidor gerencia as requisições dos clientes e as envia ao experimento, passando por *drivers* de comunicação das portas seriais e circuitos de condicionamento de sinais, a fim de conciliar seus níveis adequados.

Vale ser apontado que a metodologia e arquitetura desenvolvidas para o laboratório remoto não são novas. Saygin e Kahraman (2004) propuseram um laboratório remoto onde era possível programar e controlar um CLP baseado na *Web*, possibilitando seu uso por alunos na prática de exercícios educacionais remotos. Tal trabalho serviu de inspiração para diversas pesquisas demonstrando a efetividade da tecnologia, como mostra Bellmunt et al. (2006), onde através dos conceitos levantados por Saygin e Kahraman foi criado um curso operacional para o laboratório remoto com CLP. Entretanto, em tal trabalho não houve computadores atuando como servidores, mas sim uma ligação direta via Internet dos usuários (alunos) ao CLP.

4 RESULTADOS

Nesta seção serão apresentados os resultados do projeto. O *design* de páginas e suas funções, bem como a operação completa do servidor. Também será informado o funcionamento da aplicação através de diagramas e como ocorre a comunicação com a bancada.

4.1 COMUNICAÇÃO COM A BANCADA

Inicialmente, foi pensada a maneira como os arquivos de programação gerados pelo *software* Atos A1 poderiam ser enviados ao CLP via JavaScript. Através de um *software serial sniffer* - um utilitário que captura pacotes de dados seriais - foram forçados os arquivos pelo Atos A1, o que gerou os pacotes apresentados resumidamente na Figura 4.1.

Figura 4.1 – Pacotes de dados enviados pelo Atos A1.

```
5C 01 04 E3 01 5D

5C 01 0B E7 41 5D

5C 01 02 E1 81 5D

5C 01 0B E7 41 5D

5C 01 0B E7 41 5D

5C 01 06 20 10 00 00 00 00 00 11 B7 D9 5D

5C 01 0E 00 20 00 00 00 01 00 E4 FF 49 6B 5D

5C 01 01 00 20 00 00 00 C0 DC 00 01 00 20 27 85
F7 80 00 00 00 98 00 00 00 FF FF FF FF FF FF FF
FF E0 00 00 00 00 00 01 00 FF FF FF FF 40 02 00
00 FF FF FF FF FF FF FF FF 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 F4 01 00 00 FF FF FF
FF FF FF FF FF D4 01 00 00 84 01 00 00 3C 02 00
00 FF FF FF FF 14 02 00 00 01 00 00 00 FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF 75 3D E8 22 10 00 01 00 00 18 00 10 00 02 00
08 00 01 00 02 00 01 00 20 01 00 14 00 00 00 00
00 01 00 00 00 00 00 FF FF AC 00 00 00 34 00 30
00 31 00 33 00 33 00 46 00 50 0D 77 5D
```

Fonte: Autor

Pela Figura, é possível verificar que o protocolo utilizado para arquivos de programação do CLP é diferente do APR03, visto que seus delimitadores de início e fim são “5C” e “5D”, respectivamente. Com isso, começou-se a fazer testes a fim de en-

contrar semelhanças entre os arquivos enviados, descobrindo a natureza do protocolo. Desta forma, descobriu-se que as mensagens do *header* e do *footer* são relativas a funções de formatação, preparação e conclusão de recebimento do arquivo, sendo suas funções as seguintes:

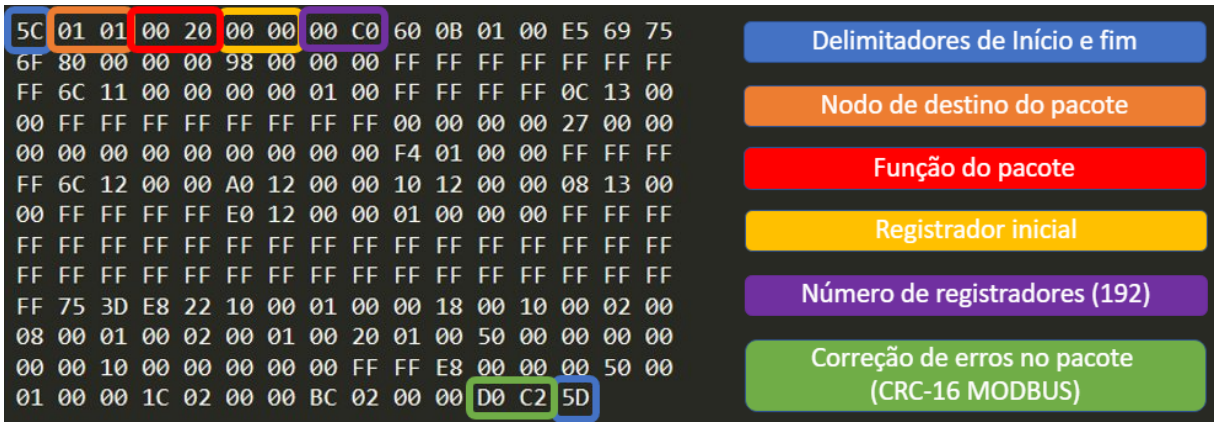
Tabela 4.1 – Função de pacotes do *header* e *footer*.

Função	Pacote
Entrar no modo programação do CLP	5C 01 02 E1 81 5D
Prepara o CLP para formatar programação antiga	5C 01 06 20 10 00 00 00 00 00 11 B7 D9 5D
Informa os registradores que devem ser apagados	5C 01 0E 00 20 00 00 00 01 00 E4 FF 49 6B 5D
Conclui o processo de gravação no CLP	5C 01 07 00 20 00 00 20 10 00 00 00 01 00 E4 15 49 5D
Reinicia o CLP	5C 01 0C 25 00 5D

Fonte: Autor

As três primeiras funções descritas na Tabela 4.1 fazem parte do *header*, e as duas últimas, do *footer*. Após o *header*, as mensagens que possuem o conteúdo do arquivo binário são enviadas e através da comparação delas que foi possível identificar a estrutura dos pacotes, que pode ser visualizada na Figura 4.2.

Figura 4.2 – Estrutura dos pacotes de programação.



Fonte: Autor

O modelo de estrutura do pacote apresentado na Figura 4.2 é mantido por todas as mensagens que possuem o conteúdo do arquivo binário, sendo este dividido em pacotes de 192 *bytes* cada. Também vale ser apontado que pacotes vazios - cujo conteúdo são apenas *bytes* “FF” - não são enviados ao CLP. Com isso, as mensagens

com o arquivo binário tem, em sua totalidade, 204 *bytes*.

Entretanto, alguns pacotes possuem *bytes* adicionais em seções aparentemente aleatórias, que não foram identificados pelo Autor. Tais *bytes* adicionais impediam o envio correto do arquivo binário ao CLP através da reconstrução das mensagens por JavaScript, sendo então procurada outra solução para o envio de arquivos ao CLP.

