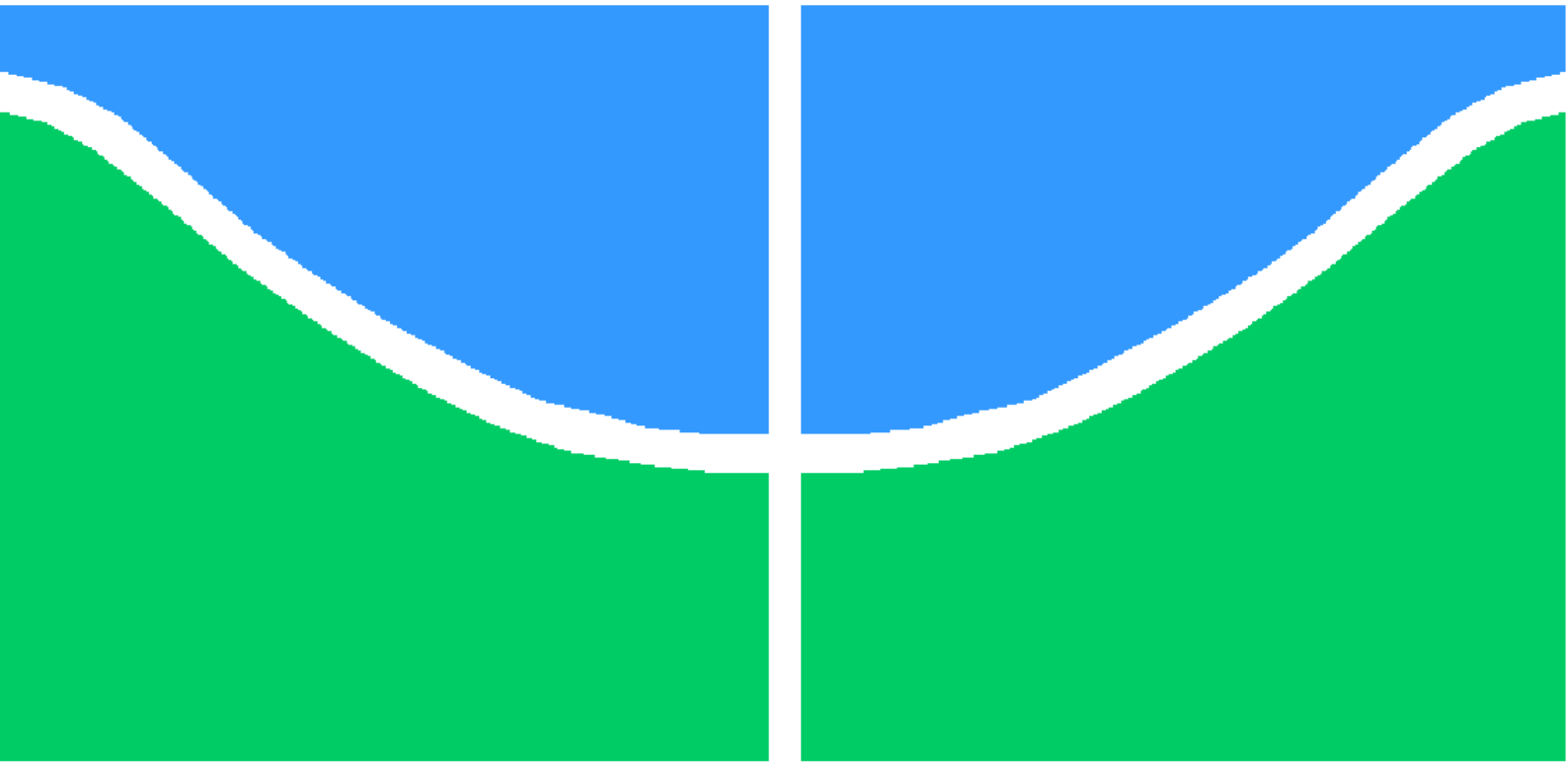

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia eletrônica

Desenvolvimento de Sistema Supervisório para Eclusas

Autor: Rafael Gomes da Silva

Orientador: Professor Doutor Sandro Augusto Pavlik Haddad

Brasília, DF

2024

Rafael Gomes da Silva

Desenvolvimento de Sistema Supervisório para Eclusas

Monografia submetida ao curso de graduação em (Engenharia eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia eletrônica).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Sandro Augusto Pavlik Haddad

Coorientador: Mestre Gilvandson Costa Cavalcante

Brasília, DF

2024

Rafael Gomes da Silva
Desenvolvimento de Sistema Supervisório para Eclusas/ Rafael Gomes da Silva. – Brasília, DF, 2024-
94 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Sandro Augusto Pavlik Haddad

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2024.

1. Eclusa. 2. Supervisório. I. Professor Doutor Sandro Augusto Pavlik Haddad. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Desenvolvimento de Sistema Supervisório para Eclusas

CDU 02:141:005.6

Rafael Gomes da Silva

Desenvolvimento de Sistema Supervisório para Eclusas

Monografia submetida ao curso de graduação em (Engenharia eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia eletrônica).

Trabalho aprovado. Brasília, DF, 20 de Setembro de 2024:

Professor Doutor Sandro Augusto Pavlik Haddad
Orientador

Mestre Gilvandson Costa Cavalcante
Convidado 1

Professor Doutor Alex Reis
Convidado 2

Brasília, DF
2024

Agradecimentos

Ao Grande Arquiteto do Universo pelos favores a ele já recebidos, ao meu falecido pai, Waldemar Alves da Silva Filho, que sempre acreditou em meu futuro, aos meus amigos André Pereira e Helton Azevedo que sempre me ajudaram. A minha amada esposa, Isabella Savi, por sempre me amar e acreditar em mim. Aos Servidores do DNIT André Martinez e Gilvandson Costa Cavalcante que encabeçaram o projeto e me deram a oportunidade de usar todo meu conhecimento em Engenharia para a criação de um produto que atenda as demandas nacionais.

Resumo

Eclusas são sistemas que trabalham como elevadores para a movimentação de embarcações em ambos os sentidos de um rio. Para seu melhor controle, um sistema supervisório é imprescindível para o monitoramento de toda a planta. Neste contexto, o uso de um protocolo industrial como o MODBUS, o uso do controlador ESP32 e de um supervisório de código aberto SCADABR são bastantes proveitosos, pois além de serem amplamente usados em industrias, possuem uma característica de fácil manipulação e controle por parte dos operadores e engenheiros. Neste trabalho será abordado a implementação de um sistema supervisório em uma eclusa, e toda a eletrônica por trás da implementação, bem como um protótipo para teste de paradigma. Na primeira parte, serão tratados conceitos a cerca do que é uma eclusa, e tecnologias que a compõem. Logo após serão discutidos tecnologias a serem implementadas, resultados e recomendações para os engenheiros e operadores das eclusas.

Palavras-chaves: Eclusas. *Modbus*. *SCADA*. ESP32.

Abstract

Locks are systems that work as elevators for moving vessels in both directions on a river. For better control, a supervisory system is essential for monitoring the entire plant. In this context, the use of an industrial protocol such as MODBUS, the use of the ESP32 controller and an open source supervisory system SCADABR are quite beneficial, since in addition to being widely used in industries, they are easy to manipulate and control by operators and engineers. This work will address the implementation of a supervisory system in a lock, and all the electronics behind the implementation, as well as a prototype for paradigm testing. The first part will address concepts about what a lock is and the technologies that comprise it. Soon after, technologies to be implemented, results and recommendations for engineers and lock operators will be discussed.

Key-words: *Locks. Modbus. SCADA.* ESP32

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AC	Corrente Alternada
ADC	Conversor Analógico-Digital
CFTV	Circuito Fechado de Televisão
CLP	Controlador Lógico Programável
DC	Corrente Contínua
DNIT	Departamento Nacional de Infraestrutura de Transportes
EEPROM	memória de somente leitura programável apagável
ENOB	Relação sinal-ruído e distorção
GCC	*GNU C Compiler*
GPIO	*General Purpose Input Output*
HVAC	*Heating, Ventilation, and Air Conditioning*
IGBT	*Insulated Gate Bipolar Transistors*
IoT	*Internet of Things*
MBAP	*ModBus Application Protocol Header*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
OSI	Interconexão de Sistemas Abertos
PDU	Unidade de Dados do Protocolo
RTU	*Remote Terminal Unit*
SCADA	*Supervisory Control and Data Acquisition*
SFRD	intervalo dinâmico livre de componentes espúrias
SNR	Relação Sinal-Ruído
SRAM	memória de acesso aleatório dinâmica
TCP/IP	*Transmission Control Protocol/Internet Protocol*
THD	distorção harmônica total
VFD	*Variable Frequency Drivers*

Lista de símbolos

A	Corrente em Ampère
C	Capacitância
dB	Decibel
e_q	Erro de quantização
H	Altura
KB	Kilobyte
L	Quantidade de líquido armazenado
N	Número de Bits
Ω	Resistência em Ohms
Q	Vazão
R	Resistência
s	Segundo
V	Tensão em Volts
V_{GS}	Tensão de *Gate-Source*
$V_{GS(Th)}$	Tensão de *Gate-Source* de *threshold*
V_{ref}	Tensão de Referência
ΔV	Resolução do Conversor A/D

Sumário

Parte I

Considerações Iniciais

1 Introdução

As eclusas desempenham um papel fundamental na infraestrutura de transporte aquaviário, facilitando a navegação de embarcações em rios, canais e vias navegáveis interiores. Desde os tempos antigos, as eclusas têm sido utilizadas para superar diferenças de nível em corpos d'água, permitindo que embarcações naveguem em trechos de água com diferentes altitudes. Ao longo dos séculos, as eclusas evoluíram de estruturas rudimentares para sofisticados sistemas de engenharia que desempenham um papel vital no comércio e no transporte de mercadorias em todo o mundo.

O funcionamento das eclusas é baseado no princípio da elevação ou descida controlada das embarcações entre diferentes níveis de água. As eclusas consistem em câmaras de água fechadas por comportas, onde as embarcações entram e são então elevadas ou baixadas conforme necessidade de operação em determinados horários. Esse processo é controlado por meio de sistemas hidráulicos e eletromecânicos, permitindo que as embarcações naveguem em trechos de água com com alturas diferentes tendo como referência o nível do mar.

Sob a responsabilidade do Departamento e Infraestrutura de Transportes (DNIT) existem oito eclusas. Quatro delas ficam no Rio grande o sul (Amarópolis, Bom Retiro do Sul, Dom Marco e Fandango) duas em São Paulo (Jupiá e Três Irmãos), uma no Pará (Tucuruí) e uma na Bahia (Sobradinho).

Este trabalho tem como objetivo da criação de um sistema de controle do supervisório da eclusa de sobradinho sobradinho, usando equipamentos de hardware aberto e com conceito de Internet das Coisas (IoT). Para a criação do software do supervisório será com ferramentas disponíveis no DNIT, com o objetivo de visualização em tempo real do status da eclusa.

1.1 Objetivos Gerais

Este trabalho tem como objetivo investigar os princípios de engenharia eletrônica aplicados na automação de eclusas, analisando os sistemas de controle, sensores, atuadores e protocolos de comunicação utilizados.

1.2 Objetivos Específicos

1. Investigar as tecnologias de automação empregadas em eclusas, com foco na engenharia eletrônica e utilizando soluções de hardware aberto, a fim de:

- Explorar as possibilidades de aplicação de soluções de hardware aberto, como Arduino, *Raspberry Pi*, ESP32 ou outras plataformas similares, para controle e monitoramento das operações das eclusas.
- Desenvolver e implementar um sistema de automação de eclusas utilizando soluções de hardware aberto, incluindo a seleção e integração de sensores, atuadores e outros componentes eletrônicos necessários.
- Avaliar a viabilidade técnica e operacional da utilização de soluções de hardware aberto em comparação com sistemas tradicionais de automação baseados em CLPs, considerando critérios como desempenho, custo, flexibilidade e escalabilidade.

2. Investigar os desafios específicos relacionados à implementação de soluções de hardware aberto em ambientes críticos, como eclusas, incluindo aspectos de confiabilidade, segurança cibernética e interoperabilidade com sistemas legados.

3. Avaliar os benefícios e limitações das soluções de hardware aberto em termos de impacto econômico, destacando oportunidades de redução de custos, aumento da acessibilidade tecnológica e potencial para inovação.

4. Criar um protótipo da usabilidade de um supervisório, mostrando seu funcionamento como paradigma.

1.2.1 Justificativa

Com a necessidade de desenvolver um modelo de controle que seja sustentável para o DNIT, se criou a necessidade de um sistema que fosse feito *in house* com robustês e economia. Para com isso, criar um modelo que seja replicado para outras eclusas no Brasil.

1.3 Estrutura do trabalho

A partir dos objetivos gerais e específicos, delimitamos o escopo do trabalho para que possamos mostrar como será para o TCC2 a estrutura de trabalho. O trabalho é divido em 4 partes: I) Considerações Iniciais II) Referncial teórico III) Decisões de projeto e trabalho e IV) Conclusões e trabalhos futuros.

No capítulo 1 será dada uma introdução e os objetivos do trabalho;

No capítulo 2 será abordado o referencial teórico do trabalho;

No capítulo 3 Será mostrado decisões de engenharia acerca do projeto proposto e as escolhas dos componentes para controle e software de interface;

No capítulo 4 Será abordado a estrutura de prototipagem, e das ferramentas de hardware aberto;

No capítulo 5 Será a implementação do projeto no software de supervisório e os itens para implementação do projeto na eclusa para os operadores;

No capítulo 6 Será dada as conclusões sobre o trabalho e os trabalhos futuros.

Parte II

Referencial Teórico

2 As eclusas

Neste capítulo serão abordados conceitos a cerca das eclusas, seu funcionamento e os problemas enfrentados a cerca da eclusa de sobradinho

2.1 O sistema de Eclusagem

2.1.1 Introdução ao sistema de eclusagem

As eclusas são uma obras da engenharia hidráulica que permitem o transporte de embarcações por canais com diferenças de altitude (MCCARTNEY, 1998). Elas funcionam como elevadores de água, permitindo que os navios subam e desçam através de um sistema de comportas.

O funcionamento de uma eclusa se baseia na gravidade. Quando um navio vai fazer o trajeto de descida por uma eclusa, a primeira porta da eclusa se abre permitindo que o navio entre. Assim que o navio entra, a porta é fechada novamente e a água é retirada até que atinja o mesmo nível do corpo d'água a jusante da eclusa. Quando atinge o mesmo nível, a segunda porta se abre e o navio pode sair.

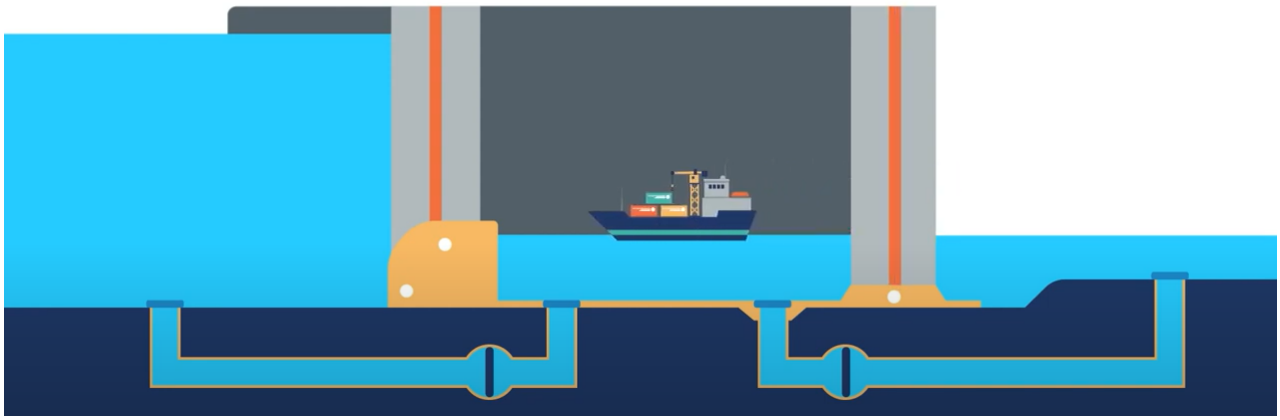

Figura 1 – Imagem básica do processo de uma eclusa adaptado de (Acervo DNIT)

Dependendo da diferença de altitude entre o corpo d'água a montante e o corpo d'água a jusante da eclusa, ela pode ser classificada em: eclusa de baixa queda, eclusa de média queda, eclusa de alta queda e eclusa de altíssima queda. O tamanho da queda influencia em fatores importantes como a maior ou menor propensão à formação de turbulência, o tempo de enchimento e esvaziamento, variação maior ou menor no pico das vazões de enchimento/esvaziamento, problemas de cavitação, velocidade de condução nos

tubos de enchimento/esvaziamento e a necessidade de sistemas mais eficientes dissipadores de energia.