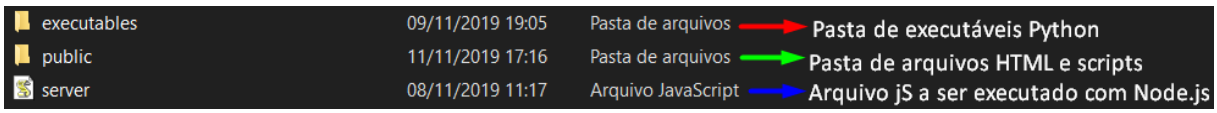
A solução foi controlar a máquina do servidor a distância, com movimentos do *mouse* e teclado através da biblioteca Python “pyautogui”. Tais movimentos e cliques seriam orientados a selecionar os arquivos corretos e enviar à bancada através do próprio Atos A1 instalado no servidor. Mesmo com tal biblioteca, ainda seria necessária uma maneira de chamar o código Python diretamente do servidor escrito em JavaScript. Para tanto, duas condições deveriam ser saciadas: transformar o arquivo Python em um executável e chamá-lo no JavaScript, colocando em uso então a biblioteca “pyinstaller” e o módulo “child-process”.

Desta forma, para enviar os arquivos binários ao CLP foi criado um executável baseado em Python e sempre que o cliente quiser testar sua programação, ele deve encaminhar o arquivo ao servidor e este chamar tal executável.

O servidor do laboratório remoto pode se comunicar com a bancada de duas formas: através do módulo “child-process” chamando os executáveis baseados em Python (para a interface do Atos A1); ou ainda através de outro módulo, o “serialport”. Este último envia sequências de *bytes* de dados diretamente ao CLP, o que torna a comunicação mais rápida quando utilizado. Porém, dado problema com *bytes* desconhecidos, ele somente pode ser aplicado na situação da seção 4.2.6.

Ambos os módulos para comunicação da bancada foram chamados no arquivo “.js” do servidor e os executáveis dispostos em uma pasta separada, a ser chamada pelo “child-process”. A Figura 4.3 mostra a organização do servidor.

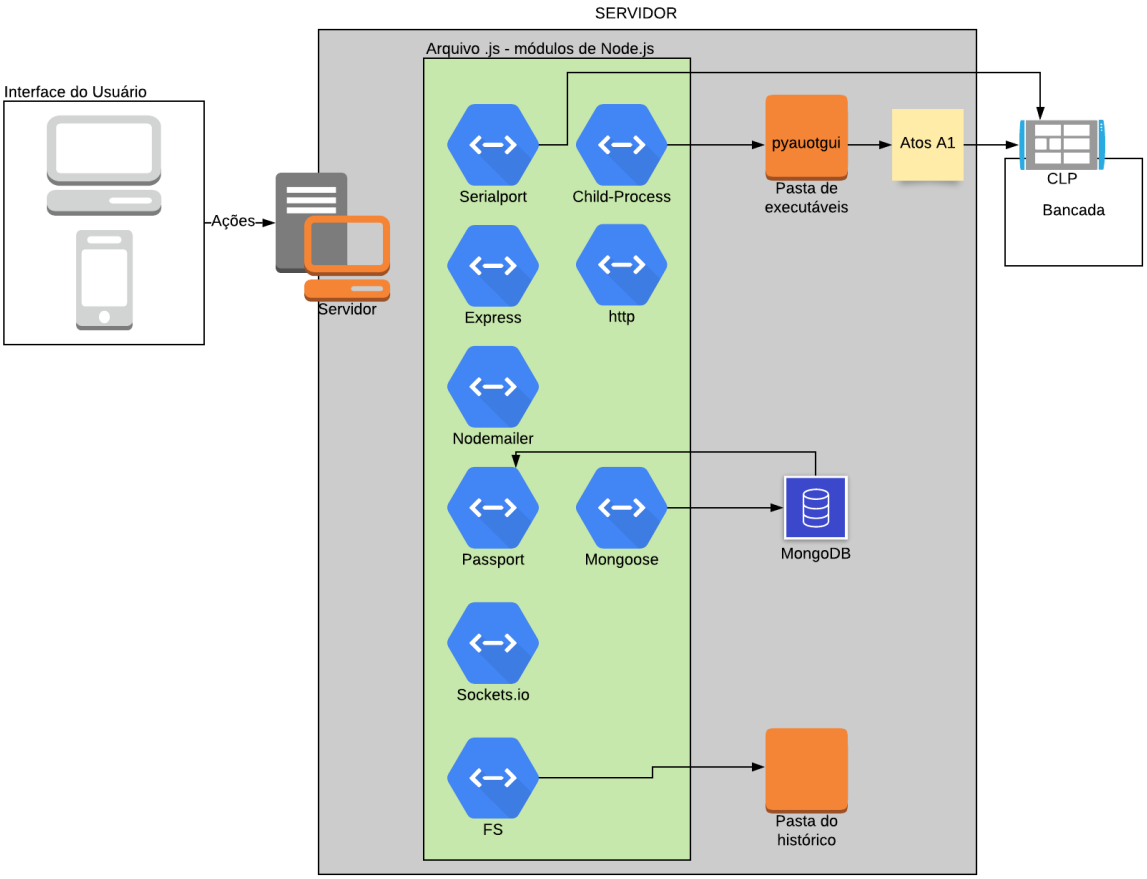
Figura 4.3 – Pasta do servidor.



Fonte: Autor

Vale salientar que o arquivo “.js” da Figura 4.3 não é um JavaScript comum, uma vez que possui chamadas de módulos e funções de Node.js e caso for utilizado por um interpretador comum, resultará em erros de sintaxe.

Figura 4.4 – Servidor com módulos de Node.



Fonte: Autor

A Figura 4.4 mostra a localização e funcionamento geral do servidor. A interface de usuário se comunica com o servidor (via *sockets*) e este possui o arquivo ".js", onde são chamados os módulos de Node. Enquanto alguns realizam funções internas da aplicação, o módulo "FS" tem a função de escrita/leitura da pasta do histórico de envios do usuário (funcionamento descrito na seção 4.2.7). O módulo "mongoose" e "passport", responsáveis pelo controle de acesso se comunicam com o banco de dados MongoDB. O módulo "serialport" se comunica com a bancada de maneira direta e o "child-process" de maneira indireta, passando por executáveis baseados em Python, cuja função é realizar comandos na interface do Atos A1.

4.2 PÁGINAS

Com o problema do envio de arquivo resolvido, as páginas HTML foram criadas utilizando CSS e Bootstrap e separadas por função, visando um *design* e funciona-

lidade favorável ao usuário. As páginas desenvolvidas, bem como sua função serão descritas nesta seção. Todas as páginas foram desenvolvidas com um menu de navegação, capaz de redirecionar para outras páginas do servidor e responsivas para dispositivos móveis.

4.2.1 Home

Desenvolvida com a função de servir como a página inicial da aplicação assim que acessado o endereço IP e porta do servidor, a página Home é apenas um intermédio para a aplicação. Porém, caso o usuário não esteja em sessão no *website*, a página Home só permitirá o acesso à página de Login e de Histórico de Envios. Caso já haja alguém com sessão iniciada na aplicação, a Home ainda exibirá um aviso ao cliente, mostrando que já há um usuário em sessão e informando o nome do usuário.

A Figura 4.5 mostra a página Home, com o menu de navegação. Caso um usuário estiver com sessão ativa, uma mensagem aparece acima do botão *Login*, informando o nome do usuário.

Figura 4.5 – Página Home.

Fonte: Autor

4.2.2 Login

Uma vez que o objetivo da aplicação é programar e testar um único CLP a distância, é necessário que um único usuário por vez possa acessar as funcionalidades do servidor relacionadas a bancada. Para isso, um sistema de controle de acesso foi criado, utilizando o banco de dados MongoDB para guardar dados de *login* como *e-mail*, nome de usuário e senha. Para acessar o servidor, uma página de Login foi criada, a fim de permitir estabelecer sessões dos clientes.

A comunicação do MongoDB com o servidor se deu através do módulo “Mongoose” e a autenticação de usuários para o estabelecimento de sessões com *cookies* foi realizada através do módulo “Passport”.

Outro fator sobre o controle de acesso é no caso de um usuário esquecer de fechar a sessão (*logout*). Neste caso, após um tempo pré-estabelecido, contado a partir da última vez que o usuário fechou uma página, o servidor força o fechamento da sessão.

A página de *Login* pode ser visualizada na Figura 4.6. Caso o início de sessão falhe, um alerta na página aparece, informando que usuário e/ou senha estão errados.

Figura 4.6 – Página Login.

Fonte: Autor

4.2.3 Registro de Novo Usuário

Caso o cliente ainda não possuir dados de usuário, é possível criar um novo usuário a partir da página de Registro. Um novo usuário será criado uma vez que informado um nome de usuário, uma senha, um *e-mail* e um código de verificação,

que será enviado ao *e-mail* registrado através do módulo “Nodemailer”. Para esta ação ser realizada, foi criado uma conta para a bancada, a fim de ter acesso ao serviço de *e-mails* da Google, o Gmail.

A página de registro desenvolvida pode ser observada na Figura 4.7

Figura 4.7 – Página de Registro.

Fonte: Autor

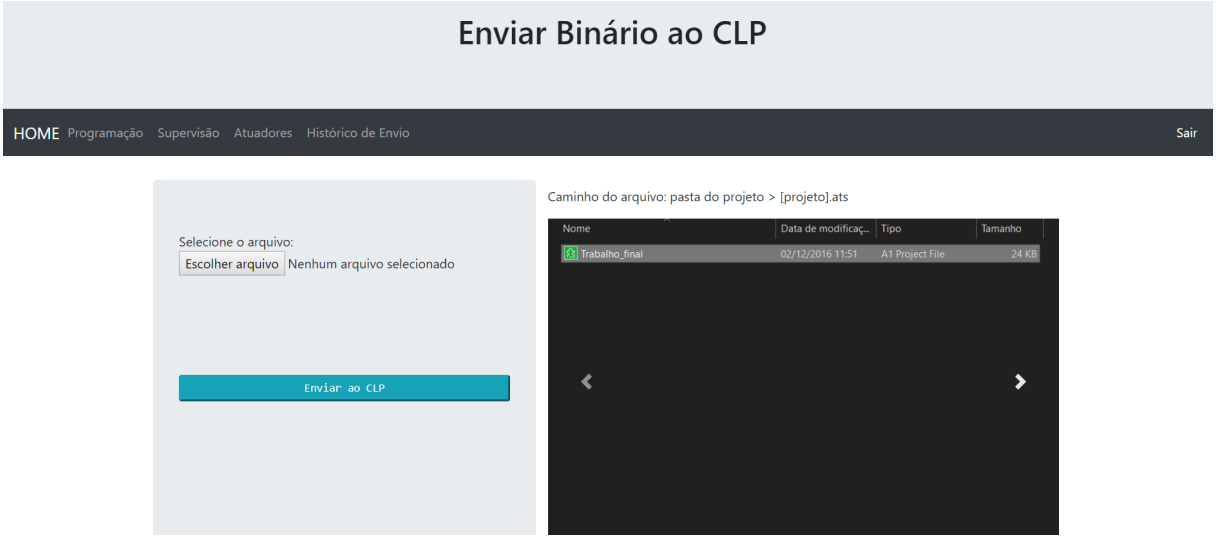
4.2.4 Programação

A página de programação possui imagens para auxiliar o usuário a encontrar o arquivo correto para o envio. Somente arquivos com extensão “.ats” são aceitos

pela página. Também possui uma ferramenta de *upload* simples, que lê o arquivo do cliente e o envia por *sockets* ao servidor, onde é salvo em uma pasta específica. O botão que envia o arquivo ao CLP está nesta página, mas inicialmente desabilitado, sendo apenas ativado quando o arquivo ".ats" for enviado.

A Figura 4.8 mostra a página de programação, com uma das imagens que mostra o caminho até o arquivo e seus botões de envio (à esquerda).

Figura 4.8 – Página de Programação.



Fonte: Autor

4.2.5 Supervisão

Após o envio da programação, o usuário deve testar o processo programado. Na página de Supervisão, um sistema SCADA foi desenvolvido, onde é possível ver a planta em vídeo e conferir variáveis de interesse do programa, configuradas pelo usuário em seu arquivo de projeto. Há botões na página que simulam os botões físicos de Liga, Desliga e Emergência. Quando o cliente aperta em qualquer um, uma mensagem do protocolo APR03 é enviada à bancada contendo um pulso colocando temporariamente o valor do registrador do botão em 1. Entretanto, para o processo funcionar completamente, é necessário haver uma maneira de forçar sensores em estados diferentes remotamente, de modo que o CLP identifique onde está a peça no processo, mesmo sem ela estar lá fisicamente. Forçar os sensores poderia ser realizado através do protocolo APR03, porém, quando a bancada entra em operação, com o fim de mostrar as variáveis de interesse ao usuário, o canal serial é ocupado pelo Atos A1, impedindo o acesso ao canal de comunicação para outros programas (como o desenvolvido em JavaScript). A solução encontrada foi utilizar novamente a

biblioteca Python “pyautogui” e realizar as funções diretamente no Atos A1 para forçar tais elementos.