As eclusas possuem um potencial econômico muito importante, pois elas permitem a navegação em rios e canais, facilitando o transporte de mercadorias e reduzindo os custos logísticos além de ser um impulso em algumas regiões do Brasil, como a região norte-nordeste, facilitando o transporte de cargas e contribuem para o crescimento econômico dessas áreas.

2.1.2 A eclusa de Sobradinho

A Eclusa de Sobradinho foi construída em 1979 e está situada no rio São Francisco. Ela é composta por uma câmara de eclusagem de 120 metros de comprimento, 17 metros de largura e 2,5 metros de calado. Essas dimensões permitem que as embarcações superem um desnível de 32,5 metros, o equivalente a um prédio de dez andares, possibilitando assim a transposição da barragem da Usina Hidrelétrica(SANTOS et al., 2023).

As operações na Eclusa de Sobradinho foram retomadas pelo Governo Federal, por meio do Departamento Nacional de Infraestrutura de Transportes (DNIT), em 26 de março de 2021, após terem sido interrompidas em 20181. A retomada das operações faz parte das ações executadas pelo Programa Nacional de Recuperação, Operação, Manutenção e Gestão de Eclusas (PROECLUSAS), liderado pelo DNIT.

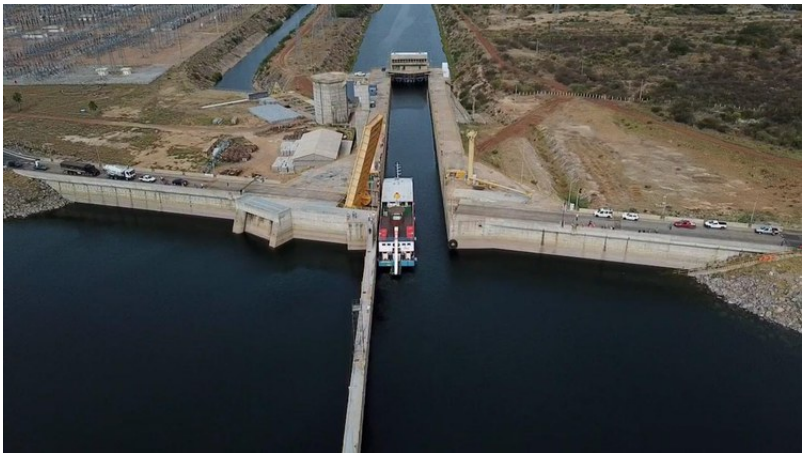

Figura 2 – Elcusa de Sobradinho, Vista superior adaptado de (Acervo DNIT)

2.1.3 Uma Breve descrição do funcionamento das Eclusas

Uma eclusa possui no centro as câmaras de água, que servem como grandes reservatórios fechados por comportas em ambos os lados. Essas câmaras são projetadas para acomodar as embarcações durante o processo de transposição de níveis de água. As comportas são dispositivos de controle de fluxo de água que regulam a entrada e saída de

Figura 3 – Elcusa de Sobradinho, Vista frontal adaptado de (Acervo DNIT)

embarcações das câmaras de água. Elas também são responsáveis por manter o nível de água adequado dentro da eclusa durante a eclusagem.

Os sistemas de controle desempenham um papel fundamental no funcionamento da eclusa, monitorando e coordenando todas as operações relacionadas à sua operação. Isso inclui o acionamento das comportas, o controle dos sistemas de enchimento e esvaziamento das câmaras de água e a comunicação com as embarcações durante o processo de eclusagem.

Além disso, as eclusas são equipadas com dispositivos de ancoragem e amortecimento para garantir a segurança das embarcações durante a eclusagem. Esses dispositivos ajudam a evitar colisões e danos às embarcações e à própria estrutura da eclusa.

O processo de eclusagem é complexo e envolve três fases: A primeira fase (Fig.2.1.3) envolve a entrada das embarcações na câmara de água da eclusa e as comportas são fechadas atrás delas para garantir a estanqueidade da câmara. Na segunda fase(Fig.2.1.3), o nível de água na câmara de água é ajustado para igualar o nível da água com o trecho de água a ser alcançado. Já na terceira fase, uma vez que o nível de água é ajustado, as comportas na extremidade oposta da eclusa são abertas e a embarcação pode sair para o trecho de água de nível diferente (SANTOS et al., 2023).

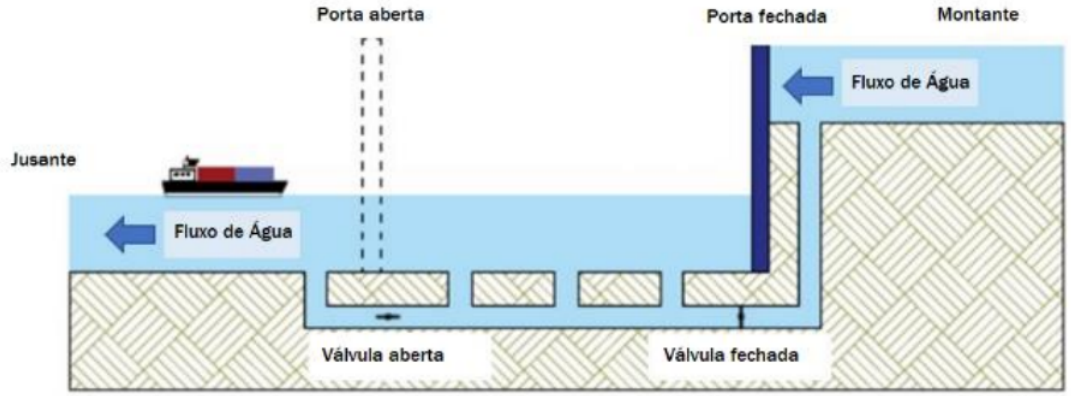

Figura 4 – Primeira Fase: Entrada da embarcação na eclusa adaptado de (Acervo DAQ)

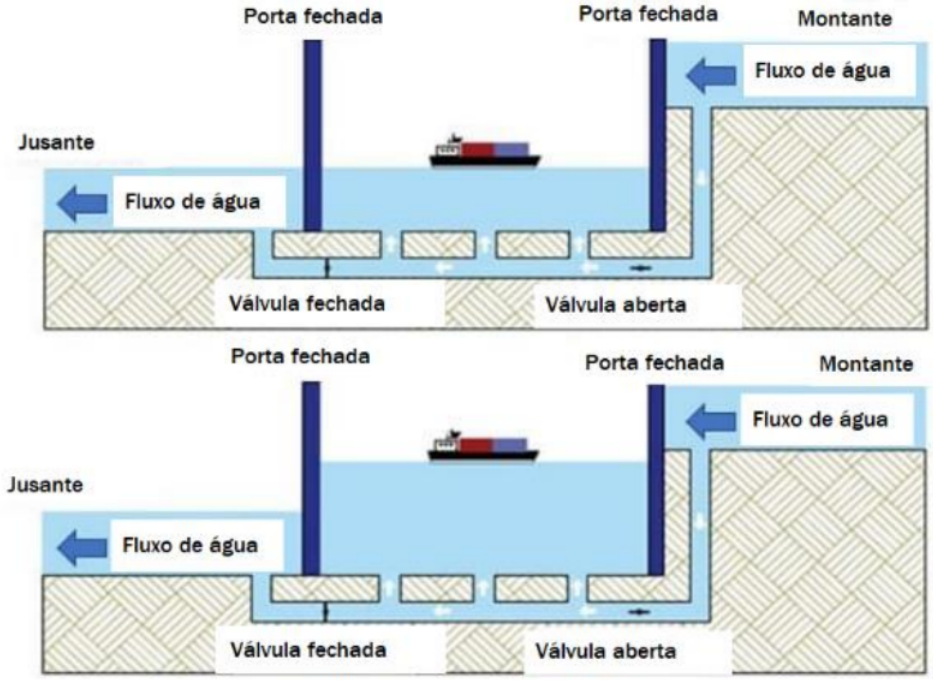

Figura 5 – Segunda Fase: Entrada da embarcação na eclusa adaptado de (Acervo DAQ)

2.2 Automação de uma Eclusa

2.2.1 Uma introdução sobre Automação Industrial

A automação industrial é um campo de estudo em constante evolução que se concentra na aplicação de tecnologias avançadas para otimizar processos industriais. Ela desempenha um papel crucial na melhoria da eficiência, segurança e produtividade nas indústrias modernas(GROOVER, 2018).

A automação industrial é a utilização de sistemas de controle, como computadores ou robôs, e tecnologias de informação para lidar com diferentes processos e máquinas em

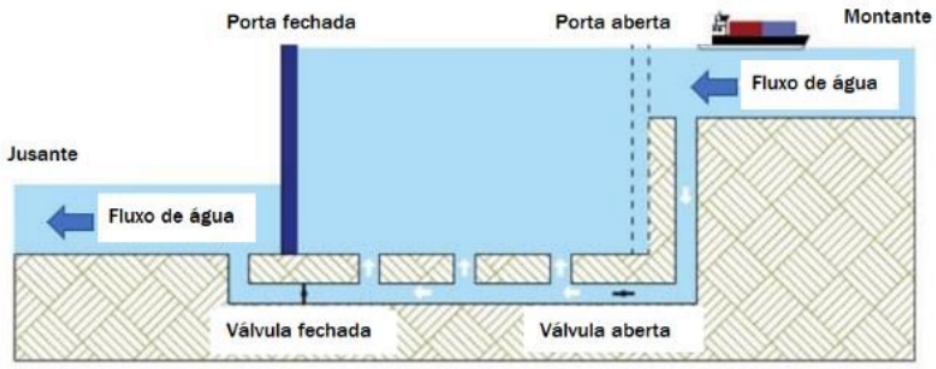

Figura 6 – Primeira Fase: Entrada da embarcação na eclusa adaptado de (Acervo DAQ)

uma indústria para substituir um humano. Ela é vital para muitas indústrias, incluindo manufatura, transporte, serviços públicos, defesa, energia, e saúde.

Suas vantagens consiste em vários benefícios, incluindo aumento da produtividade, melhoria da qualidade, redução de custos, e melhoria da segurança no local de trabalho. Além disso, ela permite que as empresas se mantenham competitivas em um mercado global cada vez mais competitivo.

2.2.2 Componentes de uma eclusa

- Controlador Lógico Programável (CLP): O CLP é um computador compacto de alta robustez com o objetivo de uso em Indústria, que tem a função de executar as linhas dos diagramas Ladder, permanecendo como elemento principal de controle do sistema da eclusa.

- Sistema Supervisório (SCADA): Este é um software usado para controlar processos industriais à distância. Ele permite aos operadores visualizar e interagir com o estado operacional da eclusa.

- Sensores e Atuadores: Os sensores são usados para coletar dados do ambiente, como nível de água, posição das portas, etc. Os atuadores, como motores e válvulas, realizam ações físicas baseadas nas instruções do CLP.

- Redes de Comunicação Industrial: Estas redes permitem a comunicação entre o CLP, o sistema supervisório e os vários sensores e atuadores, na eclusa se usa o protocolo *Modbus*.

- Sistema de CFTV: Um sistema de Circuito Fechado de Televisão (CFTV) é utilizado para acompanhamento de eclusagens.

2.3 Modelagem matemática do sistema de uma eclusa

Consideramos o fluxo de água ao longo da comporta (Figura 2.5), e podemos assumir o sistema como três reservatórios: um reservatório a montante, um reservatório a jusante e um reservatório na câmara de eclusagem. Existe também duas comportas, onde conectam os reservatórios da montante e da jusante a câmara de eclusagem. Em cada comporta (Que para o exemplo será modelada como uma válvula de restrição) temos uma resistência de restrição R dada por:

$$R = \frac{\mathrm{d}H}{\mathrm{d}Q} \tag{2.1}$$

Onde:

H: Variação da altura, em metros (m) Q: Variação da vazão em volume, em metros cúbicos por segundo (m^3/s)

Em todos os reservatórios, temos uma capacitância, que é dada por:

$$C = \frac{\mathrm{d}L}{\mathrm{d}H} \tag{2.2}$$

Onde:

L: Variação da quantidade de líquido armazenado, em metros cúbicos (m^3) H: Variação da altura, em metros (m)

Podemos fazer uma modelagem de acordo com (OGATA, 2010) sendo Q' a vazão em regime permanente, em metros (m^3/s), q_1 e q_0 pequenas variações das vazões de entrada e saída, H' a altura do nível e h um pequeno desvio de nível a partir do seu valor em regime permanente.

Então, se considerarmos apenas a abertura de comportas entre montante e câmara de eclusagem, e jusante e câmara de eclusagem e que temos uma variação da altura e do fluxo ao longo do tempo, temos a seguinte sistema de equação diferencial (OGATA, 2010):

$$RC\frac{dh}{dt} + h = Rq_i \tag{2.3}$$

observando que RC é a constante de tempo do sistema. Fazendo a transformada de Laplace, temos que a razão $H(s)$ e $Q_i(s)$ é:

$$\frac{H(s)}{Q_i(s)} = \frac{R}{RCs + 1} \tag{2.4}$$

Com isso, podemos admitir que a razão entre a entrada e a saída do fluxo podemos retirar a função de transferência sendo:

$$\frac{Q_0(s)}{Q_i(s)} = \frac{1}{RCs+1} \tag{2.5}$$

Apesar de 2.5 ser a modelagem básica do sistema, o sistema Montante - Câmara de Eclusagem - Jusante é um sistema de nível que possui mais de uma interação. Então de acordo com (OGATA, 2010), temos que para um fluxo q_k da câmara de eclusagem, R_k e C_k resistência e capacitância da câmara , $q1$ o fluxo da jusante e $q2$ o fluxo da montante, Então o resultado é a seguinte função de transferência:

$$\frac{Q_1(s)}{Q_k(s)} = \frac{1}{R_kC_kR_1C_1s^2 + (R_1C_1 + R_kC_k + R_1Ck)s + 1} \tag{2.6}$$

e

$$\frac{Q_2(s)}{Q_k(s)} = \frac{1}{R_kC_kR_2C_2s^2 + (R_2C_2 + R_kC_k + R_2Ck)s + 1} \tag{2.7}$$

Com 2.6 e 2.8 podemos retirar a função de transferência entre montante e jusante, dada por

$$\frac{Q_2(s)}{Q_1(s)} = \frac{R_1C_1 + R_1}{R_2C_2 + R_2} = \frac{\tau_1}{\tau_2} + \frac{R_1}{R_2} \tag{2.8}$$

onde τ_1 e τ_2 São as constantes de tempo de jusante e montante

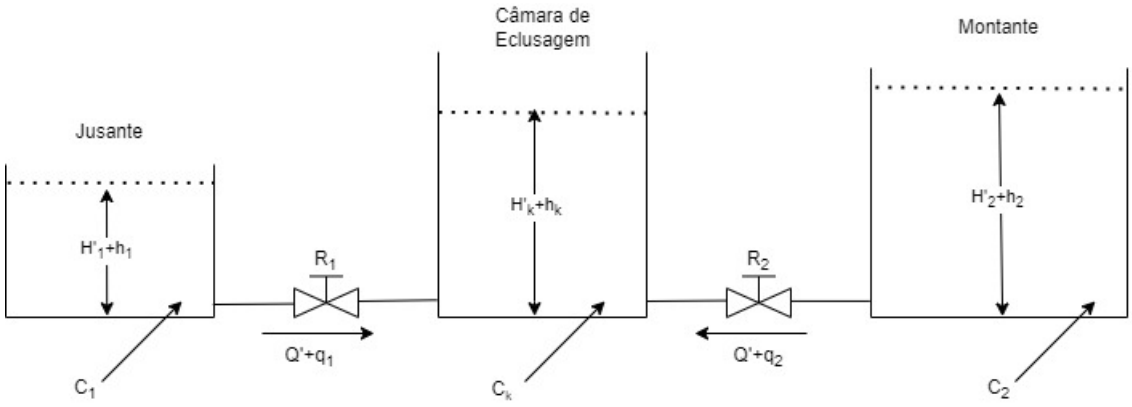

Figura 7 – Modelo básico de modelagem de uma eclusa

2.4 Motores e bombas industriais

No cenário industrial moderno, bombas e motores desempenham um papel crucial em diversas aplicações, desde o fornecimento de água até processos de manufatura complexos. Eles são essenciais para o funcionamento eficiente de sistemas de produção, transporte de fluidos e manuseio de materiais.

No que abrange os motores temos os Motores Trifásicos que são os mais comuns em aplicações industriais devido à sua robustez, eficiência e baixo custo de manutenção. O seu funcionamento é baseado baseados no princípio de indução eletromagnética, onde a corrente elétrica induzida no rotor é responsável por criar torque, Motores Monofásicos que são Utilizados em aplicações de menor potência onde a rede elétrica disponível é monofásica. Embora menos eficientes que os trifásicos, são adequados para pequenos equipamentos e ferramentas.