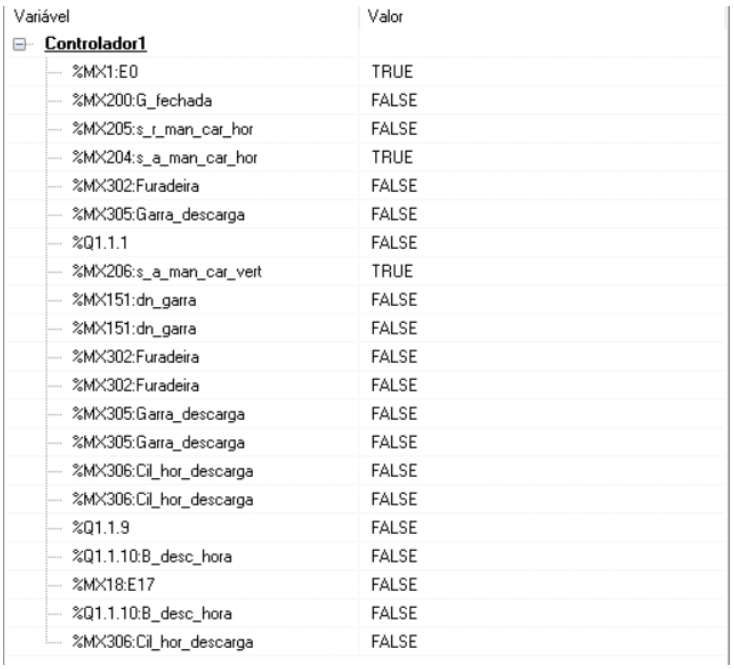
Ainda nesta página, uma imagem é mostrada após os botões e o vídeo. A imagem trata-se de uma captura de tela, repetida a cada 250 ms, na região onde estão as variáveis de interesse do usuário. Desta forma, utilizando a própria interface do Atos A1 é possível verificar o estado dos atuadores, sensores e memória interna do CLP, conforme configurados pelo usuário.

A Figura 4.9 mostra a página de Supervisão, com todos os sensores não forçados e com a interface do Atos A1 vinda da captura de tela abaixo da câmera do experimento e dos botões. A Figura 4.10 mostra um exemplo da captura de tela que está na página.

Figura 4.9 – Página de Supervisão.

Fonte: Autor

Figura 4.10 – Exemplo de captura de tela da supervisão das variáveis no Atos A1.



Variável	Valor
Controlador1	
%MX1:E0	TRUE
%MX200:G_fechada	FALSE
%MX205:s_r_man_car_hor	FALSE
%MX204:s_a_man_car_hor	TRUE
%MX302:Furadeira	FALSE
%MX305:Garra_descarga	FALSE
%Q1.1.1	FALSE
%MX206:s_a_man_car_vert	TRUE
%MX151:dn_garra	FALSE
%MX151:dn_garra	FALSE
%MX302:Furadeira	FALSE
%MX302:Furadeira	FALSE
%MX305:Garra_descarga	FALSE
%MX305:Garra_descarga	FALSE
%MX306:Cil_hor_descarga	FALSE
%MX306:Cil_hor_descarga	FALSE
%Q1.1.9	FALSE
%Q1.1.10:B_desc_hora	FALSE
%MX18:E17	FALSE
%Q1.1.10:B_desc_hora	FALSE
%MX306:Cil_hor_descarga	FALSE

Fonte: Autor

4.2.6 Atuadores (Emergência)

Caso o usuário tenha feito algum erro durante o processo, é necessária uma maneira da bancada voltar a condição inicial de operação, para que outros testes sejam realizados e não ocorra acidentes. A página de Atuadores possui um botão que envia um arquivo “base” - um arquivo que possui a programação “de fábrica” da bancada, com todos os módulos funcionando corretamente. Após o envio deste arquivo através do botão, outros botões da página são ativados, capazes de mudar os estados dos atuadores, bem como recalibrar motores de giro da mesa e do braço de descarga. Ao pressionar qualquer um destes botões, uma mensagem é enviada ao servidor e este, através do módulo “serialport”, envia um *buffer* construído com o protocolo APR03 informando o atuador a ser modificado.

A Figura 4.11 mostra a página de Atuadores, posterior ao envio do arquivo base.

Figura 4.11 – Página de Atuadores.

Fonte: Autor

4.2.7 Histórico de Envio

Quando o usuário envia seu arquivo na página de programação, uma pasta com seu nome é criada no servidor, contendo tal arquivo. Ao mesmo tempo, um registro é realizado em um arquivo de texto, responsável por armazenar um histórico de envios, informando o nome do usuário e o horário enviado, com data. A página do histórico mostra até os últimos 50 envios, porém o arquivo de texto que armazena tais informações nunca é apagado.

O histórico de envios pode ser pode ser visualizado na Figura 4.12, onde são observadas 4 entradas de arquivo, informando o usuário e a data do envio.