De acordo com (CHAPMAN, 2013), os motores de indução trifásicos operam baseados no campo magnético girante criado pela corrente trifásica no estator. Este campo induz uma corrente no rotor, que interage com o campo magnético para produzir torque. A velocidade do motor é determinada pela frequência da corrente alternada e o número de polos do motor.

Figura 8 – bomba usada para adução e escoamento dos poços

Segundo (GROOVER, 2018), A integração de bombas e motores com sistemas de automação e controle é importante para a para otimizar a eficiência energética, reduzir custos operacionais e melhorar a confiabilidade dos processos industriais. Tecnologias como inversores de frequência (VFDs) permitem o controle preciso da velocidade de motores, adaptando-se às necessidades do processo em tempo real. Sensores e CLPs monitoram parâmetros críticos, ajustando automaticamente as operações para maximizar a eficiência e minimizar o desgaste dos equipamentos.

2.5 Inversores de Frequência

(HASHID, 2014) define inversores de frequência, também conhecidos como drives de frequência variável (VFDs), como dispositivos eletrônicos utilizados para controlar a velocidade e o torque de motores elétricos AC. Eles convertem a tensão de entrada fixa e a frequência da rede elétrica em uma saída de tensão e frequência variável, permitindo um controle preciso do motor.

Como princípio de funcionamento, segundo (HASHID, 2014) Os inversores de frequência operam em três estágios principais: Retificação: A corrente alternada da rede elétrica é convertida em corrente contínua através de um retificador. Filtragem: A corrente contínua é suavizada por capacitores e indutores para eliminar ondulações. Inversão: A corrente contínua é convertida de volta em corrente alternada com a frequência e tensão desejadas por meio de um inversor, que utiliza dispositivos de comutação eletrônica como IGBTs (*Insulated Gate Bipolar Transistors*).

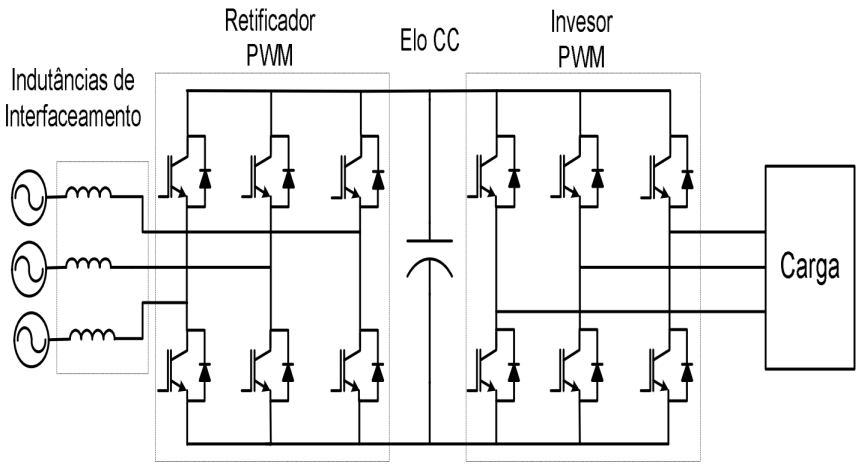

Figura 9 – Funcionamento básico de um inversor. adaptado de (FILHO et al., 2010)

Os benefícios dos Inversores são em suma a Eficiência Energética, Permitindo ajustar a velocidade do motor conforme a demanda do processo, reduzindo o consumo de energia. O Controle Preciso, proporcionando um controle mais preciso da velocidade e do torque do motor, melhorando a qualidade do processo. A proteção do Equipamento, incluindo diversas proteções contra sobrecorrente, sobretensão, sobretemperatura, e inversão de fase, prolongando a vida útil do motor e do sistema e a redução de desgaste, sendo a partida suave e o controle de aceleração/desaceleração reduzindo o desgaste mecânico.

Inversores são amplamente utilizados em diversas indústrias, incluindo controle de velocidade de motores em linhas de produção, HVAC , controle de bombas e ventiladores para eficiência energética, Controle de bombas em estações de tratamento, Controle de guindastes, elevadores e transportadores.

2.6 Protocolo de Comunicação Modbus

De acoro com (ModBus Organization, 2006), o protocolo Modbus TCP é uma variante do protocolo Modbus original, desenvolvido pela Modicon (agora *Schneider Electric*) em 1979 para comunicação entre CLPs. Adaptado para operar sobre redes *Ethernet* utilizando o protocolo TCP/IP, o Modbus TCP tornou-se amplamente utilizado na automação industrial devido à sua simplicidade e eficácia na comunicação entre dispositivos heterogêneos.

O Modbus TCP tem funcionamento em três camadas principais dentro de um modelo Referência para Interconexão de Sistemas Abertos (OSI), a Camada de Aplicação, onde temos a Unidade de Dados do Protocolo (PDU) que Consiste em um cabeçalho e os dados e as Funções Modbus, que são instruções específicas para leitura/escrita de registradores e bobinas, assim como funções de diagnóstico. Em segundo temos a Camada de Transporte, onde o Modbus TCP utiliza o protocolo de transporte TCP, que garante a entrega confiabilidade a entrega de dados e por ultimo a Camada de Rede, onde fornce a via ethernet a interconexão física e a camada de enlace de dados para os dispositivos na rede local.

Segundo a documentação de (ModBus Organization, 2006) Um pacote Modbus TCP ou mesmo RTU possui 8 bytes como identificador, sendo 7bytes de Cabeçalho MBAP (*Modbus Application Protocol Header*), e desses 7 bytes, 2 bytes são de Identificador de Transação, que são Associados a requisição e resposta. 2 bytes de Identificador de Protocolo, sendo sempre 0 para Modbus. 2 bytes de Comprimento, que indica o tamanho dos dados subsequentes e 1 byte de Identificador de Unidade que Identifica o dispositivo de destino.

O ultimo byte é dedicado ao PDU (*Protocol Data Unit*), que possui vários tipos de Código de Função, que identifica e Indica a função a ser executada (por exemplo, leitura, escrita). Os principais dados enviados pelo protocolo são:

- 0x01 (Leitura de Bobinas): Leitura do estado das bobinas (saídas digitais).
- 0x02 (Leitura de Entradas Discretas): Leitura do estado das entradas discretas.
- 0x03 (Leitura de Registradores de Retenção): Leitura dos registradores de armazenamento.
- 0x04 (Leitura de Registradores de Entrada): Leitura dos registradores de entrada.
- 0x05 (Escrita em Única Bobina): Escrita no estado de uma bobina.
- 0x06 (Escrita em Único Registrador): Escrita em um único registrador.
- 0x0F (Escrita em Múltiplas Bobinas): Escrita no estado de múltiplas bobinas.

- 0x10 (Escrita em Múltiplos Registradores): Escrita em múltiplos registradores.

O Modbus TCP oferece diversas vantagens, dentre as quais a facilidade de comunicação entre diversos componentes e fabricantes, não causar problemas nas diversas arquiteturas de rede, a maior facilidade entre interfaces humano-maquina e sistemas de supervisório.

2.7 Conversores Analógico/Digital

Os conversores Analógico-Digital (A/D) são componentes essenciais em sistemas de aquisição de dados, permitindo a conversão de sinais analógicos contínuos em dados digitais discretos que podem ser processados por microcontroladores e sistemas de processamento digital.

A quantização é o processo de mapeamento de um sinal analógico contínuo para valores digitais discretos. Este processo introduz um erro chamado erro de quantização, que é a diferença entre o valor analógico real e o valor quantizado. O sinal analógico é dividido em intervalos de amplitude iguais Cada intervalo é representado por um valor digital específico, sendo um conversor A/D com N bits pode representar 2^N níveis distintos.

A quantização de um valor de entrada V_{in} para um nível digital $Q(V_{in})$ pode ser expressa como (IEEE, 2023):

$$Q(V_{in}) = \Delta V \cdot \left\lfloor \frac{V_{in}}{\Delta V} + 0.5 \right\rfloor \tag{2.9}$$

Onde ΔV é a resolução do conversor A/D.

O erro de quantização é dado pela diferença entre o valor analógico real V_{in} e o valor quantizado $Q(V_{in})$ (IEEE, 2023)::

$$e_q = V_{in} - Q(V_{in}) \tag{2.10}$$

Já a resolução em nível de tensão (ΔV) é dada pela relação entre o intervalo de entrada total (V_{ref}) e o número de níveis distintos(IEEE, 2023)::

$$\Delta V = \frac{V_{ref}}{2^N} \tag{2.11}$$

É importante analisar que em um conversor A/D, além do processo de quantização, de erro e resolução, segundo (IEEE, 2023), os principais parâmetros de avaliação para a caracterização de um conversor A/D são a Relação Sinal- Ruído (SNR), a relação sinal-ruído e distorção (SINAD), o número efetivo de bits (ENOB), a distorção harmônica total (THD) e o intervalo dinâmico livre de componentes espúrias (SFRD).

O SNR (Signal-to-Noise Ratio) é uma métrica que descreve a relação entre o nível do sinal e o nível do ruído em um conversor A/D. É uma medida importante da qualidade do sinal digitalizado.

Para um conversor ideal, o SNR é dado por:

$$SNR = 6,02N + 1,76dB \tag{2.12}$$

onde N é o número de bits do conversor.

A Relação Sinal-Ruído e Distorção (SINAD) é uma métrica que inclui todos os componentes de ruído e distorção presentes no sinal. Ela é usada para avaliar a qualidade de um conversor A/D, considerando não apenas o ruído, mas também as distorções harmônicas, ela é definida como(IEEE, 2023):

$$SINAD_{dB} = 20 \times \log_{10}\left(\frac{S}{\sqrt{\sum_{f \neq f_{in}} S_i^2}}\right) \tag{2.13}$$

onde S é a potência do sinal útil e $\sum_{f \neq f_{in}} S_i^2$ é a soma da potência do ruído e das distorções harmônicas.

O ENOB (Effective Number of Bits) é uma métrica que representa a resolução efetiva de um conversor A/D, levando em conta os efeitos do ruído e de outros erros (IEEE, 2023):.

O ENOB é calculado a partir do SNR real medido do sistema:

$$ENOB = \frac{SINAD_{dB} - 1,76}{6,02} \tag{2.14}$$

O ENOB fornece uma medida mais realista da performance do conversor, refletindo a resolução efetiva que pode ser alcançada na presença de ruído e distorções.

A Distorção Harmônica Total (THD) é uma métrica que quantifica a soma das potências de todas as componentes harmônicas de um sinal em relação à potência da componente fundamental.

$$THD_{dB} = 20 \times \log_{10}\left(\frac{\sqrt{\sum_{i=2}^{\infty} S_i^2}}{S_1}\right) \tag{2.15}$$

Onde S_1 é a potência do sinal útil e S_i é a soma da potência do ruído e das distorções harmônicas. Um THD mais baixo indica que o sinal tem menos distorções harmônicas.

O Intervalo Dinâmico Livre de Espúrios (SFDR) é uma métrica que quantifica a faixa dinâmica de um conversor A/D, considerando o maior tom espúrio presente no

sinal. O SFDR é definido como a razão entre a potência da componente fundamental e a potência do maior tom espúrio, expressa em decibéis:

$$SFDR_{dB} = 20 \times \log_{10}\left(\frac{S_1}{S_{espurios}}\right) \tag{2.16}$$

Os principais tipos de conversores A/D no mercado são o por Aproximação Sucessiva (SAR), O sigma-delta (Σ-Δ), o Fash e o Rampa-Dupla

O SAR é um tipo de conversor A/D que opera de maneira iterativa. Ele utiliza um Conversor Digital-Analógico (DAC) interno para gerar um sinal analógico a partir de um valor digital. Este sinal é então comparado com o sinal de entrada. O processo é realizado bit a bit, começando pelo bit mais significativo (MSB) e indo até o bit menos significativo (LSB).

Em sua inicialização O bit mais significativo (MSB) do registrador SAR é definido como 1.logo após, O valor binário atual no registrador SAR é convertido para uma tensão analógica pelo DAC. Esta tensão é então comparada com a tensão do sinal de entrada. Se a tensão do DAC for maior que a tensão de entrada, o bit atual do registrador é definido como 0. Caso contrário, o bit permanece como 1. O processo é repetido para o próximo bit menos significativo até que todos os bits no registrador tenham sido testados. Após todos os bits serem testados, o valor no registrador SAR é a representação digital do sinal analógico de entrada. Este método é eficiente e rápido, pois o número de iterações é igual ao número de bits de resolução do conversor, tornando o tempo de conversão previsível. Além disso, os conversores SAR são amplamente utilizados devido à sua simplicidade relativa e desempenho adequado para muitas aplicações

Os conversores A/D Sigma-Delta são uma classe de conversores analógico-digital que utilizam a técnica de oversampling para obter alta resolução. Eles são amplamente utilizados em aplicações que requerem alta precisão, como medições científicas, instrumentação médica e áudio digital (ANDRADE et al.,).

Seu processo consiste em integração, onde a tensão de entrada analógica é integrada durante um certo tempo. A Quantização,onde a parte analógica do conversor é responsável pela quantização e modulação do sinal de entrada O conversor Sigma-Delta utiliza o princípio da realimentação do sinal quantizado. Isso resulta em um filtro passa-baixas para o sinal e passa-altas para o ruído de quantização. A parte digital do conversor é responsável pela decimação e filtragem. A alta resolução é alcançada pelo processo de decimação (redução da taxa de amostragem) e filtragem digitais1.

De acordo coo (SEDRA; SMITH, 2014), um conversor ADC flash é extremamente rápido e funciona ao comparar o sinal de entrada com uma série de tensões de referência geradas por divisores resistivos. Ele utiliza uma rede de comparadores, onde cada um

compara o valor de entrada com uma tensão de referência específica. Quando o valor de entrada ultrapassa um determinado nível de referência, o comparador correspondente muda seu estado. Como o processo ocorre simultaneamente em todos os comparadores, o ADC flash é muito rápido, permitindo conversões em apenas um ciclo de clock. Contudo, essa velocidade tem um custo, já que o número de comparadores cresce exponencialmente com a resolução, tornando-o caro e complexo para altas resoluções.

Já o conversor ADC de rampa dupla (ou "dual slope") segundo (SEDRA; SMITH, 2014) é bastante utilizado em medidores de precisão, como multímetros digitais, devido à sua alta precisão, apesar de ser mais lento. Ele funciona de maneira diferente, integrando o sinal de entrada por um período fixo de tempo, durante o qual o capacitor armazena uma carga proporcional à tensão de entrada. Após esse período de integração, o capacitor é descarregado por meio de uma tensão de referência fixa e o tempo necessário para descarregar o capacitor é medido. Esse tempo é diretamente proporcional ao valor da tensão de entrada. Como o processo é dependente de uma média do sinal de entrada, os conversores de rampa dupla são imunes a ruídos e oferecem alta precisão, embora o tempo de conversão seja relativamente longo.

Parte III

Decisões de projeto e implementação do trabalho

3 Decisões de engenharia a cerca do projeto

3.1 Problemas atuais levantados nas Eclusas

3.1.1 Descrição dos problemas a serem tratados

Tomando os requisitos levantados pela equipe que fica dentro das eclusas em conjunto com a equipe da sede, foram encontrados alguns problemas foram levantados para a melhor resolução do sistema. Entre os problemas mais críticos que podem ocorrer nas eclusas estão a falta de controle do nível de água no sistema de poços de adução e e esgotamento, a falta de semáforo na jusante e montante, a ausência de um software de supervisório *SCADA* e o problemas de sincronização e controle nos servos motores da ponte.

Atualmente, os poços de adução e esgotamento não possuem um sistema de controle. Mesmo com sensores para monitorar os níveis de água e garantir que o processo de esvaziamento e enchimento das câmaras da eclusa ocorra de maneira controlada, todo o processo é feito de modo manual. A falta de um sistema instrumentalizado e eletronicamente controlado gera a operação ineficiente, podendo causar erros como o total enchimento dos poços, sendo problemático a retirada de água.

O *SCADA* é um sistema crítico para a supervisão e controle das operações de uma eclusa. A ausência de um software *SCADA* pode resultar em monitoramento ineficiente, dificultando a monitoração dos processos em tempo real e reduzindo a capacidade de resposta a falhas e emergências. Além disso, há dificuldade na análise de dados, pois a coleta e análise de dados operacionais tornam-se limitadas, dificultando a identificação de padrões de desempenho e áreas que necessitam de melhorias. A falta de automação é outra consequência, já que a ausência de um software *SCADA* impede a implementação de automação nos processos, o que poderia aumentar a eficiência e reduzir erros humanos.