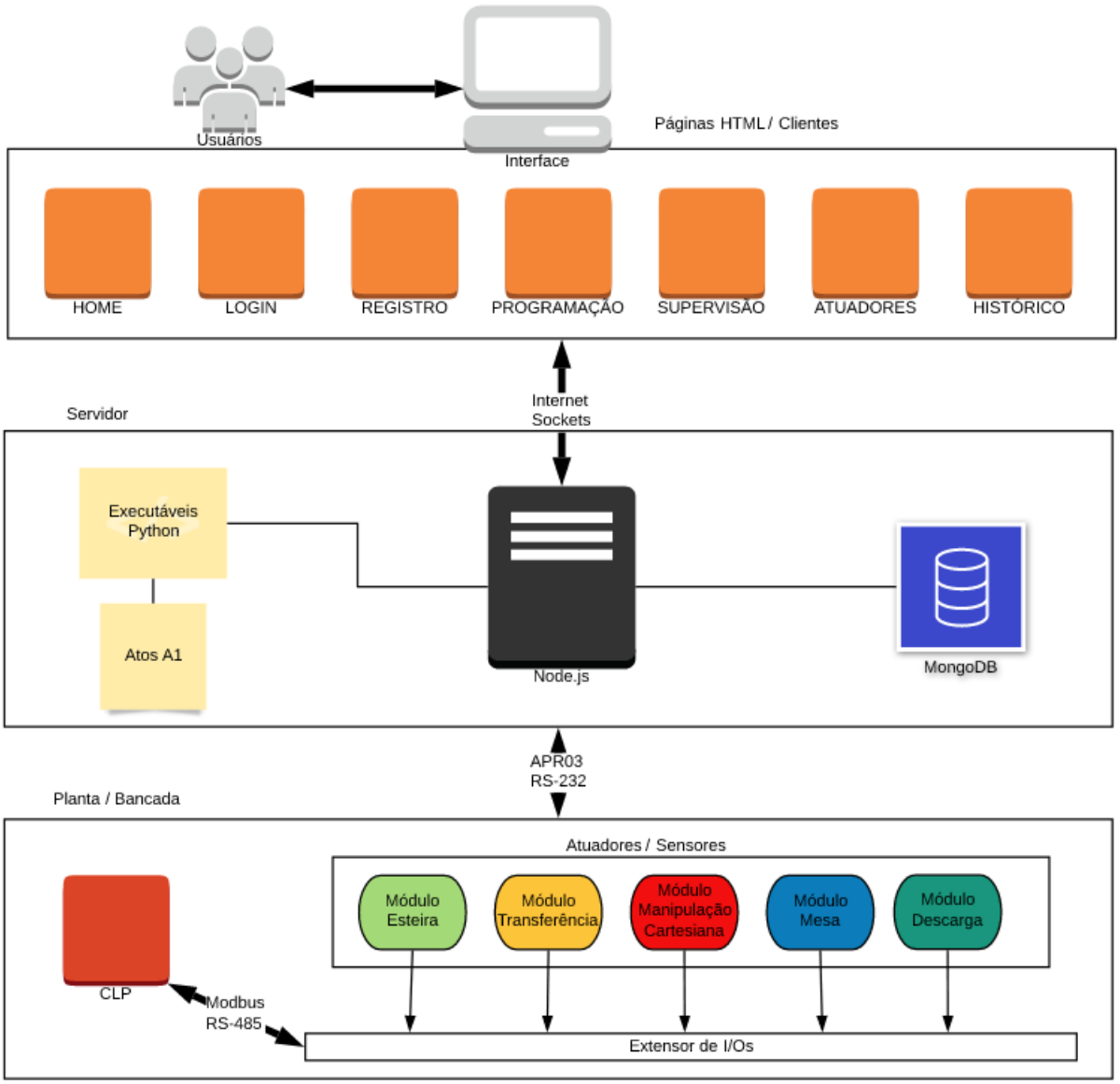
Figura 4.12 – Página de Histórico de Envios.

Fonte: Autor

4.3 FUNCIONAMENTO DO LABORATÓRIO REMOTO

A arquitetura de todo o laboratório remoto proposto está ilustrada na Figura 4.13. O cliente é aberto pelos usuários diretamente nos navegadores de rede através de páginas HTML, realizando operações em linguagem JavaScript. As páginas possuem maneira de navegar entre si e a comunicação personalizada entre servidor e páginas ocorre através de *sockets* do módulo “Sockets.io”.

Figura 4.13 – Comunicação do laboratório remoto.



Fonte: Autor

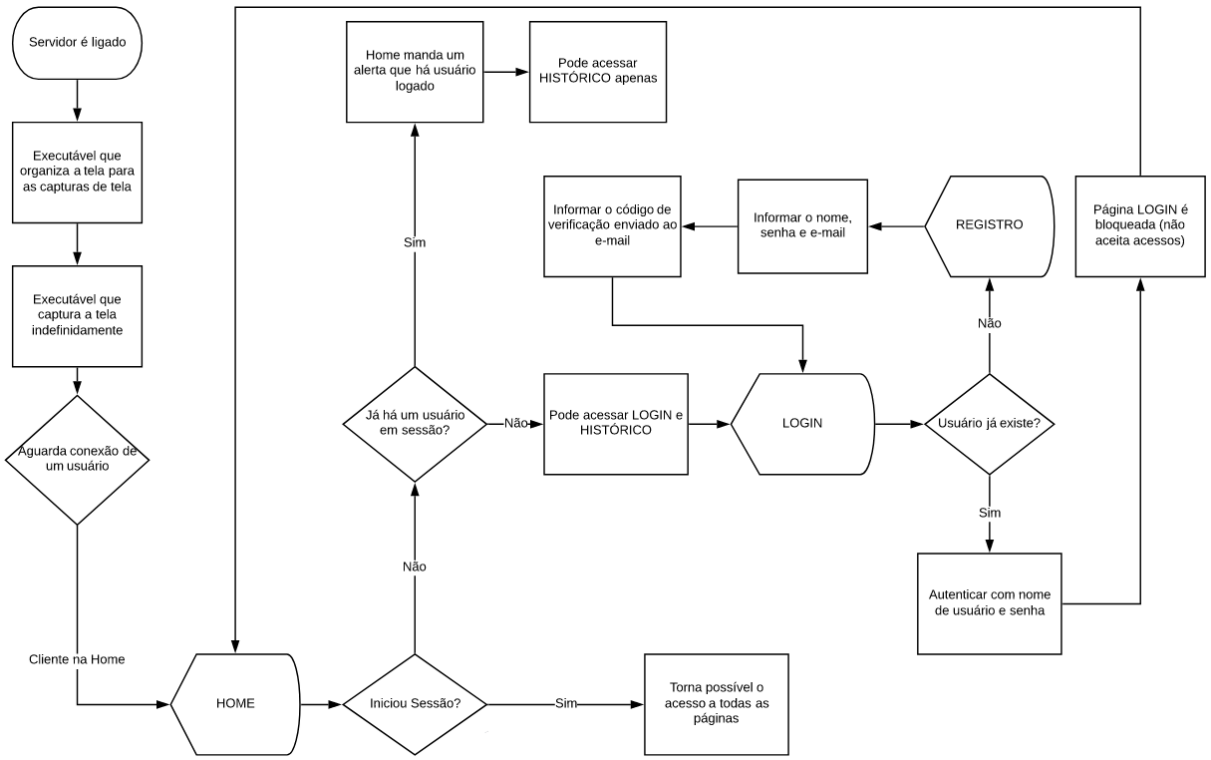
Foi definido que o servidor estaria localizado na porta TCP 100 da máquina o qual se encontra, e como ele estará localizado no laboratório do NUPEDEE, ele usará a rede local através de Ethernet. A rede local do NUPEDEE não pode ser acessada externamente, ou seja, um usuário fora do laboratório (em outra rede local, fora ou dentro da UFSM) não terá acesso ao servidor do laboratório remoto. Para solucionar isto pode-se usar um *proxy*. Assim, o servidor do NUPEDEE foi configurado como o *proxy* do servidor do laboratório remoto. O endereço para acessar o servidor após esta configuração pode ser utilizado na forma nominal, configurada pela rede local, sendo esta “nupedee.ufsm.br:65100”. A porta 65100 é relativa ao *proxy*. Foi feita uma operação manual no protocolo de rede DHCP para que seja um IP estático. O envio de *e-mails* também teve que ser alterado o protocolo, uma vez que a rede fechada do

NUPEDEE impede a utilização da porta do protocolo SMTP, sendo necessário o uso do IMAP. Assim como o servidor, a *webcam* também precisou ser registrada, uma vez que utiliza uma porta diferente no mesmo endereço IP.