Os semáforos são dispositivos essenciais para regular o tráfego de embarcações na entrada e saída das eclusas. A ausência desses semáforos pode causar confusão no tráfego, pois sem sinalização clara, as embarcações podem não saber quando é seguro entrar ou sair da eclusa, levando a possíveis colisões e congestionamentos. Isso resulta em atrasos operacionais, pois a falta de controle visual pode levar os operadores a se comunicarem manualmente com as embarcações para coordenar a passagem. Além disso, os riscos à segurança aumentam, já que a ausência de semáforos eleva a possibilidade de acidentes devido à falta de orientação clara.

Em suma, esses problemas destacados – falta de controle nos poços, problemas de

sincronização nos servos motores da ponte, falta de um software de supervisório SCADA e falta de semáforo na jusante e montante – apresentam sérias consequências para a operação de eclusas, comprometendo a eficiência, a segurança e a manutenção das estruturas. A identificação e correção desses problemas são essenciais para garantir a operação segura e eficiente de uma eclusa.

3.2 Análise dos problemas

3.2.1 Análise do tratamento da problemática

Para analise dos problemas descritos, foram separados os problemas em 4 módulos, um módulo de inovação, sistemas auxiliares, supervisório e ponte. basicamente, o módulo de inovação irá resolver um item que antes não foi resolvido, que é a automação dos poços das eclusas. O sistema auxiliar será o controle dos semáforos que existem na montante e da jusante de uma eclusa. O supervisório será feito a partir de um software de código aberto com existência no mercado para a melhor observação do operador e a análise da ponte será feita através do estudo dos motores de passo.

3.2.2 Módulo 01: Inovação

Ao levantar os requisitos dos problemas das eclusas com a equipe de manutenção, um dos problemas foi o de controle dos níveis dos poços de adução e esgotamento. Um poço de adução é um poço onde se recebe a água bruta ou da jusante ou da montante e logo após é jogada para a câmara de eclusagem. Um poço de esgotamento é um poço onde a água da baixa da eclusagem é descarregada.

Cada câmara contém 4 sensores redundantes de pressão que variam de 4 a 20mA e duas bombas, uma de 30cv na adução e outra de 45cv no esgotamento, como mostrado na figura 2.4. a figura 3.2.2 mostra o sensor utilizado pelos poços. Segundo o datasheet do produto (GULTON, 2023), o Invólucro possui aço inoxidável AISI 304, possui Sinal de saída de 4 a 20mA, Resistência de carga de 0 a 1200 Ω, 600 Ω para 24Vcc, Alimentação de 12 a 36Vcc, sendo a alimentação típica de 24Vcc e Temperatura de operação: 0 a 60ºC.

Para a automação dos poços, foi traçado uma lógica do processo de adução e esgotamento. A figura 3.2.4 mostra o funcionamento básico. O sistema em si deve detectar o aumento do nível de água, verificar as fases da rede elétrica (para evitar o acionamento com fase invertida, fazendo com que a bomba coloque água no poço) e acionar o motor via inversor. Para o teste em paradigma será feito um modelo em escala menor, sendo seu passo a passo descrito no capítulo 04.

Figura 10 – Transmissor de pressão usado nos poços usado como sensor de nível adaptado de (GULTON, 2023)

3.2.3 Módulo 02: Semáforos

Os semáforos são componentes essenciais na operação de eclusas, garantindo a segurança e a eficiência do tráfego de embarcações. Eles regulam a entrada e a saída das embarcações, prevenindo colisões e otimizando o fluxo de tráfego.

Os semáforos em uma eclusa funcionam como sinais de trânsito que regulam a passagem de embarcações. Situados na jusante (parte inferior do rio) e montante (parte superior do rio) da eclusa, eles indicam aos navegantes quando é seguro entrar e sair das câmaras. Esse controle é fundamental para coordenar o trânsito de embarcações, especialmente em vias navegáveis movimentadas, onde a segurança e a eficiência são prioridades.

Para garantir a operação segura e eficiente das eclusas, é crucial implementar e manter adequadamente os sistemas de semáforos. Para o projeto se usará um microcontrolador ligado aos semáforos e ao sistema de supervisório.

A integração dos semáforos aos sistema de supervisório deve permitir que suas operações sejam coordenadas automaticamente com outros processos, como o enchimento e esvaziamento das câmaras. Essa integração melhora a eficiência e reduz a necessidade de intervenção manual.

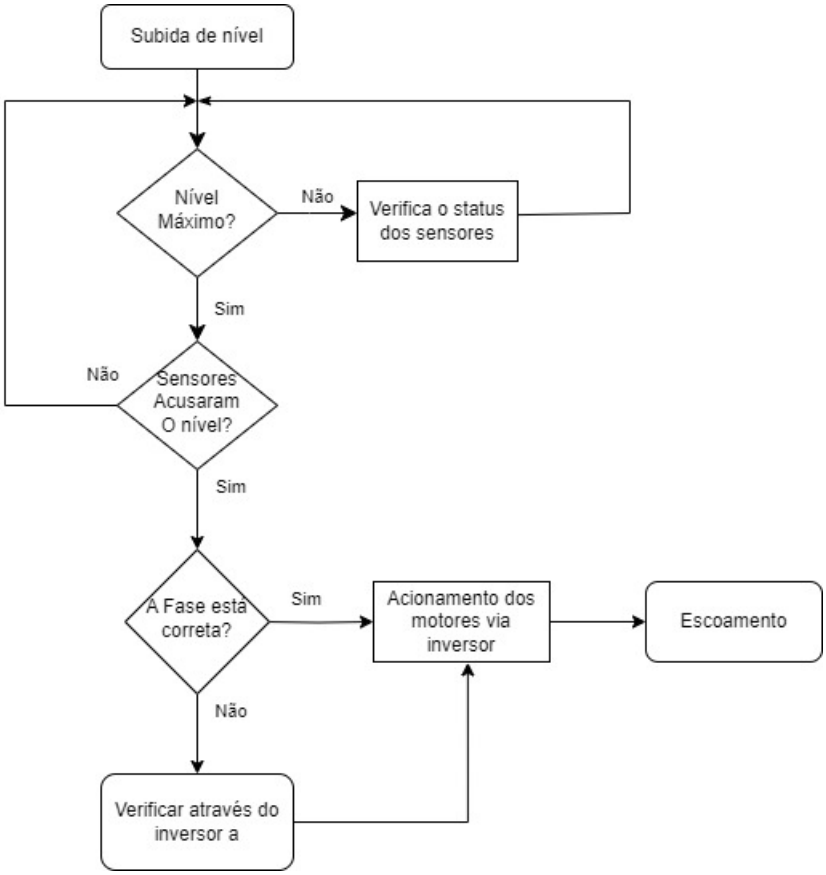

Figura 11 – Lógica básica do processo de adução e de esgotamento

3.2.4 Módulo 03: O Supervisório

A operação eficiente e segura de uma eclusa depende de um controle rigoroso e da supervisão contínua de vários parâmetros operacionais. Um sistema supervisório *SCADA* é importante para o monitoramento e o controle dos parâmetros em tempo real.

Figura 12 – SCADABR logo adaptado de (SCADABR, 2020)

Os sistemas supervisórios *SCADA* são projetados para coletar dados em tempo real, processá-los e exibir informações críticas para os operadores. Em uma eclusa, esses sistemas desempenham várias funções vitais, das quais estão o Monitoramento em tempo real, o controle e a automação dos processos, o registro de dados em um banco de dados para análise posterior, alarmes e notificações e a integração de vários sistemas.

O *SCADA* permite a visualização e o monitoramento em tempo real dos níveis de água, status de portas, bombas, válvulas e outros componentes críticos da eclusa. Isso possibilita uma resposta imediata a qualquer anomalia, minimizando o risco de falhas operacionais. Os sistemas *SCADA* facilitam o controle automatizado das operações da

eclusa, como o enchimento e esvaziamento das câmaras, abertura e fechamento de portas e operação de sistemas auxiliares. A automação reduz a dependência de intervenção manual, aumentando a eficiência e a precisão das operações.O *SCADA* armazena dados históricos que são essenciais para análises de desempenho, identificação de tendências e planejamento de manutenção preventiva. Isso ajuda na tomada de decisões informadas e na melhoria contínua das operações.O sistema pode ser configurado para gerar alarmes e notificações em caso de condições anômalas, como níveis de água fora dos limites, falhas em equipamentos ou situações de emergência. Isso permite uma resposta rápida e eficaz para mitigar riscos.

Um sistema *SCADA* pode ser integrado com outros sistemas de controle e gestão, proporcionando uma visão holística das operações e facilitando a coordenação entre diferentes componentes e processos da eclusa. De acordo com (SCADABR, 2020) O SCADABR é uma solução de código aberto, o que o torna uma opção economicamente viável, especialmente para projetos com orçamento limitado. A ausência de licenças de software caras reduz significativamente os custos iniciais e de manutenção.Como um projeto de código aberto, o SCADABR possui uma comunidade ativa de desenvolvedores e usuários que contribuem com melhorias contínuas, documentação e suporte. Esse aspecto comunitário garante que o sistema evolua e se mantenha atualizado com as melhores práticas do setor.

Figura 13 – Vários dashboards produzidos pelo SCADABR adaptado de (Companhia de Saneamento Catarinense)

3.2.5 Modulo 04: Ponte

A ponte de uma eclusa desempenha um papel vital na integração de transporte terrestre e fluvial, permitindo a passagem de veículos e pedestres sobre a estrutura enquanto regula a entrada e saída das embarcações. A operação eficaz desta ponte depende fortemente dos servos motores, que controlam seu movimento. Este capítulo examina a

função da ponte, a importância dos servos motores, os problemas de sincronização e as melhores práticas para a sua manutenção e operação.

As pontes em eclusas são projetadas para se mover, permitindo a passagem de embarcações de diferentes alturas e garantindo o fluxo contínuo de tráfego terrestre. Existem diferentes tipos de pontes móveis, como pontes levadiças, basculantes e giratórias, cada uma projetada para atender às necessidades específicas do local.

A função principal dessas pontes inclui: Permitir a Navegação, Ao se levantar ou girar, a ponte permite que embarcações maiores passem através da eclusa, ajustando-se às variações de nível da água. Garantir a Conectividade Terrestre, Quando abaixada ou em posição fechada, a ponte proporciona um caminho seguro para veículos e pedestres, mantendo a conectividade entre as duas margens do curso d'água. Os servos motores são dispositivos críticos que controlam o movimento das pontes móveis. Eles são responsáveis por garantir que a ponte opere suavemente, de forma coordenada e segura. Os servos motores são fundamentais por várias razões. Eles fornecem um controle preciso sobre a velocidade e a posição da ponte, permitindo operações seguras e eficientes.Servos motores garantem que todos os componentes móveis da ponte operem em perfeita harmonia, evitando desalinhamentos que possam causar danos à estrutura. São projetados para suportar cargas pesadas e operar em condições ambientais adversas, garantindo a longevidade da ponte. Facilitam a automação do movimento da ponte, reduzindo a necessidade de intervenção manual e aumentando a eficiência operacional.

Para o controle dos motores será feito um estudo sobre a ponte e como uma lógica de motores pode melhorar e otimizar o processo como um todo.

4 Implementação do Hardware do Projeto

Neste capítulo serão apresentados o desenvolvimento do projeto, seus problemas ocorridos e as tomadas de decisão para implementação do código e do protótipo

4.1 Criação de um protótipo para o controle dos poços

Para teste das ferramentas de controle e automação, serão criados dois tipo de testes: Um teste de sinais e um teste de prototipagem em dimensões pequenas. O testes de sinais consiste em testar as saídas e entradas digitais e analógicas em relação ao supervisório, além de testar os registradores ModBus. Já o tetes em dimensões pequenas será a criação de um protótipo que consiste numa versão diminuta dos problemas levantados pela equipe de engenharia das eclusas.

4.2 Especificações do microcontrolador

O primeiro controlador levantado para a implementação do projeto foi o microcontrolador ATmega328P. Ele é amplamente utilizado em projetos de eletrônica devido às suas características versáteis e robustas, além de sua facilidade de encapsulamento em um sistema embarcado e fácil manutenção por terceiros. Segundo (MICROCHIP, 2018) O ATmega328P é baseado na arquitetura AVR de 8 bits e possui as seguintes especificações:

- **Arquitetura**: AVR 8-bit
- **Velocidade de Clock**: até 20 MHz
- **Memória Flash**: 32 KB (com 0,5 KB usados pelo bootloader)
- **SRAM**: 2 KB
- **EEPROM**: 1 KB
- **Pinos de I/O**: 23
- **ADC**: 10-bit, 6 canais (em encapsulamento PDIP) ou 8 canais (em encapsulamento TQFP e QFN/MLF)
- **Interfaces**: USART, SPI, I2C (TWI)
- **Timers**: 2x 8-bit e 1x 16-bit
- **Tensão de Operação**: 1.8V a 5.5V

- **Consumo de Corrente**: 0.2 mA em modo ativo a 1 MHz, 1.8V; 0.1 µA em modo Power-down
- **Temperatura de Operação**: -40°C a 85°C

4.2.1 Parâmetros de Desempenho Analógico do ATMega28p

Para a avaliação do desempenho analógico do ATmega328P, são considerados os seguintes parâmetros: Relação Sinal-Ruído (SNR), Relação Sinal-Ruído e Distorção (SINAD), Número Efetivo de Bits (ENOB), Distorção Harmônica Total (THD) e Intervalo Dinâmico Livre de Componentes Espúrias (SFDR).

4.2.1.1 Relação Sinal-Ruído (SNR)

Usando a equação 2.12, temos para o ADC do ATMEGA 328p:

$$SNR = 6.02 \times 10 + 1.76 = 61.96\,dB$$

4.2.1.2 Relação Sinal-Ruído e Distorção (SINAD)

Com a equação 2.13 achamos valor exato. Porém, Em um ADC, o SINAD é aproximadamente igual ao SNR.

Então temos:

$$SINAD \approx SNR = 61.96\,dB$$

4.2.1.3 Número Efetivo de Bits (ENOB)

O ENOB pode ser calculado utilizando a euqação 2.14. Substituindo o valor de SINAD, temos:

$$ENOB = \frac{61.96 - 1.76}{6.02} \approx 9,7\,bits$$

4.2.1.4 Distorção Harmônica Total (THD)

O THD pode ser calculado a partir das distorções harmônicas geradas pelo ADC. A equação 2.15 é a referencia para o calculo do THD. Para simplificação, segundo (MICROCHIP, 2018) podemos usar um valor típico de THD para o ATmega328P, que é aproximadamente -70 dB.

4.2.1.5 Intervalo Dinâmico Livre de Componentes Espúrias (SFDR)

O SFDR pode ser calculado de acordo com a equação 2.16. Para o ATmega328P, segundo (MICROCHIP, 2018) o SFDR típico é aproximadamente 61 dB.

4.2.2 Resumo dos Valores Calculados

- **SNR (Relação Sinal-Ruído)**: 61.96 dB
- **SINAD (Relação Sinal-Ruído e Distorção)**: 61.96 dB
- **ENOB (Número Efetivo de Bits)**: 10 bits (ideal), 9,7 bits (prático)
- **THD (Distorção Harmônica Total)**: -70 dB
- **SFDR (Intervalo Dinâmico Livre de Componentes Espúrias)**: 61 dB

Esses valores são típicos para um ADC ideal de 10 bits, mas as imperfeições práticas no ATmega328P podem causar variações.

4.3 Problemas de desenvolvimento com o ATmega328p

Com o andar do desenvolvimento do projeto, houveram problemas de desenvolvimento com o ATmega328p e o interfaceamento com a biblioteca <Modbus.h>.

É de suma importância de como um comando Modbus é realizado e como em um controlador ele é enviado. Como discutido no capítulo 02, Um comando Modbus clássico é uma palavra que contém de 8 a 32 bits em data. Os 7 primeiros bytes (ou 56 bits) de uma mensagem transmitida em modbus são a composição PDU como mostrado na figura 4.3. Esse valor é fixo, sendo os dois primeiros Bytes o tipo de transação a ser feita, mais dois bytes de identificação de protocolo mais o tamanho da palavra a ser enviada e um identificador unitário. Como descrito no manual do modbus (ModBus Organization, 2006) a escrita de bobina possui Hexadecimal 0x05, sendo em binário 00000101b. Esse valor é precedido pelo bit de paridade, que pode ser 0 ou 1 e pela escrita em bobina, que também é 0 ou 1. Portanto neste exemplo, um simples comando de identificação de protocolo enviada é 1100000101b, uma palavra simples de 10 bits. Esses valores são o tempo todo modificados e enviados via TCP/IP ou serial RTU.