De acordo também com a Figura 4.4, o servidor do laboratório remoto utiliza o módulo “http”, o que indica que o protocolo de rede responsável por responder requisições de clientes é o de mesmo nome. Apesar de ser mais seguro, o protocolo “HTTPS” não foi empregado devido a dificuldade de implementação, uma vez que certificados de segurança são necessários e o *proxy* não pôde fornecer suporte.

Como pode ser visto na Figura 4.13, os usuários entram nas páginas pelo navegador de Internet, se comunicando com o servidor aberto em Node.js através de *sockets*. O servidor possui o banco de dados MongoDB e está sempre com o Atos A1 aberto na máquina, para estabelecer a comunicação com a bancada. O Atos A1 é utilizado através dos controles de *mouse* e teclado advindos de bibliotecas Python. O MongoDB é utilizado apenas para o controle de acesso do servidor, que permite apenas um usuário a cada momento. O diagrama mostrado na Figura 4.14 informa como o processo de inicialização do servidor e de que forma é realizada a autenticação de usuários na aplicação.

Figura 4.14 – Controle de acesso do laboratório remoto.

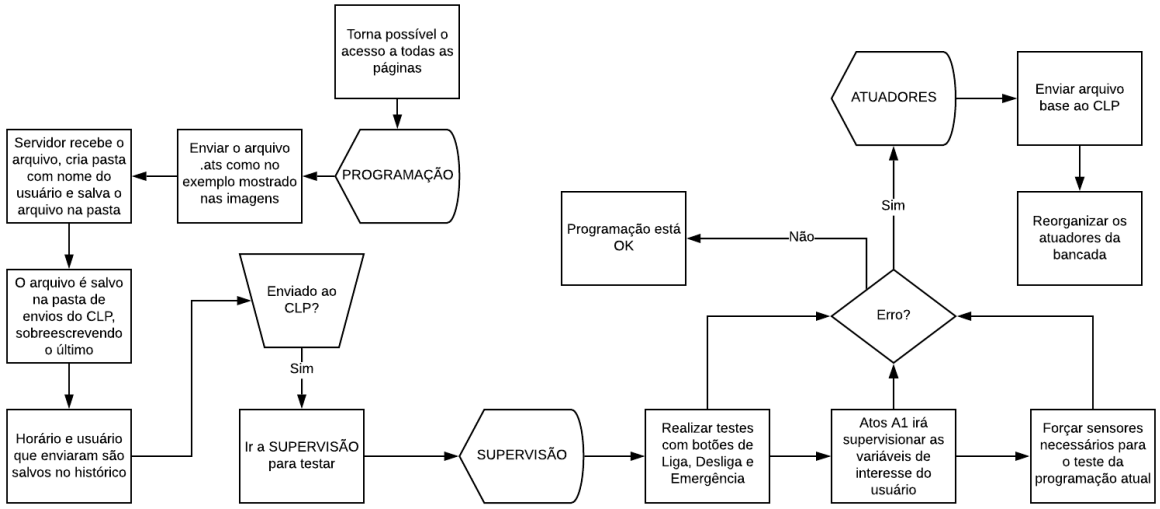


Fonte: Autor

Quanto ao restante das funcionalidades referentes ao laboratório remoto, a Figura 4.15 mostra como é esperado o fluxo de um usuário que possui a finalidade de

testar seu projeto.

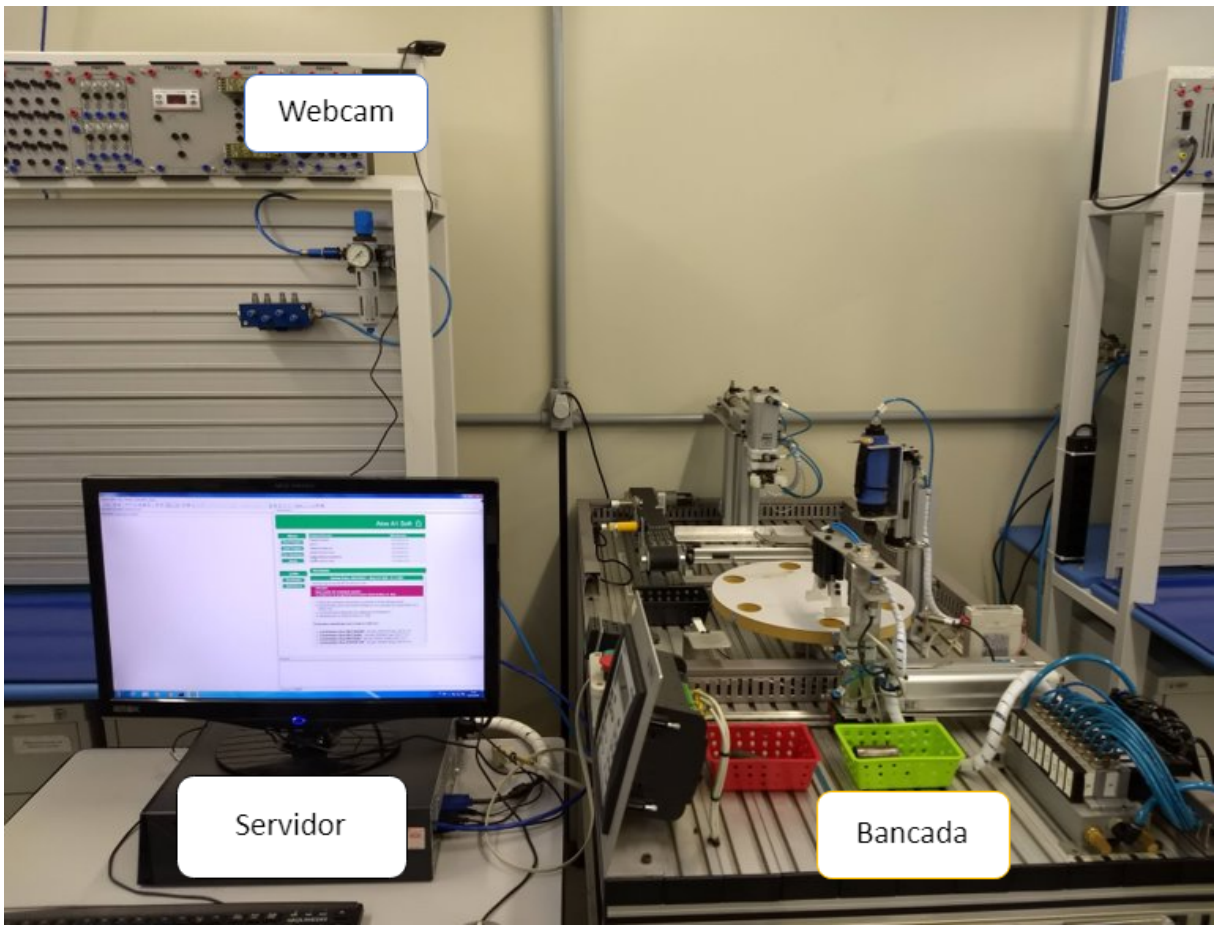
Figura 4.15 – Funcionalidades do laboratório remoto.