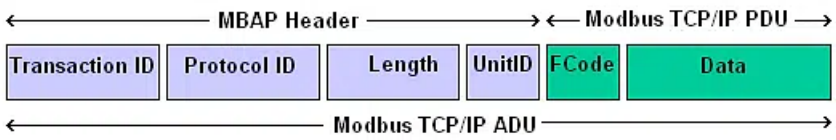

Figura 14 – Palavra de comunicação Modbus TCP adaptado de (ModBus Organization, 2006)

O compilador do ATMega328P padrão é o AVR-GCC. Este compilador para esta versão de microcontrolador não possui uma implementação de Vector, sendo que a figura

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <Arduino.h>
#include <HardwareSerial.h>

struct MODBUS_BUFFER{
    size_t size;
    char data[];
};

struct MODBUS_BUFFER* CreateBuffer(size_t data_size){
    struct MODBUS_BUFFER* buffer = (struct MODBUS_BUFFER*)malloc(sizeof(struct MODBUS_BUFFER)+ data_size);
    buffer -> size = data_size;
    return buffer;
}

void setup(){
    Serial.begin(9600);
}

void loop(){
    const char* modbus_string = "1100000101"; //mensagem modbus flexível em C
    size_t string_tam = strlen(modbus_string)+1;

    struct MODBUS_BUFFER* buffer = CreateBuffer(string_tam);
    //enviado a mensagem alocada dinamicamente na memória do AVR
    Serial.write(memcpy(buffer->data, modbus_string, string_tam));
    free(buffer);
}
```

Figura 15 – Código com alocação dinâmica de uma string no ATMega328p

4.3 um código onde para enviar uma string modbus de forma serializada usando a alocação de memória.

Como problema com esse tipo de abordagem de código é ter problemas com a memória dinâmica do AVR e com possiveis erros e *overflows*. Uma solução para esse problema é a implementação do método $<Vector>$ em que consiste em um array de tamanho dinâmico. A figura 4.3 mostra uma implementação em que se usa menos da manipulação da memória física.

```cpp
#include <iostream>
#include <vector>
#include <Arduino.h>
#include <HardwareSerial.h>

std::vector<char> CreateBuffer(const char* modbus_string) {
    size_t string_tam = strlen(modbus_string) + 1; // tamanho da string + '\0'
    std::vector<char> buffer(string_tam);
    memcpy(buffer.data(), modbus_string, string_tam);
    return buffer;
}

void setup() {
    Serial.begin(9600);
}

void loop() {
    const char* modbus_string = "1100000101"; // mensagem Modbus flexível em C++
    std::vector<char> buffer = CreateBuffer(modbus_string);

    // Enviando a mensagem armazenada no buffer dinâmico
    Serial.write(buffer.data(), buffer.size());
}
```

Figura 16 – Código com implementação de Vector em um microcontrolador

Com isso, foi escolhido um segundo microcontrolador para a implementação do projeto. O objetivo era um microcontrolador que assim como o ATMega328p pudesse ter fácil manutenção, fácil implementação por parte dos engenheiros com alto código legado. Para essa tarefa foi escolhido o ESP32 com o encapsulamento do tipo WROOM-32, encontrado em kits do tipo NodeMCU.

4.4 O microcontrolador ESP32

A ESP32 WROOM-32 é um dos módulos mais populares da família ESP32, fabricado pela Espressif Systems. De acordo com (Espressif Systems, 2023) Ele combina Wi-Fi e Bluetooth, oferecendo uma solução completa para aplicações IoT (Internet of Things). Abaixo estão as especificações principais desse módulo de acordo com o datasheet (Espressif Systems, 2023):

4.4.1 Dados do processador

- **Arquitetura:** Xtensa® dual-core 32-bit LX6 microprocessor.
- **Velocidade de Clock:** Até 240 MHz.
- **Memória Flash:** 4 MB (embutido).
- **SRAM:** 520 KB.
- **ROM:** 448 KB.
- **Memória RTC (Real-Time Clock):** 16 KB.

Além disso, o ESP32 possui itens adicionais que o ATMega328p não possui, como:

4.4.2 Conectividade

- **Wi-Fi:**
  - 802.11 b/g/n.
  - Frequência: 2,4 GHz.
  - Suporte para modo Access Point, Station e modo dual.
- **Bluetooth:**
  - Bluetooth v4.2 BR/EDR.
  - Bluetooth Low Energy (BLE).

4.4.3 Periféricos

- **GPIOs:** 34 pinos de GPIO.
- **ADC:** 18 canais de 12 bits.
- **DAC:** 2 canais de 8 bits.

- **Interfaces:** SPI, I2C, I2S, UART, SDIO, PWM.
- **PWM:** 16 canais.
- **Sensores Internos:** Sensor de toque capacitivo, sensor Hall.

4.4.4 Alimentação

- **Tensão de Operação:** 3,0V a 3,6V.
- **Consumo de Energia:**
  - Modo ativo: $\sim$ 160 mA @ 240 MHz.
  - Modo de espera profundo (Deep Sleep): $<$ 10 µA.
  - Modo de escuta (Light Sleep): $\sim$ 0.8 mA.

4.4.5 Dimensões Físicas

- **Tamanho:** 18 mm x 25,5 mm x 3 mm.
- **Peso:** Aproximadamente 2,5 g.

4.4.6 Certificações

- **Certificações de Regulamentação:** FCC, CE, IC, TELEC, KCC, SRRC, NCC.

4.4.7 Outras Características

- **Temperatura de Operação:** -40°C a 85°C.
- **SoC Integrado:** ESP32-D0WDQ6 ou ESP32-D0WD.
- **Suporte para OTA (Over-the-Air) Updates:** Sim.

4.4.8 Aplicações Típicas

- Dispositivos IoT.
- Automação residencial.
- Sistemas de monitoramento e controle remoto.
- Aplicações com Bluetooth e Wi-Fi integrados.

O ESP32 WROOM-32 é conhecido por sua versatilidade e alto desempenho, tornando-o ideal para uma ampla gama de aplicações, especialmente aquelas que exigem conectividade sem fio. Porém, usaremos seu modo RS485 para transmissão de dados para os testes.

4.4.9 Parâmetros de Desempenho Analógico

As especificações de desempenho analógico, como SNR, SINAD, ENOB, THD e SFDR, são importantes para avaliar a qualidade dos conversores analógico-digital (ADC) e digital-analógico (DAC) integrados em microcontroladores como o ESP32 WROOM-32. A seguir será mostrado essas especificações com base nas informações disponíveis para o ESP32:

Relação Sinal-Ruído (SNR)

Utilizando a esquação 2.12, temos o valor próximo de acordo com o datasheet (Espressif Systems, 2023). Ao se calcular os valores , temos para o ADC de 12 bits:

$$SNR_{ideal} = 6.02 \times 12 + 1.76 = 72.24dB$$

Na prática, o SNR observado pode ser menor devido a ruídos e imperfeições no sistema. Para o ESP32 WROOM-32, um SNR típico observado via datasheet é em torno de 70 dB.

4.4.9.1 Relação Sinal-Ruído e Distorção (SINAD)

O SINAD inclui tanto o ruído quanto a distorção no cálculo, Utilizando a equação 2.13, e de acordo com o datasheet e com a relação descrita em (IEEE, 2023) e (SEDRA; SMITH, 2014), temos que

$$SINAD \approx SNR - 2dB$$

Portanto, com um SNR de 70 dB, o SINAD pode ser estimado como:

$$SINAD \approx 70 - 2 = 68dB$$

No entanto, devido a imperfeições, um valor típico para o ESP32 de acordo com seu datasheet é 67 dB.

4.4.9.2 Número Efetivo de Bits (ENOB)

Usando a equação 2.14, temos que Substituindo o valor do SINAD:

$$ENOB = \frac{67 - 1.76}{6.02} = \frac{65.24}{6.02} \approx 10.83bits$$

Arredondando, temos que o ENOB típico do ESP32 é entre 10 e 11 bits.

4.4.9.3 Distorção Harmônica Total (THD)

O THD para o ADC do ESP32 de acordo com (Espressif Systems, 2023) é geralmente inferior a -70 dB, o que indica uma baixa distorção. Embora o cálculo exato do THD dependa das características específicas do sistema e da forma de onda de entrada, para o ESP32, um valor típico é inferior a -70 dB.

4.4.9.4 Intervalo Dinâmico Livre de Componentes Espúrias (SFDR)

O datasheet informa que é aproximadamente 70 dB para o ADC do ESP32. O cálculo exato do SFDR pode envolver medições detalhadas e depende do sinal de entrada e da configuração do ADC, mas para um sistema de 12 bits, o SFDR ideal pode ser próximo do SNR.

4.4.10 Resumo dos Valores Típicos para o ESP32 WROOM-32

- **SNR (Relação Sinal-Ruído):** ∼70 dB
- **SINAD (Relação Sinal-Ruído e Distorção):** ∼67 dB
- **ENOB (Número Efetivo de Bits):** 10 a 11 bits
- **THD (Distorção Harmônica Total):** < -70 dB
- **SFDR (Intervalo Dinâmico Livre de Componentes Espúrias):** ∼70 dB

Os valores exatos podem variar dependendo das condições de operação, como a frequência de amostragem, a tensão de alimentação e o ambiente de trabalho. Além disso, as características do PCB (como layout e componentes periféricos) podem impactar o desempenho analógico do ESP32 WROOM-32.

A tabela 1 mostra a diferença entre os controladores ATmega328p e ESP32, o que mostra a superioridade da ESP32 para a aquisição de dados analógicos em relação ao outro controlador. E a tomada de decisão foi a escolha da ESP32 para prosseguimento do projeto.

Especificações	ATmega328P	ESP32
Clock Speed (MHz)	20,0	240,0
Flash Memory (KB)	32,0	4096,0
SRAM (KB)	2,0	520,0
GPIO Pins	23,0	34,0
ADC Channels	6,0	18,0
SNR (dB)	55,0	70,0
SINAD (dB)	53,0	67,0
ENOB (bits)	8,8	10,8
THD (dB)	-70,0	-70,0
SFDR (dB)	61,0	70,0

Tabela 1 – Comparativo entre ATmega328P e ESP32

4.5 Comandos da bilbioteca modbus.h

A biblioteca `Modbus.h` usada no ESP32 é parte da implementação do protocolo Modbus, ela é uma bilbioteca criada por André Sarmento Barbosa e modifcicada por Alexander Emelianov, amplamente utilizado em sistemas industriais e de automação. O endereço da bilioteca é https://github.com/emelianov/modbus-esp8266. Esta biblioteca facilita a comunicação entre dispositivos que suportam o protocolo Modbus, como Controladores Lógicos Programáveis (CLPs), sensores, entre outros dispositivos de automação, sendo até mais fácil de se usar que a biblioteca oficial da Espressif.

Definição de Endereço e Configurações

- `ModbusRTU modbus;`
  - Cria uma instância do Modbus RTU.
- `ModbusTCP modbus;`
  - Cria uma instância do Modbus TCP.

4.5.1 Configurações Gerais

As configurações gerais servem para inicializar as instâncias do Modbus e para o recebimento dos comandos por um controlador mestre ou por um sistema de supervisório.

- `modbus.begin(uint8_t);`
  - Inicializa a comunicação Modbus. Para RTU, isso configura a porta serial.
- `modbus.master();`
  - Configura o dispositivo como mestre Modbus.

- `modbus.slave(uint8_t id);`
  - Configura o dispositivo como escravo Modbus com o ID especificado.
- `modbus.poll();`
  - Deve ser chamado regularmente no loop principal em caso de um sistema de controladores mestre-escravo. Processa as mensagens Modbus e executa ações.

4.5.2 Funções de Leitura

As funções de leitura servem basicamente para que um controlador Modbus mestre leia os comandos do controlador escravo. Em um sistema supervisório, esses comandos já são implementados por padrão.

- `modbus.readCoil(uint8_t id, uint16_t offset, cbTransaction cb, uint16_t numregs = 1);`
  - Lê o estado de uma ou mais bobinas de um dispositivo escravo.
- `modbus.readDiscrete(uint8_t id, uint16_t offset, cbTransaction cb, uint16_t numregs = 1);`
  - Lê o estado de uma ou mais entradas discretas de um dispositivo escravo.
- `modbus.readHoldingRegisters(uint8_t id, uint16_t offset, cbTransaction cb, uint16_t numregs = 1);`
  - Lê um ou mais registros de retenção de um dispositivo escravo.
- `modbus.readInputRegisters(uint8_t id, uint16_t offset, cbTransaction cb, uint16_t numregs = 1);`
  - Lê um ou mais registros de entrada de um dispositivo escravo.

4.5.3 Funções de Escrita

A função de escrita serve para que um controlador mestre escreva um comando para o controlador escravo ou uma escrita de escravo-escravo. Em um software de supervisório, a melhor forma de escrita é por meio de manipulaçao direta de registradores.

- `modbus.writeCoil(uint8_t id, uint16_t offset, bool value, cbTransaction cb);`
  - Escreve um valor booleano em uma bobina de um dispositivo escravo.

- `modbus.writeHreg(uint8_t id, uint16_t offset, uint16_t value, cbTransaction cb);`
  - Escreve um valor de 16 bits em um registro de retenção de um dispositivo escravo.
- `modbus.writeCoils(uint8_t id, uint16_t offset, uint8_t* value, uint16_t numregs, cbTransaction cb);`
  - Escreve múltiplas bobinas em um dispositivo escravo.
- `modbus.writeHregs(uint8_t id, uint16_t offset, uint16_t* value, uint16_t numregs, cbTransaction cb);`
  - Escreve múltiplos registros de retenção em um dispositivo escravo.

4.5.4 Funções de Callback

Uma função de callback serve para que um controlador mestre ou mesmo um supervisório saibam que um comando transicional é realizado (e.g. escrita em um registrador de um mestre para um escravo). O comando de callback é usado para depuração do sistema.

- `void cbTransaction(uint8_t id, uint8_t status, uint16_t* data, uint16_t len);`
  - Função de callback que é chamada quando a transação Modbus é completada. O status da transação pode ser usado para verificar o sucesso ou falha.

4.5.5 Funções para Configuração de Parâmetros

São comandos que tem como objetivo de set e preset de transmissão.

- `modbus.setBaudrate(uint32_t baudrate);`
  - Define a taxa de transmissão para Modbus RTU.
- `modbus.setTimeOut(uint16_t timeout);`
  - Define o tempo limite (timeout) para uma transação Modbus.
- `modbus.setMessageTimeOut(uint16_t timeout);`
  - Define o tempo limite para mensagens.

4.5.6 Funções para Configuração Avançada

- `modbus.onDataHandler(cbModbus cb);`
  - Define um callback que será chamado ao receber dados Modbus.
- `modbus.onErrorHandler(cbError cb);`
  - Define um callback para tratar erros de comunicação.

4.5.7 Manipulação Direta de Registros Internos

A manipulação de registradores tem como objetivo a escrita direta de um sistema de supervisório para o controlador escravo. É a forma mais simples e direta de escrita, sendo inclusive de fácil depuração. Para o uso no sistema, será utilizado os métodos de configuração e a manipulação dos registradores de entrada e saída.

4.5.7.1 Manipulação de Registros de Retenção (Holding Registers)

- `mb.Hreg(uint16_t offset, uint16_t value = 0);`
  - **Descrição:** Define ou lê o valor de um registro de retenção em uma posição específica (`offset`).
  - **Parâmetros:**
    - `offset`: Posição do registro de retenção.
    - `value`: (Opcional) Valor a ser definido no registro. Se não for fornecido, a função retorna o valor atual do registro.
  - **Exemplo:**

```
mb.Hreg(100, 12345); // Define o valor 12345 no registro de retenção 100
uint16_t value = mb.Hreg(100); // Lê o valor do registro de retenção 100
```

4.5.7.2 Manipulação de Entradas Discretas (Discrete Inputs)

- `mb.Ists(uint16_t offset, bool value = 0);`
  - **Descrição:** Define ou lê o estado de uma entrada discreta em uma posição específica (`offset`).
  - **Parâmetros:**
    - `offset`: Posição da entrada discreta.

    * `value`: (Opcional) Estado a ser definido na entrada. Se não for fornecido, a função retorna o estado atual da entrada.

  – **Exemplo:**

```
mb.Ists(10, true); // Define o estado da entrada discreta 10 para true (li
bool state = mb.Ists(10); // Lê o estado da entrada discreta 10
```

4.5.7.3 Manipulação de Bobinas (Coils)

- `mb.Coil(uint16_t offset, bool value = 0);`

  – **Descrição:** Define ou lê o estado de uma bobina em uma posição específica (`offset`).

  – **Parâmetros:**

    * `offset`: Posição da bobina.
    * `value`: (Opcional) Estado a ser definido na bobina. Se não for fornecido, a função retorna o estado atual da bobina.

  – **Exemplo:**

```
mb.Coil(5, true); // Define o estado da bobina 5 para true (ligado)
bool coilState = mb.Coil(5); // Lê o estado da bobina 5
```

4.5.7.4 Manipulação de Registros de Entrada (Input Registers)

- `mb.Ireg(uint16_t offset, uint16_t value = 0);`

  – **Descrição:** Define ou lê o valor de um registro de entrada em uma posição específica (`offset`).

  – **Parâmetros:**

    * `offset`: Posição do registro de entrada.
    * `value`: (Opcional) Valor a ser definido no registro. Se não for fornecido, a função retorna o valor atual do registro.

  – **Exemplo:**