Fonte: Autor

A Figura 4.16 mostra o laboratório remoto desenvolvido e implementado com um computador fornecido pelo NUPEDEE. Na Figura estão marcados a máquina do servidor (com este em execução), a bancada e a *webcam* acima do servidor.

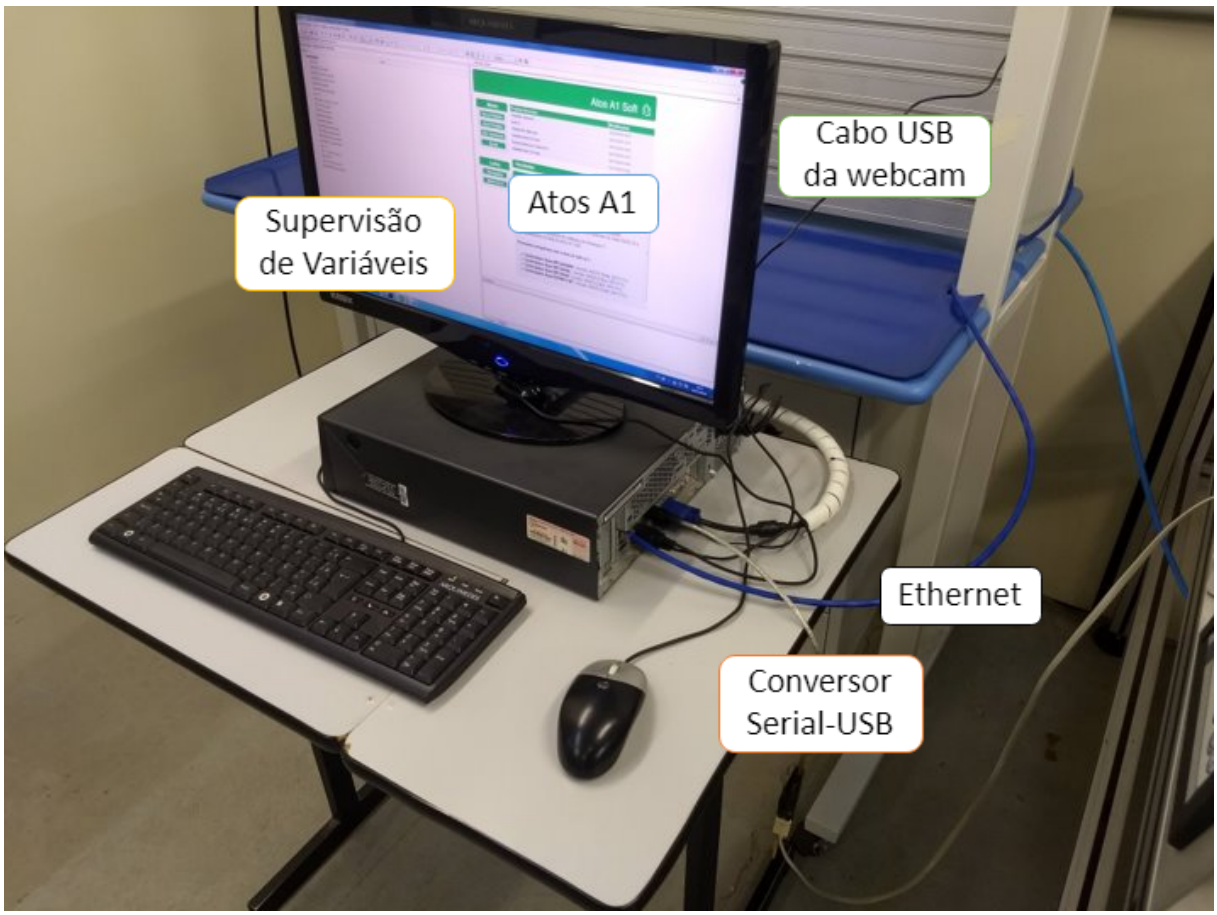
Figura 4.16 – Servidor, bancada e *webcam*.



Fonte: Autor

Quando em execução, o monitor do computador apresenta a interface do Atos A1 com a supervisão de variáveis de interesse aberta (para as capturas de tela ocorrerem de maneira adequada). A Figura 4.17 mostra a tela da máquina do servidor quando este está em execução e os cabos de comunicação com a bancada, *webcam* e Ethernet.

Figura 4.17 – Interface do Atos A1 e cabos de conexão.



Fonte: Autor

5 CONSIDERAÇÕES FINAIS

Estabelecer comunicação a distância entre equipamentos é algo de grande tendência para qualquer área no mundo globalizado. Ser possível acessar e monitorar processos a qualquer hora, em qualquer lugar torna a flexibilidade de tais sistemas algo expressivamente vantajoso. Desta forma, é natural que estas vantagens devem ser transmitidas para o meio acadêmico e assim, permitir que estudantes acessem os laboratórios de pesquisa até mesmo quando não estão presentes.

Um fator que facilitou fortemente a vinda deste tipo de ideia foram os *sockets*, já que proporcionam a comunicação *Web* com servidores e estes, com as portas seriais, dando um certo "roteamento" a algo que antes não era possível. De certa forma, empresas também cada vez mais apostam em tal área, através da abertura de suas propriedades para que mais pessoas entendam como funciona e assim, profissionais mais capacitados entrem no mercado, como por exemplo o protocolo ATOS APR03, que se encontra aberto ao público através da página da Schneider Eletric.

Um grande exemplo de como a abertura de propriedade pode facilitar o avanço desta tecnologia vem sendo apresentado através do desenvolvimento das ferramentas voltadas para estas aplicações. O Node.js por exemplo possui seus códigos abertos a desenvolvedores, para o desenvolvimento de mais módulos, assim como o Python faz com suas bibliotecas. Isto faz com que a flexibilidade, complexidade e variedade de funcionalidades das ferramentas aumente significativamente.

O aprendizado de tais tecnologias acaba se tornando um fator imensurável para o futuro de acadêmicos de diversos cursos. E uma maneira de abrir o interesse e mostrar o que pode ser feito é criar aplicações que envolvam seu âmbito de trabalho: o meio acadêmico. Por isso o desenvolvimento de laboratórios remotos deve ser encorajado e difundido para todas as áreas se possível.