```
mb.Ireg(50, 6789); // Define o valor 6789 no registro de entrada 50
uint16_t inputValue = mb.Ireg(50); // Lê o valor do registro de entrada 50
```

4.6 Implementação do código e testes de sinais

Para implementação do código, serão separados em 3 partes: A lógica de como os motores dos poços funcionarão, a lógica dos semáforos da jusante e montante e a lógica da ponte. O código do projeto irá usar a biblioteca `<Arduino.h>` e toda a lógica consagrada pela fundação Arduino. O objetivo de uso é a facilidade de código legado, para que um operador, ou mesmo um outro engenheiro possa trabalhar no sistema.

A lógica dos motores consiste no sistema possuir 3 motores em que são acionados com a subida do nível do sistema, conforma demonstrado na figura 3.2.4. Partindo do pressuposto da existência de sensores analógicos, o controle do sinal analógico será dado pelo registrador de retenção `mb.Hreg()`. É através dele que serão lidos os valores analógicos do sistema. Para os motores, podemos utliziar tanto registradores de entrada discreta como registradores de bobina. Como uma decisão de projeto, e por se tratar de um acionamento, será utilizado no código o acionamento de bobinas `mb.Coil()`. Para o acionamento dos semáforos da jusante e da montante, será também utilizado registradores de manipulação de boninas. Para a ponte, que consiste em um motor de passo com um conjunto de fim de curso, que seu acionamento se dará pelo registrador de coil e para os fim de curso será utilizado a manipulação de entrada discreta `mb.Ists()`.

Com isso, será usado o controlador como modo *slave* (slave ID 1) e serão separados os registradores da seguinte forma:

- Registradores dos motores dos poços e sensor: 101, 102, 103 e 104
- Registradores dos semáforos: 201, 202, 203, 204, 205 e 206
- Registradores dos itens da ponte: 301, 302, 303 e 304

A separação dessa forma serve para melhor depurar o projeto por parte dos controladores e operadores das eclusas. Será usado a diretiva de compilação `define` como boa prática de programação. Então, será feito `define MOTOR_01 101` como diretiva para que o valor do registrador seja salvo na variável diretiva. Com isso, os métodos `mb.addCoil()`, `mb.addHreg()`, `mb.addIreg()` serão adicionados no `void setup()` do código.

A figura 4.6 mostra as GPIOs disponíveis para o controlador, suas funções analógicas e digitais, assim como as portas de comunicação I²C e UART (Serial). Para igualar os registradores com os pinos da ESP32, utilizaremos os pinos 39,34, 35 e 36 para os registradores 101 a 104; pinos 16, 17, 18, 19, 21 e 5 para os registradores 201 a 206 e pinos 33, 25, 26 e 9 para os registradores 301 a 304.

Para cada ação, será usado no *loop* principal quatro funções. A primeira será `lerSensor()`, que tem por objetivo ler o sensor analógico pelo método `analogRead()`,

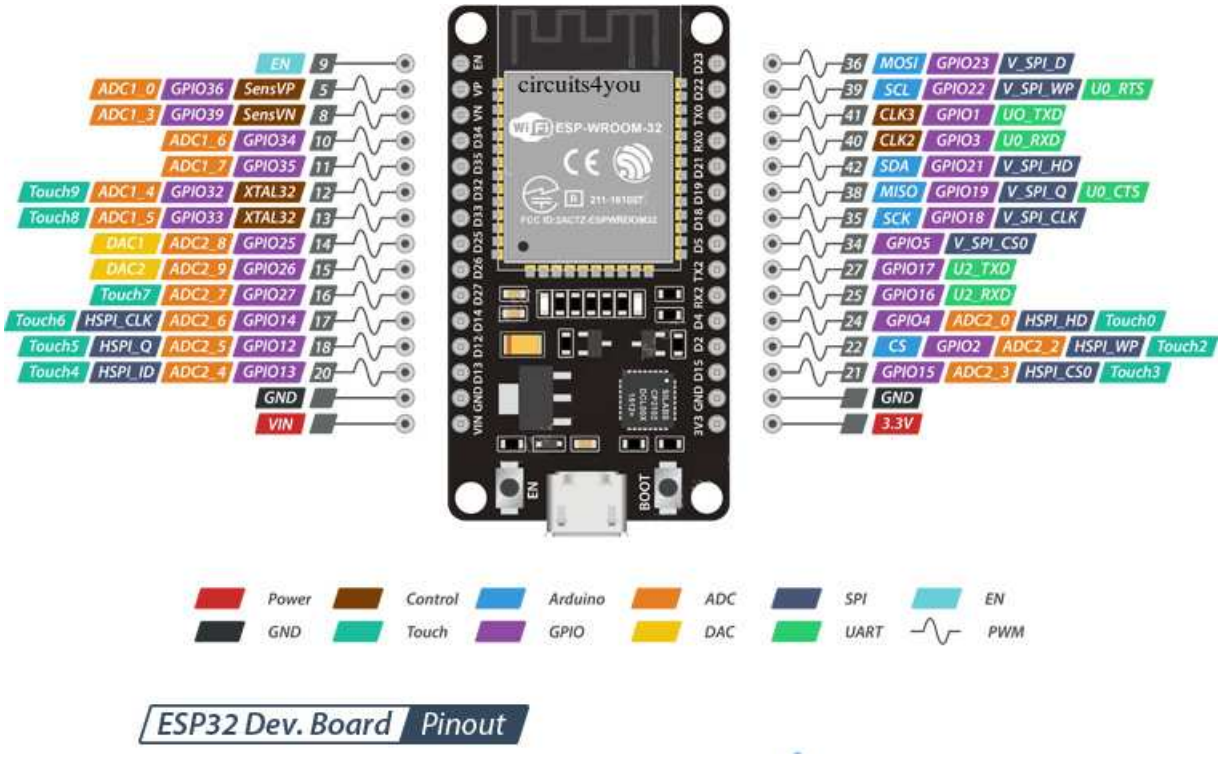

Figura 17 – Pinagem do esp32 adaptado de https://circuits4you.com/)

com a leitura do pino do sensor e a escrita no registrador (`analogRead(SENSOR_PIN, mb.Hreg(SENSOR))`. A segunda e a terceira função `acionarMotor()` e `acionarSemaforos()` possuem o mesmo objetivo, que é a escrita nas coils dos valores dos motores e dos semáforos. Utilizando o método `digitalWrite(pino, Registador)` será a forma de escrita via controlador.

Para o método `acionarPonte()`, será adicionado um botão do tipo digital no supervisório. este botão será lido pelo controlador pelo registrador 302 e a partir do acionamento será movido o motor. Para a movimentação do motor de passo, será usado a biblioteca `<AccelStepper.h>`. Os valores do motor serão configurados pelo construtor `AccelStepper motor(AccelStepper::DRIVER, MOTOR_PASSO_EN_PIN, MOTOR_PASSO_DIR_PIN)` e no *setup* serão configurados a velocidade do motor e sua aceleração. A função então testará se o botão digital será acionado, e se acionado, verificar os fins de curso e assim mover o motor.

O Anexo A possui o código completo para ser implementado e compilado.

4.7 Implementação o protótipo

Para implementação do projeto para simulação de sinais e para a criação de um protótipo de apresentação, o projeto será prototipado em uma placa do tipo *protoboard*. Os motores para a simulação das bombas serão três bombas DC de operação de 5V DC.

Para os semáforos, serão utilizados um conjunto de 6 *LEDs* de cores Vermelho, Amarelo e Verde e para o motor da ponte será usado um servo motor DC. O teste de paradigma tem como objetivo a demonstração de resultados que podem ser usados em um modelo de maior escala usando o mesmo código com poucas modificações.

4.7.1 Acionamento das bombas DC

Para o acionamento das bombas DC, será realizado o *drive* do motor DC. De acordo com (HASHID, 2014) Um *drive* de motor DC é um dispositivo eletrônico usado para controlar a velocidade, direção e torque de motores de corrente contínua (DC). O *drive* regula a quantidade de corrente e a tensão fornecida ao motor, ajustando a potência de acordo com a necessidade de aplicação.

Para a aplicação do sistema, o acionamento será direto (Ou seja, sem o uso de PWM). O *drive* servirá para a proteção do circuito e para o melhor acionamento dos motores. Para isso, um transistor do tipo MOSFET será usado como uma chave eletrônica para o acionamento dos motores DC. Para maior segurança, um diodo será colocado entre os polos do motor para aumentar a segurança contra corrente reversa e um divisor de tensão será colocado para que a tensão seja sempre 3.3 volts no *Gate*.

4.7.1.1 Escolha de um transistor para o acionamento

A escolha do transistor a principio foi uma escolha de um componente comumente usado para realizar o *drive* de motores. A primeira escolha foi do MOSFET IRF1404, porém, de acordo com o datasheet do IRF1404 (RECTIFIER, 2015) a tensão V_{GS} de *threshold* é em torno de 2 a 4 Volts. Para um bom acionamento, ou seja, para funcionamento em modo de saturação, a tensão V_{GS} tem que ser maior que a tensão tensão V_{GS} de *threshold*, e o IRF1404 tipicamente possui uma tensão de 10V para que a resistencia $R_{DS(on)}$ seja mínima.

Com isso, foi realizado uma pesquisa de componentes no mercado para o acionamento e o mais adequado é utilizar um transistor do tipo *Logic-Level*, onde seu acionamento se dá por níveis lógicos de 3,3V ou 5V e seu V_{GS} de *threshold* seja menor que V_{GS}. Então como resultado foi usado o MOSFET IRLZ44N, que seu $V_{GS(Th)}$ é entre 1 e 2V.

4.7.2 Esquemático dos itens do projeto

A figura 4.7.2 mostra como ficou o resultado da montagem de cada uma das bombas DC no sistema. Para o aumento de proteção, foi usado um diodo 1N4007 para evitar corrente reversa e um divisor de tensão na entrada para que o valor seja fixo em 3,3V e dar mais uma camada de proteção ao sistema. Para a simulação de um sensor analógico, foi colocado um potênciometro de 10kΩ conectado a GPIO G32 que fará o trabalho da

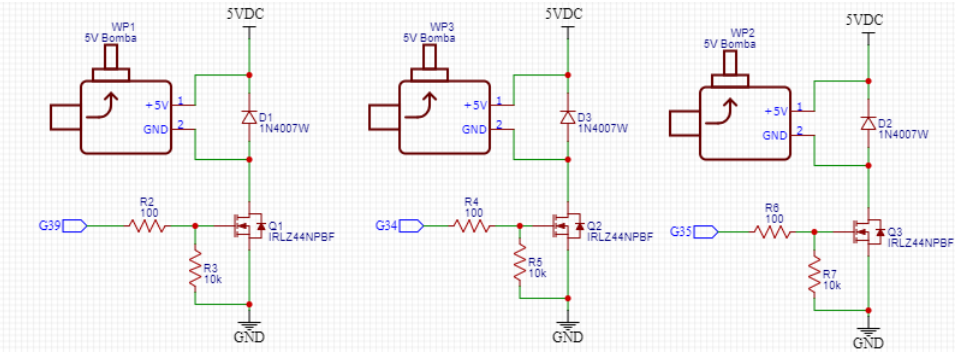

Figura 18 – Esquemático das bombas DC

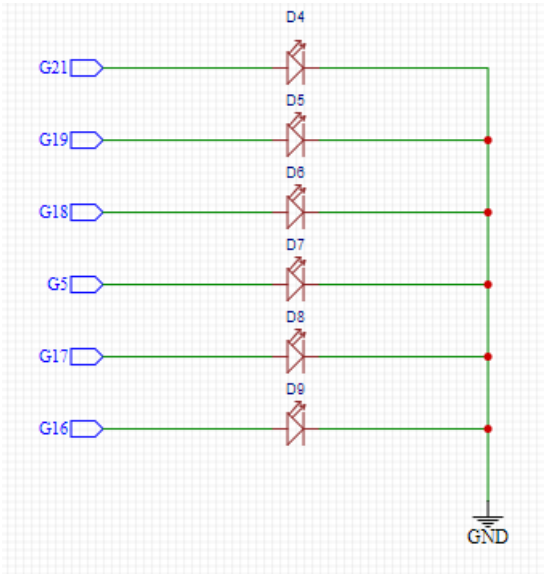

Figura 19 – Esquemático dos LEDs

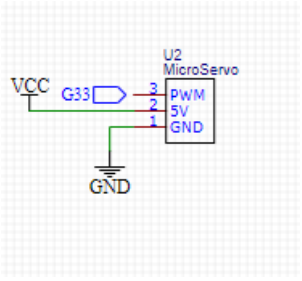

Figura 20 – Esquemático do Servo Motor

variação do sinal analógico no conversor AD. Para a colocação dos LEDs, foram usados três leds do tipo Vermelho, Amarelo e Verde. Não foram colocados resistores. O servo motor foi colocado na porta G35 e seu acionamento para este caso será por PWM.

5 Implementação do Sistema de supervisório

Neste capítulo serão apresentados a implementação do projeto e os resultados obtidos com o interfaceamento do supervisório.

5.1 Interfaceamento do protótipo com o supervisório

Ao realizar o login no sistema, o primeiro item a ser tratado é a adição de novos *datasources*.

Para adicionar um *datasource* no tipo Modbus RTU no SCADABR, o primeiro passo é acessar a interface do sistema e fazer o login. Uma vez logado, ir até o menu superior e selecionar a opção "*Datasources*"para gerenciar as fontes de dados disponíveis. Na nova tela, escolher a opção "Adicionar Fonte de Dados"e, entre as alternativas apresentadas, selecionar Modbus Serial, que corresponde ao protocolo RTU.

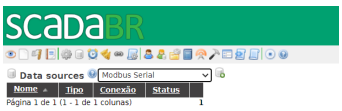

Figura 21 – Seleção do tipo de comunicação

Após selecionar o tipo Modbus Serial, a próxima tela é a de configuração do novo *datasource*. Nesta etapa, será necessário preencher algumas informações importantes para garantir que a comunicação com o dispositivo Modbus RTU funcione corretamente. A primeira parte é atribuir um nome ao *datasource*, de forma que seja facilmente identificado, como por exemplo "Controlador de Motor Modbus RTU". Em seguida, definir a porta serial à qual o dispositivo está conectado, como /dev/ttyUSB0 no caso de sistemas Linux, ou COM3 caso o sistema operacional seja Windows.

Figura 22 – Seleção dos parâmetros de configuração

A configuração da taxa de comunicação, ou *baud rate*, é crucial e deve corresponder à configuração do dispositivo escravo. O valor selecionado é de 9600 bps. O número de bits é 8N&1 e o bit de paridade é 0, o *timeoui* por padrão é de 500ms, mas para a melhor leitura, será colocado 1000ms.

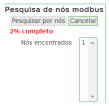

Figura 23 – Pesquisa de nós Modbus

Logo após, será identificado os nós e serão lidos cada ponto de leitura, de acordo com o registrador adicionado. A leitura é fundamental para saber se os registradores estão respondendo. Por último, cada *datasource* será colocado e adicionado para a observação no *watchlist*.

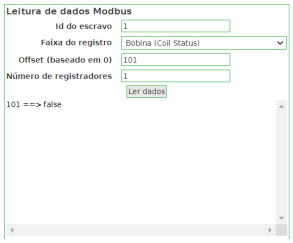

Figura 24 – Leitura dos valores digitais

Figura 25 – Leitura dos valores dos sensores

Após acessar e testar as variáveis do *watchlist* é preciso criar um novo *Dashboard* ou Tela de Visualização.

Na tela de criação do *dashboard*, haverá a opção de definir um nome para sua nova tela, algo que ajude a identificar o propósito, como "Controle de Temperatura"ou "Monitoramento de Motores". Após definir o nome, é possível começar a adicionar elementos à tela, como gráficos, indicadores, botões e outros componentes que ajudarão na visualização dos dados.

Para adicionar um componente, clicar em Adicionar Componente ou arraste e solte o item desejado da biblioteca de *widgets* para a área de edição do *dashboard*. Os componentes incluem gráficos, medidores, tabelas e outros elementos visuais que podem

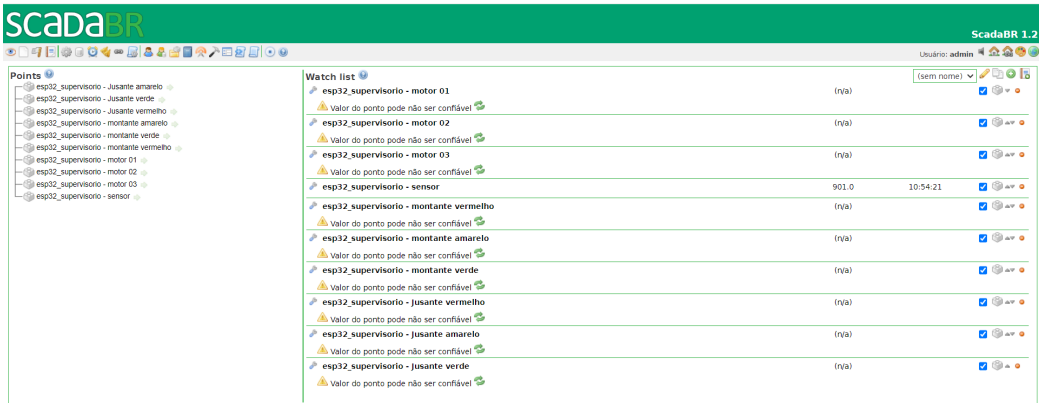

Figura 26 – *watchlist* de todos os sinais do sistema

ser configurados para exibir dados provenientes dos *datasources* que você configurou anteriormente.

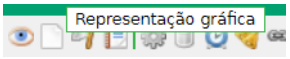

Figura 27 – Botão de representação Gráfica do sistema

Depois de adicionar um componente, é necessário configurá-lo. Para isso, será necessário clicar sobre o componente para abrir as opções de configuração. Defina as propriedades necessárias, como a fonte dos dados, o tipo de gráfico ou indicador, e quaisquer outras configurações específicas. Por exemplo, para um gráfico, você precisará selecionar o *datasource* e configurar os parâmetros de visualização, como os eixos e a escala.

Com os componentes configurados, ajuste o layout e o tamanho dos itens na tela para que eles fiquem organizados e visualmente agradáveis. Finalmente, após configurar todos os componentes e ajustar o layout, será necessário salvar as alterações no dashboard.

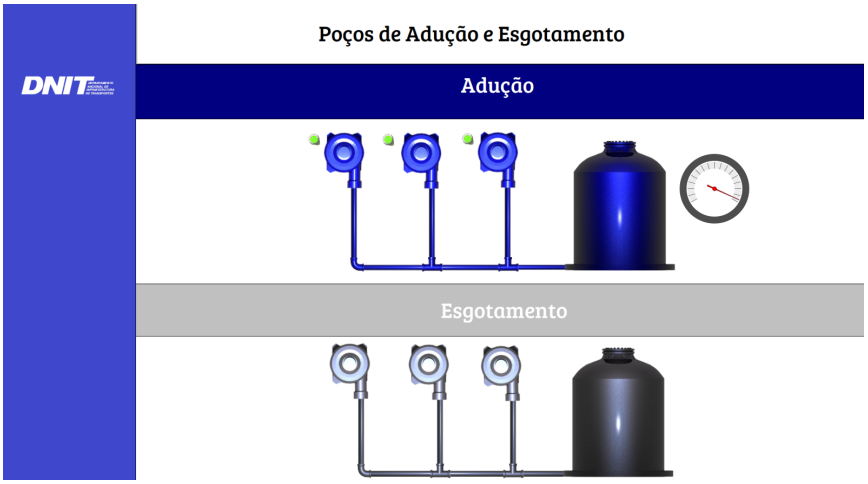

Figura 28 – Dashboard dos motores dos poços

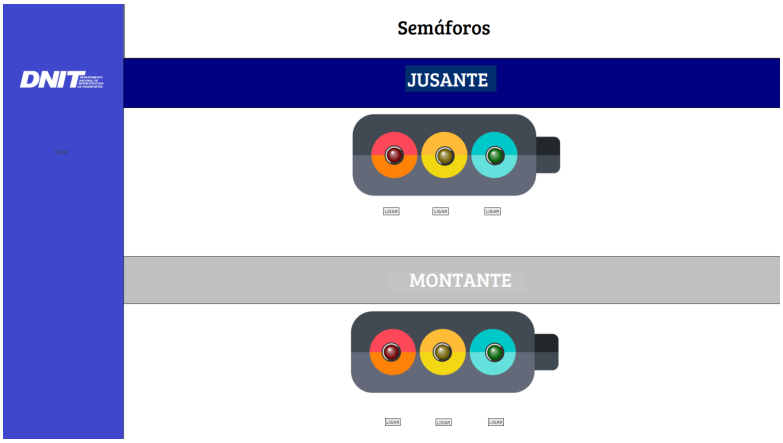

Figura 29 – Dashboard dos semáforos

Figura 30 – Dashboard da ponte

As figuras 28 a 30 mostram o modelo de cada *dashboard* feito para o sistema. O seu interfaceamento se dá pelos ícones criados pelos *dashboards* do sistema.