O presente projeto tratou de mostrar como a comunicação pode ser feita de maneira simples utilizando linguagens e métodos adequados. Muitas linguagens descritas neste trabalho possuem vastas maneiras de aprendizado pela Internet e os *softwares* utilizados são todos gratuitos. A abordagem, apesar de simples, mostrou-se plenamente funcional e sugestiva, alcançando todos os objetivos estabelecidos.

O ideal, porém, seria não necessitar controlar o servidor para que este use o Atos A1 e envie os arquivos; seria a reconstrução dos pacotes em JavaScript e o envio destes pelo módulo Node "Serialport", uma vez que chamadas de arquivos executáveis costumam ter um atraso para iniciarem. Este atraso por si só faz o sistema se comportar de maneira imperfeita, sendo necessários por vezes 3 segundos ociosos para então a ação ser executada. Uma maneira de contornar que o manuseio dos usuários causasse erros na aplicação foi justamente desabilitar os botões uma vez

pressionados, impedindo o servidor de executar a mesma ação em um mesmo período de tempo mais de uma vez. Este atraso também foi o motivo de deixar o protocolo APR03 como uma forma de controlar os botões normais da aplicação (Liga, Desliga e Emergência) e não apenas forçá-los com arquivos executáveis; também o porquê de usar o mesmo protocolo para a situação de emergência da página de Atuadores.

Outro fator a ser melhorado é o manuseio de sessões no servidor. A ideia de impedir que mais de um usuário entre nas aplicação bloqueando a página de Login não é efetiva o suficiente, podendo ainda causar erros caso duas pessoas entrem em tal página ao mesmo tempo. Um sistema de "filas" poderia ser desenvolvido para resolver esta adversidade. Mecanismos de confirmação de *e-mail* e senha, bem como a função de "recuperação de senha" também deveriam ser planejados para haver um acesso ideal à aplicação. Uma solução simples, contudo, seria vincular o acesso ao laboratório remoto com as credenciais da plataforma do aluno da própria universidade, retirando a necessidade de um banco de dados na aplicação e de um sistema de controle de acesso muito robusto.

Entretanto, mesmo lembrando de tais problema, os benefícios da presente aplicação devem ser lembrados em um escopo mais amplo. Tendo em vista o desenvolvimento cada vez maior de ferramentas de auxílio para o desenvolvimento de aplicações como esta e o incentivo de empresários para tal área, a adoção de experimentos a distância possui uma grande tendência a crescer no futuro. Instrumentos como novos *softwares*, bibliotecas de linguagens de programação novas e mais funcionais e responsivas além de códigos abertos para entendimento do público, ajudam na propagação do conceito de IoT, sinalizando um mundo ainda mais conexo.

REFERÊNCIAS BIBLIOGRÁFICAS

BELLMUNT, O. et al. A distance plc programming course employing a remote laboratory based on a flexible manufacturing cell. **IEEE Transactions on Education**, v. 49, p. 278–284, 2006.

BOOTSTRAP. **Bootstrap 4 Documentation**. [S.l.], 2019. Disponível em: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>.

BSQUARE. **New research shows growing interest in industrial IoT adoption, companies in early stages**. 2017. Acesso em 21/03/2019. Disponível em: <https://www.worldoil.com>.

CONVERSOR USB x SERIAL RS232. Mercadão da Informática, 2019. Disponível em: <http://www.mercadaodainformatica.com.br/conversor-usb-x-serial-rs232-e-paralelo>.

FARIAS, G.; MEDEIROS, E. O modelo osi da iso. In: ____. **Introdução a Computação**. [s.n.], 2013. cap. 7.4.1, p. 99–102. Disponível em: <http://producao.virtual.ufpb.br/books/camyle/introducao-a-computacao-livro/livro/livro.pdf>.

JS FOUNDATION DEVELOPERS. **Nodejs Documentation**. [S.l.], 2019. Disponível em: <https://nodejs.org/en/docs/>.

LOEFFLERO, J. **How WiFi 6 is About to Revolutionize the Internet of Things**. Interesting Engineering, 2019. Disponível em: <https://interestingengineering.com/how-wifi-6-is-about-to-revolutionize-the-internet-of-things>.

LONGEN, A. **Protocolos de Rede: O Que São, Como Funcionam e Tipos de Protocolos de Internet**. Weblink, 2018. Disponível em: <https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/>.

LUNDVALL, M. **Yet Another Webcam Software**. 2016. Disponível em: <https://www.yawcam.com/>.

PINHEIRO, D. **Câmeras IP permitem vigiar a casa pela Internet; veja como funciona**. UOL Tecnologia, 2006. Disponível em: <https://tecnologia.uol.com.br/ultnot/2006/12/13/ult2870u215.jhtm>.

RIO, R. I. **A 1ł Rede de Internet das Coisas do Brasil**. Rede IoT Rio, 2019. Disponível em: <http://www.redeiot.rio/>.

SAYGIN, C.; KAHRAMAN, F. A web-based programmable logic controller laboratory for manufacturing engineering education. **The International Journal of Advanced Manufacturing Technology**, v. 24, p. 590–598, 2004. Disponível em: <https://link.springer.com/article/10.1007/s00170-003-1787-7>.

SCHAF, F. M. **Arquitetura Modular para Ambientes Virtuais de Ensino de Automação com Suporte à Realidade Mista e Colaboração**. 2011. 162 p. Tese (Doutorado em Engenharia Elétrica) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2011.

SCHNEIDER ELETRIC. **Atos APR03**: Manual de comunicação. [S.l.], 2010. 19 p. Disponível em: <https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=ma01300-0510_apr03.pdf&p_Doc_Ref=ma01300.0510+APR03>.

____. **Atos Expert BF**: Manual do hardware. [S.l.], 2010. 39 p. Disponível em: <http://www.runtal.com.br/pdf/manual_Atos_Expert_BF.pdf>.

SILVEIRA, C. B. **Como Funciona a Linguagem LADDER**. CitiSystems, 2016. Disponível em: <https://www.citisystems.com.br/linguagem-ladder/>.

SIMON, R. **As perspectivas da IoT no Brasil e as vantagens de conhecê-la a fundo**. 2018. Acesso em 20/03/2019. Disponível em: <http://www.administradores.com.br/noticias/negocios/as-perspectivas-da-iot-no-brasil-e-as-vantagens-de-conhece-la-a-fundo/123515/>.

UMBLER. **Boas Práticas com MongoDB**. Umbler, 2017. Disponível em: <https://blog.umbler.com/br/boas-praticas-com-mongodb/>.

ZANCAN, M. D. **Controladores Programáveis**. CTISM, 2011. Disponível em: <https://www.ufsm.br/unidades-universitarias/ctism/cte/wp-content/uploads/sites/413/2018/11/17_controladores_programaveis.pdf>.