5.2 Resultados obtidos

Após configurados os parâmetros no SCADABR, é verificado que os resultados no geral foram satisfatórios. O controle dos motores dos poços, os semáforos e o motor da ponte foram feitos via Modbus com sucesso.

A figura 5.2 mostra o circuito montado na *protoboard*. No caso dos três motores DC, o controle do acionamento foi realizado com comandos Modbus enviados ao ESP32, e controlados via supervisório, que por sua vez acionou os motores de forma individual, de acordo com os parâmetros recebidos. Cada motor pode ser iniciado, parado de acordo com os botões acionados via SCADABR

O conjunto de semáforos, composto pelos LEDs, também foi controlado via Modbus. Os sinais de entrada determinaram o estado de cada luz, possibilitando a sincroni-

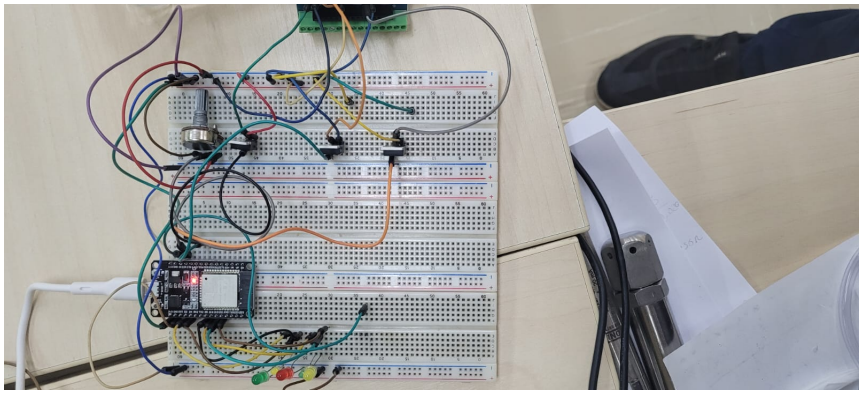

Figura 31 – Protoboard básica de montagem

zação das transições entre elas, simulando um ciclo de semáforo típico para gerenciar o tráfego na jusante e montante.

Por fim, o servo motor foi utilizado para controlar o movimento de abrir e fechar a ponte, acionado conforme os comandos recebidos pelo ESP32. Esses comandos definiram o ângulo de rotação do servo, permitindo a execução precisa de movimentos predeterminados.

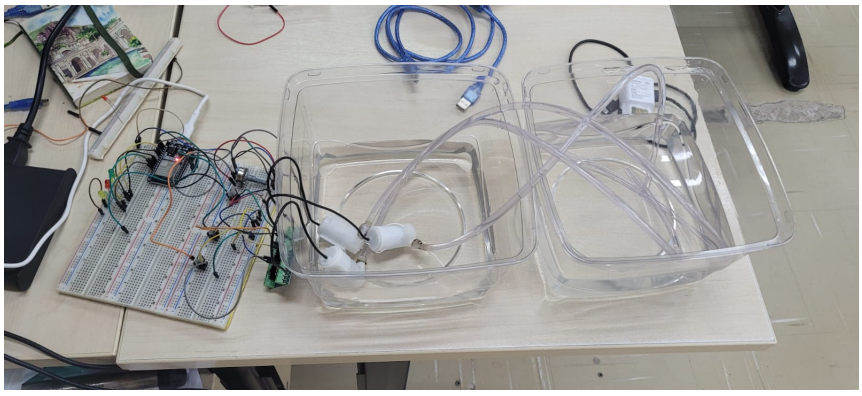

Figura 32 – protoboard com as vasilhas simulando os poços

Alguns problemas ocorreram, como o motor DC não parar após o comando Modbus. Para isso, foi identificado durante a montagem que a alimentação de 5v da ESP32 estava causando interferência com o acionamento da porta. Para a resolução desse problema foi necessário o uso de uma fonte externa para alimentar os motores de forma segragada ao controlador, sempre interligando os terras existentes.

Todos esses dispositivos funcionaram de forma coordenada, com o ESP32 gerenciando o envio e recebimento de dados via Modbus e drivers correspondentes controlando a alimentação e movimento dos motores. O sistema permitiu uma operação eficiente e integrada dos dispositivos, com feedback dos estados sendo monitorado continuamente para garantir a precisão do controle e a segurança da operação.

5.3 Itens para implementação do projeto

Para a implementação do projeto nas eclusas, alguns itens são necessários para que ocorra o sucesso na implementação.

5.3.1 Instrumentação dos sensores analógicos para eclusa

Para ler o sensor de nível que mede valores de 4 a 20 mA usando a ESP2, é necessário um circuito de interface para converter essa corrente em uma tensão que possa ser lida pelo controlador, já que os pinos analógicos da esp32 leem tipicamente tensão (0-3,3V). O circuito para a leitura de interface consiste em um resistor de carga. Para converter a corrente de 4-20 mA em uma tensão que a esp32 possa ler, será usada a transformação tensão/corrente ($R = V/i$). logo, é selecionado um resistor adequado para obter uma tensão entre 0-3,3V quando a corrente varia de 4 a 20 mA.

Com a resolução da transformação, o valor do resistor é de $1,65k\Omega$, a corrente de 4-20 mA se traduzirá em uma tensão na faixa de 1-3,3V está dentro da capacidade de leitura da esp32.

5.3.2 Instrumentação dos motores

O uso de inversores de frequência em bombas e motores é altamente recomendado por diversas razões que beneficiam tanto o desempenho quanto a eficiência dos sistemas. Primeiramente, o inversor de frequência permite um controle mais preciso da velocidade de rotação do motor, ajustando-a de acordo com a demanda real do processo. Isso resulta em uma redução significativa do consumo de energia, uma vez que o motor não precisa operar sempre em sua capacidade máxima. Além disso, o uso do inversor diminui o desgaste mecânico de componentes, como rolamentos e acoplamentos, ao eliminar a necessidade de partidas diretas bruscas, que causam picos de corrente e sobrecarga momentânea. Em bombas, esse controle também evita fenômenos como golpes de aríete, que podem danificar a tubulação. Outro ponto relevante é a flexibilidade operacional, pois os inversores permitem a adaptação a diferentes condições de operação, o que torna o sistema mais versátil e econômico. Portanto, o uso de inversores de frequência é uma solução inteligente para melhorar a eficiência, reduzir custos operacionais e aumentar a vida útil de motores e bombas.

5.3.3 Acionamento dos Semáforos

Para acionar cada uma dessas luzes de forma automática e segura, utilizamos relés de estado sólido, que são dispositivos eletrônicos projetados para ligar e desligar cargas elétricas sem partes móveis, oferecendo maior durabilidade e confiabilidade em comparação aos relés mecânicos.

No sistema de controle do semáforo, a ESP32 irá mandar os sinais através das saídas G5 e G16 a G21, recebe as instruções de tempo e sequência para o acionamento das luzes. Cada relé de estado sólido é conectado a uma das lâmpadas do semáforo, e a função do relé é interromper ou permitir a passagem da corrente elétrica, acionando ou desligando a lâmpada correspondente. O microcontrolador envia sinais elétricos de baixa tensão para o lado de controle do relé, que então permite ou bloqueia a passagem de corrente no circuito de alta tensão, alimentando as lâmpadas do semáforo.

Como os relés de estado sólido são mais rápidos e silenciosos, o acionamento das luzes ocorre de forma suave, sem ruídos de comutação, e sem o desgaste causado por arcos elétricos, comum em relés mecânicos. Além disso, eles apresentam menor aquecimento e oferecem maior eficiência energética.

Parte IV

Conclusões e Trabalhos futuros

6 Conclusões

Neste capítulo serão tratado as conclusões do trabalho e trabalhos futuros para melhoria do projeto para o DNIT.

6.1 Conclusão

As eclusas com um sistema embarcado aberto apresenta um potencial para ser replicado em outros sistemas, como o de atracadouros e futuramente o controle do sistema de monitoramento das obras de arte do DNIT. Agora, a eclusa passou a depender menos de soluções caras e dispendiosas, como o problema que os gestores do órgão tinham de ter que pagar R$ 40.000,00 para que o programador pudesse fazer o upload de um código em ladder.

O próprio controlador ESP32 possui a possibilidade de ser programado em Ladder, usando o pacote de software OpenPLC. Porém, o uso com a linguagem c++ mostra-se mais prática para os servidores da casa, dado ao alto uso de ferramentas de código aberto e com alto poder de código legado e documentação. O próprio uso da linguagem Arduíno, mesmo que com suas limitações, se faz necessário para que o operador com pouca experiência possa modificar o código e fazer um upload seguro, sem problemas com preço dispendioso para o erário público.

A substituição de um supervisório pago, como o *Elipse E3*, software que exige uma versão do windows antiga, para uma solução de código aberto robusto como o SCADABR que possui um suporte robusto, treinamentos periódicos, e casos de uso exitosos mostra-se exitoso para o DNIT. E atráves deste trabalho reforça a usabilidade e a manutenção do sistema.

As recomendações dada aos operadores no capítulo 5 são fundamentais para o funcionamento ótimo do sistema, e com o auxilio deste trabalho,

6.2 Trabalhos Futuros

Durante a execução do projeto, novas ideias surgiram para melhor efetivação do projeto. A primeira delas é a criação de um sistema de controle proporcional, integrativo e derivativo (PID) dos motores dos poços para melhor aproveitamento dos mesmos. O mesmo vale para o controle das válvulas da elevação de embarcação, conforme equação descrita no capítulo 2.

Um segundo item é a criação de uma PCB com todos os componentes necessários

para a automação de todos os itens do sistema de uma eclusa. Uma placa de circuito impresso única, em que seu reuso seja feito em outros projetos capitaneados pelo órgão.

O interfaceamento das câmeras com o sistema de supervisório também é um item a ser colocado, pois o supervisório permite que câmeras sejam integradas ao sistema, dando maior confiabiliade.

Novas imagens da interface do supervisório se fazem necessárias. A coordenação de TI do DNIT em conjunto com a Coordenação de comunicação podem melhorar as imagens feitas de acordo com o manual de imagem do DNIT de 2022. Com esse auxílio, se terá um supervisório visualmente melhor que o que foi mostrado neste trabalho.

Referências

ANDRADE, J. A. A. et al. Sigma-delta modulator for high resolution a/d converters. University of Brasília: Gama College – Gama-DF, Brazil. Citado na página 43.

CHAPMAN, S. J. *Fundamentos de Máquinas Elétricas*. [S.l.]: McGraw-Hill AMGH, 2013. Citado na página 38.

Espressif Systems. *ESP32 Technical Reference Manual*. Shanghai, China, no Zhangjiang Hi-tech Park, 2023. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf>. Citado 3 vezes nas páginas 57, 59 e 60.

FILHO, M. E. O. et al. Um controlador de corrente de carga para o conversor em matriz trifásico para trifásico e para o inversor de frequência sem capacitor do elo de corrente contínua. *Sba: Controle Automação Sociedade Brasileira de Automatica*, Sociedade Brasileira de Automática, v. 21, n. 5, p. 534–545, Sep 2010. ISSN 0103-1759. Disponível em: <https://doi.org/10.1590/S0103-17592010000500007>. Citado 2 vezes nas páginas 11 e 39.

GROOVER, M. P. *Automation, production systems, and Computer-Integrated Manufacturing*. [S.l.]: Pearson, 2018. ISBN 978-93-325-7249-2. Citado 2 vezes nas páginas 34 e 38.

GULTON INSTRUMENTOS DE MEDIçãO E AUTOMAçãO INDUSTRIA E COMéRCIO LTDA. *Datasheet do Sensor de Transmissão de Pressão GTP1000*. São Paulo, 2023. Disponível em: <https://gulton.com.br/produtos/transmissor-de-pressao-gtp1000ht/>. Citado 3 vezes nas páginas 11, 48 e 49.

HASHID, M. H. *Eletrônica de Potência: Dispositivos, Circuitos e Aplicações*. [S.l.]: Pearson, 2014. ISBN 978-85-430-0059-2. Citado 2 vezes nas páginas 39 e 68.

IEEE. Ieee standard for terminology and test methods for analog-to-digital converters. *IEEE Std 1241-2023 (Revision of IEEE Std 1241-2010)*, p. 1–143, 2023. Citado 3 vezes nas páginas 41, 42 e 59.

MCCARTNEY, B. L. *Inland navigation : locks, dams, and channels*. [S.l.]: American Society of Civil Enginners, 1998. ISBN 0-7844-0320-1,09780784403204. Citado na página 31.

MICROCHIP, T. I. *ATmega328P Datasheet*. Complete. Chandler, AZ, USA, 2018. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328P-Complete-Datasheet-DS40002061A.pdf>. Citado 2 vezes nas páginas 53 e 54.

ModBus Organization. *ModBus Application Protocol Specification V1.1b3*. 2006. Recuperado de <https://www.modbus.org/specs.php>. Citado 3 vezes nas páginas 11, 40 e 55.

OGATA, K. *Engenharia de Controle Moderno*. 5. ed.. ed. [S.l.]: Pearson / Prentice Hall, 2010. 92-96 p. Citado 2 vezes nas páginas 36 e 37.

RECTIFIER, I. *IRF1404 Power MOSFET Datasheet*. 2015. <https://www.infineon.com/dgdl/irf1404.pdf>. <https://www.infineon.com/dgdl/irf1404.pdf>. Citado na página 68.

SANTOS, I. C. et al. Um panorama da criação e implementação do programa nacional de recuperação, operação, manutenção e gerenciamento de eclusas (proeclusas), no brasil. *Revista ENINFRA*, v. 1, n. 2, p. 187–208, sep. 2023. Disponível em: <https://revistaeninfra.dnit.gov.br/index.php/inicio/article/view/40>. Citado 2 vezes nas páginas 32 e 33.

SCADABR. *Manual do SCADA BR*. Brasil, 2020. Accessed: 2024-07-30. Disponível em: <https://www.scadabr.com.br/manuais/Manual_SCADABR.pdf>. Citado 4 vezes nas páginas 11, 50, 51 e 91.

SEDRA, A. S.; SMITH, K. C. *Microelectronic Circuits*. 7th. ed. New York: Oxford University Press, 2014. ISBN 978-0199339136. Citado 3 vezes nas páginas 43, 44 e 59.

Anexos

ANEXO A – Primeiro Anexo

A.1 Código do sistema

```
#include <Arduino.h>
#include <ModbusRTU.h>
#include <AccelStepper.h>
#include <HardwareSerial.h>

#define SLAVE_ID 1

//definições dos registradores dos motores
#define MOTOR_01 101
#define MOTOR_02 102
#define MOTOR_03 103
#define SENSOR 104

//definições dos registradores dos semáforos
#define SEMAFORO_MONTANTE_VERMELHO 201
#define SEMAFORO_MONTANTE_AMARELO 202
#define SEMAFORO_MONTANTE_VERDE 203
#define SEMAFORO_JUSANTE_VERMELHO 204
#define SEMAFORO_JUSANTE_AMARELO 205
#define SEMAFORO_JUSANTE_VERDE 206

//definições dos registradores da ponte
#define FIM_DE_CURSO_INICIO 301
#define MOTOR_PASSO_EN 302
#define MOTOR_PASSO_DIR 303
#define FIM_DE_CURSO_FIM 304

//definições dos pinos da ESP32

#define MOTOR_01_PIN 17
#define MOTOR_02_PIN 16
#define MOTOR_03_PIN 04
#define SENSOR_PIN 02
#define SEMAFORO_MONTANTE_VERMELHO_PIN 36
#define SEMAFORO_MONTANTE_AMARELO_PIN 39
#define SEMAFORO_MONTANTE_VERDE_PIN 34
#define SEMAFORO_JUSANTE_VERMELHO_PIN 35
#define SEMAFORO_JUSANTE_AMARELO_PIN 32
#define SEMAFORO_JUSANTE_VERDE_PIN 33
#define MOTOR_PASSO_EN_PIN 25
#define MOTOR_PASSO_DIR_PIN 26
#define FIM_DE_CURSO_INICIO_PIN 14
#define FIM_DE_CURSO_FIM_PIN 12

ModbusRTU mb;

AccelStepper motor(AccelStepper::DRIVER, MOTOR_PASSO_EN_PIN, MOTOR_PASSO_DIR_PIN);
```

Figura 33 – includes)

```
AccelStepper motor(AccelStepper::DRIVER, MOTOR_PASSO_EN_PIN, MOTOR_PASSO_DIR_PIN);

void lerSensor(){
    analogRead(SENSOR_PIN, mb.Hreg(SENSOR));
}

void acionarMotor(){
    digitalWrite(MOTOR_01_PIN, mb.Coil(MOTOR_01));
    digitalWrite(MOTOR_02_PIN, mb.Coil(MOTOR_02));
    digitalWrite(MOTOR_03_PIN, mb.Coil(MOTOR_03));
}

void acionarSemaforos(){
    digitalWrite(SEMAFORO_MONTANTE_VERMELHO_PIN, mb.Coil(SEMAFORO_MONTANTE_VERMELHO));
    digitalWrite(SEMAFORO_MONTANTE_AMARELO_PIN, mb.Coil(SEMAFORO_MONTANTE_AMARELO));
    digitalWrite(SEMAFORO_MONTANTE_VERDE_PIN, mb.Coil(SEMAFORO_MONTANTE_VERDE));
    digitalWrite(SEMAFORO_JUSANTE_VERMELHO_PIN, mb.Coil(SEMAFORO_JUSANTE_VERMELHO));
    digitalWrite(SEMAFORO_JUSANTE_AMARELO_PIN, mb.Coil(SEMAFORO_JUSANTE_AMARELO));
    digitalWrite(SEMAFORO_JUSANTE_VERDE_PIN, mb.Coil(SEMAFORO_JUSANTE_VERDE));
}
```

Figura 34 – construtor do Motor de passo e funções dos poços e semáforos)

```
void acionarPonte(){

    int botão = mb.Ists(MOTOR_PASSO_EN);
    int fimAtivado = mb.Ists(FIM_DE_CURSO_FIM);
    int inicioAtivado = mb.Ists(FIM_DE_CURSO_INICIO);
    bool botaoPressionado = false;


 if (botao && !botaoPressionado) {
    botaoPressionado = true;  // Marca que o botão foi pressionado
  }

  // Só move o motor se o botão foi pressionado
  if (botaoPressionado) {
    if (!inicioAtivado && !fimAtivado) {
      // Se nenhum fim de curso estiver acionado, o motor se move para frente
      motor.moveTo(2000);  // Alvo: 2000 passos (pode ser ajustado)
      motor.run();
    }

    // Se o fim de curso final for acionado, o motor para e move para o início
    if (fimAtivado) {
      while (!inicioAtivado) {
        motor.moveTo(-2000);  // Move na direção oposta
        motor.run();
        inicioAtivado = digitalRead(mb.Ists(FIM_DE_CURSO_INICIO)) == 0;  // Verifica o fim de curso inicial
      }
    }

    // Se o fim de curso inicial for acionado, o motor para e move para o fim
    if (inicioAtivado) {
      while (!fimAtivado) {
        motor.moveTo(2000);  // Move para frente novamente
        motor.run();
        fimAtivado = digitalRead(mb.Ists(FIM_DE_CURSO_FIM)) == 0;  // Verifica o fim de curso final
      }
    }

    // Após completar o ciclo (de um fim de curso a outro), para o motor
    botaoPressionado = false;  // Reseta o estado do botão


}
```

Figura 35 – Função de movimento do motor da ponte)

```
void setup(){
    Serial.begin(9600, SERIAL_8N1);

    #if defined(ESP32) || defined(ESP8266)
        mb.begin(&Serial);
    #else
    mb.begin(&Serial);
    //mb.begin(&Serial, RXTX_PIN);  //ou use pinos direcionais RX/TX  (caso necessário)
    mb.setBaudrate(9600);
    #endif

    mb.slave(SLAVE_ID);

    //Adicionando os registradores dos poços
    mb.addCoil(MOTOR_01);
    mb.addCoil(MOTOR_02);
    mb.addCoil(MOTOR_03);
    mb.addHreg(SENSOR);

    //Adicionando os registradores dos semáforos
    mb.addCoil(SEMAFORO_MONTANTE_VERMELHO);
    mb.addCoil(SEMAFORO_MONTANTE_AMARELO);
    mb.addCoil(SEMAFORO_MONTANTE_VERDE);
    mb.addCoil(SEMAFORO_JUSANTE_VERMELHO);
    mb.addCoil(SEMAFORO_JUSANTE_AMARELO);
    mb.addCoil(SEMAFORO_JUSANTE_VERDE);

    //adiconando os registradores da ponto
    mb.addIsts(FIM_DE_CURSO_INICIO);
    mb.addHreg(MOTOR_PASSO_EN);
    mb.addHreg(MOTOR_PASSO_DIR);
    mb.addIsts(FIM_DE_CURSO_FIM);

    motor.setMaxSpeed(1000);  // Velocidade máxima (passos por segundo)
    motor.setAcceleration(500);  // Aceleração (passos por segundo²)

    //setando os modos de pinagem
    pinMode(SENSOR_PIN, INPUT);
    pinMode(MOTOR_01_PIN, OUTPUT);
    pinMode(MOTOR_01_PIN, OUTPUT);
    pinMode(MOTOR_01_PIN, OUTPUT);

    pinMode(SEMAFORO_MONTANTE_VERMELHO_PIN, OUTPUT);
    pinMode(SEMAFORO_MONTANTE_AMARELO_PIN, OUTPUT);
    pinMode(SEMAFORO_MONTANTE_VERDE_PIN, OUTPUT);
    pinMode(SEMAFORO_JUSANTE_VERMELHO_PIN, OUTPUT);
    pinMode(SEMAFORO_JUSANTE_AMARELO_PIN, OUTPUT);
    pinMode(SEMAFORO_JUSANTE_VERDE_PIN, OUTPUT);

    pinMode(FIM_DE_CURSO_INICIO_PIN, INPUT);
    pinmode(FIM_DE_CURSO_FIM_PIN, INPUT)
    pinMode(MOTOR_PASSO_DIR_PIN, OUTPUT);
    pinMode(MOTOR_PASSO_EN_PIN OUTPUT);

}
```

Figura 36 – Setup)

```
void loop(){
    mb.task();

    lerSensor();
    acionarMotor();
    acionarSemaforos();
    acionarPonte();
    yield();
}
```

Figura 37 – loop principal)

ANEXO B – Segundo Anexo

B.1 Instalação do supervisório SCADABR

O SCADA BR é um sistema de supervisão e controle amplamente utilizado em automação industrial e monitoramento de processos. Uma instalação adequada é essencial para garantir o funcionamento eficiente e seguro do sistema. Neste documento, detalharemos o processo de instalação do SCADA BR, abordando desde a configuração do ambiente necessário até a resolução de problemas comuns.

Para assegurar um desempenho satisfatório do SCADA BR, é fundamental que o sistema atenda aos requisitos mínimos de hardware descritos na documentação oficial (SCADABR, 2020). O sistema deve contar com um processador Intel Core i5 ou equivalente, 8 GB de memória RAM e pelo menos 10 GB de espaço em disco disponível para a instalação e armazenamento de dados.

Além das exigências de hardware, é necessário atender aos requisitos de software. O SCADA BR pode ser executado em sistemas operacionais como Windows 10 ou versões superiores, ou em distribuições Linux que suportem o servidor Apache Tomcat. O Java Development Kit (JDK), na versão 8 ou superior, também é um componente essencial para o funcionamento do SCADA BR. O JDK pode ser baixado diretamente do GitHub do SCADA BR, sendo recomendável escolher uma versão compatível com o sistema.

A instalação do JDK segue um processo simples. Após baixar o instalador, basta executá-lo e seguir as instruções fornecidas. Após a instalação, é necessário configurar as variáveis de ambiente, apontando para o diretório onde o JDK foi instalado e garantindo que o diretório bin do JDK esteja incluído no PATH, para que os comandos Java possam ser executados corretamente a partir do terminal. No Windows, essa configuração pode ser feita por meio do Painel de Controle, na seção de Configurações Avançadas do Sistema, dentro das Variáveis de Ambiente.

Com o JDK instalado e configurado, é possível prosseguir com a instalação do SCADA BR. Primeiramente, o arquivo compactado do SCADA BR deve ser extraído para um diretório à escolha do usuário. Em seguida, é necessário executar o arquivo de instalação, que pode ser o "setup.exe"no Windows ou o script "install.sh"no Linux. As instruções fornecidas pelo instalador guiarão o processo até a sua conclusão.

Um dos componentes essenciais para o funcionamento do SCADA BR é o banco de dados, utilizado para armazenar informações e configurações. MySQL e PostgreSQL são as opções mais recomendadas. Caso o banco de dados ainda não esteja instalado, será necessário instalá-lo e criar um banco de dados específico, junto com um usuário com

as permissões adequadas. Após essa configuração inicial, o arquivo de configuração do SCADA BR, geralmente localizado no diretório "config/application.properties", deve ser editado para incluir as informações referentes ao banco de dados.

Com o SCADA BR instalado, algumas configurações adicionais podem ser necessárias, como a definição da porta de comunicação que será utilizada para a integração com outros sistemas e a configuração de usuários e permissões, o que pode ser feito através da interface administrativa do SCADA BR.

Após a conclusão dessas etapas, é possível testar a instalação acessando o sistema por meio de um navegador web, utilizando o endereço "http://localhost:8080"ou a URL definida durante a instalação. Será necessário fazer login com as credenciais padrão ou aquelas definidas durante o processo de instalação. Para verificar se o sistema está funcionando corretamente, pode-se criar um projeto básico e adicionar um dispositivo, testando a comunicação e as funcionalidades de supervisão.

Durante a instalação e configuração, podem surgir alguns problemas comuns. Um dos erros mais frequentes é a falha na conexão com o banco de dados, que pode ser resolvida verificando se o banco de dados está ativo e se as credenciais inseridas estão corretas. Outro problema recorrente envolve a configuração de portas de comunicação, devendo-se garantir que a porta designada não esteja sendo usada por outro aplicativo e que o firewall esteja configurado para permitir a comunicação.

Em caso de dificuldades adicionais, recomenda-se consultar a documentação oficial do SCADA BR, que contém informações detalhadas e sugestões de solução de problemas. Além disso, a participação em fóruns e comunidades online pode ser uma valiosa fonte de suporte e compartilhamento de experiências.

ANEXO C – Terceiro Anexo

C.1 Comparativo de preços entre o uso de ferramentas abertas e ferramentas pagas

Para realizar um comparativo, serão comparadas ferramentas comerciais utilizadas no mercado e ferramentas abertas mais utlizadas pela comunidade. para o comparativo, iremos comparar o software *Elipse E3* com o SCADABR, o TIA Portal da siemens com OpenPLC/programação C++ e um CLP siemens S7 com CLPs com esp32.

O Elipse E3, um sistema SCADA comercial, pode ter um custo inicial em torno de R6.000aR 12.000, dependendo do número de tags e funcionalidades. Em contraste, ferramentas SCADA abertas como *SCADABR* ou *SCADA-LTS* são gratuitas. Embora o Elipse E3 ofereça um suporte técnico robusto e funcionalidades avançadas, soluções abertas geralmente atendem a projetos industrias, com uma curva de aprendizado maior, mas a um custo substancialmente mais baixo.

Os CLPs Siemens, como o S7-1200, custam em torno de R$ 2500 a R$ 2800 de acordo com cotação interna, o que é um preço elevado em comparação a plataformas de hardware abertas como a ESP32, que podem ser adquiridos por menos de 200, tendo ainda assim, controladores industriais localizados para ESP32 com preços de R$1500 a R$2000.

Para programar um CLP, é necessário o custo com um software de programação. A siemens possui o TIA portal, software que programa e parametriza CLPs. O custo de uma licença gira em torno de R$ 10000 a R$ 15000, sendo que com ferramentas abertas, o engenheiro pode escolher entre o uso de ferramentas com a linguagem C++ ou mesmo em Ladder usando o OpenPLC.

Ainda existe outro custo, oneroso ao DNIT, que é o custo de upload do código por parte do engenheiro, que custa R$ 40000, o que custa em média 4 meses de salario de um engenheiro pleno contratado pelo DNIT para resolver o problema de automação de uma eclusa.

Tipo de Ferramenta	Ferramenta Paga	Preço estimado (R$)	Ferramenta Grautita	Preço Estimado (R$)
Programação e simulação	TIA Portal	R$10000	OpenPLC/C++	Gratuito
Supervisório	Elipse E3	R$6000	SCADABR	Gratuito
Controlador	Siemens S7-1200	R$ 2500	Placa Industrial c/ESP32	R$ 1500
Projeto+Upload	Via contrato	R$ 40000	Contrato de um engenheiro p/ usar ferrametas Gratuitas	R$ 10000 mês
TOTAL		R$ 58500		R$ 11500

Tabela 2 – Comparativo de preços e funcionalidades entre ferramentas comerciais e abertas