UNIVERSIDADE FEDERAL DO ABC
FERNANDO MORENO NHOQUI

AQUISIÇÃO E ANÁLISE DE DADOS, EM TEMPO REAL, EM SISTEMAS DE GERENCIAMENTO DE FROTAS

Santo André
2024

FERNANDO MORENO NHOQUI

AQUISIÇÃO E ANÁLISE DE DADOS, EM TEMPO REAL, EM SISTEMAS DE GERENCIAMENTO DE FROTAS

Monografia, apresentada a Universidade Federal do ABC, como parte dos requisitos necessários para obtenção do título de Especialização em Tecnologia e Sistemas de Informação. Orientado pelo Prof. Dr. Mário Gazziro.

Santo André

2024

Sistema de Bibliotecas da Universidade Federal do ABC
Elaborada pelo Sistema de Geração de Ficha Catalográfica da UFABC
com os dados fornecidos pelo(a) autor(a).

Moreno Nhoqui, Fernando
Aquisição e Análise de dados, em tempo real, em sistemas de gerenciamaneto de frotas / Fernando Moreno Nhoqui. — 2024.

33 fls. : il.

Orientação de: Mário Gazziro

Trabalho de Conclusão de Curso — Universidade Federal do ABC, Programa de Pós-Graduação em Tecnologias e Sistemas de Informação, Santo André, 2024.

1. gestão de frotas. 2. análise de dados. 3. conectividade de dispositivos. I. Gazziro, Mário. II. Programa de Pós-Graduação em Tecnologias e Sistemas de Informação, 2024. III. Título.

Dedico este trabalho a todos que me ajudaram em minha jornada durante o curso de pós-graduação de Tecnologia e Sistema de Informação.

AGRADECIMENTOS

Agradeço a Deus por ter me dado a oportunidade de ingressar no curso e me permitir conclui-lo com saúde.

A minha esposa Elayne por me incentivar a ingressar no curso e por me apoiar durante toda a sua duração, compreendendo minha ausência em datas importantes e me motivando sempre.

A minha filha Melissa que nasceu durante o curso e me trouxe ainda mais motivação para seguir em frente e me desenvolver.

Aos meus pais Orlando e Rosana, minha irmã Talita e minha avó Maria Helena por sempre me apoiarem em minhas decisões e acreditarem em meu potencial, me incentivando nos meus objetivos e sonhos.

Ao meu orientador Mário Gazziro por acreditar em meu trabalho e me orientar a elaborá-lo da melhor forma possível com suas dicas.

Aos meus colegas de trabalho, João e Victor por sempre me apoiarem no projeto e colaborarem em seu desenvolvimento.

Aos meus colegas de classe que dedicaram tempo me ajudando nos momentos em que precisei de ajuda.

RESUMO

Com os avanços da conectividade e o grande volume de dados que conseguimos coletar atualmente, através de sensores e dispositivos móveis, podemos acompanhar e gerir os sistemas em tempo real com o objetivo de trazer maior eficiência, segurança operacional, maior controle de custos e maior qualidade de atendimento ao cliente.

A utilização de algoritmos apropriados para identificar os padrões de comportamento de um sistema, traz a possibilidade de redução das manutenções corretivas e aumento de manutenções preventivas. A manutenção preditiva é uma abordagem estratégica que visa antecipar falhas e desgastes em equipamentos, evitando paradas não programadas e reduzindo custos. Quando combinada com o uso de *big data*, essa prática se torna ainda mais eficaz.

palavras-chaves: gestão de frotas, análise de dados, big data, algoritmos de predição, dashboards de gestão, conectividade de dispositivos, protocolos de comunicação.

ABSTRACT

With advances in connectivity and the large volume of data that we are currently able to collect, through sensors and mobile devices, we can monitor and manage systems in real time with the aim of bringing greater efficiency, operational security, greater cost control and greater quality of customer service.

The use of appropriate algorithms to identify the behavior patterns of a system brings the possibility of reducing corrective maintenance and increasing preventive maintenance. Predictive maintenance is a strategic approach that aims to anticipate equipment failures and wear, avoiding unscheduled downtime and reducing costs. When combined with the use of big data, this practice becomes even more effective.

Keywords: Home automation. Home automation. Home automation components. Architecture of a home automation system, Computational intelligence.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

Com o crescimento da terceirização de frotas em centro logísticos, fica cada vez mais evidente a necessidade de uma gestão em tempo real destes sistemas. Uma boa gestão de frotas é capaz de garantir o melhor desempenho, economia, segurança e produtividade.

Antigamente os sistemas eram realizados através de planilhas que recebiam informações dos próprios usuários ou de outros membros da equipe logística. Mas com a digitalização dos sistemas e crescimento dos sensores e atuadores, podemos utilizar sistemas robustos e capazes de automatizar este processo, reduzir o índice de erros e aumentar consideravelmente o tempo de resposta para uma tomada de decisão mais rápida e assertiva. Um sistema de gestão de frotas é capaz de reduzir em até 30% os custos da entrega em uma empresa.

Os sistemas de gerenciamento de frotas dentro dos centros logísticos desempenham um papel crucial na otimização e eficiência das operações de transporte. A coleta e análise de dados em tempo real são elementos essenciais para o sucesso desses sistemas. Nesta abordagem, exploraremos um sistema de aquisição contínua de informações, armazenamento de dados, análise imediata destas informações e plataformas dinâmicas que possibilitam uma gestão simples e eficaz de frotas [1].

1.2 OBJETIVO

Este trabalho tem como objetivo aquisitar dados de tensão, corrente e informações dos operadores, estabelecer uma conexão entre um broker MQTT (*Message Queuing Telemetry Transport*) e um banco de dados de séries temporais e disponibilizar *dashboards* de gestão com a utilização de uma plataforma *open source*.

Para atingir o sucesso ao final do projeto, seguiremos os seguintes objetivos:

- Investigar como a aquisição contínua de dados em tempo real e sua análise podem otimizar a gestão de frotas;

- Identificar as principais variáveis do sistema para análise;
- Avaliar o Mosquitto como broker MQTT e suas regras para envio de dados;
- Avaliar o Prometheus como banco de dados para armazenar informações relevantes;
- Criar *dashboards* eficientes no Grafana para monitorar e tomar decisões informadas;

1.3 ESTRUTURA DO TEXTO

Nesse primeiro capítulo é apresentada a justificativa da escolha do projeto, sustentada pela perspectiva do mercado e do desenvolvimento das áreas abordadas. Além disso, são definidos todos os objetivos a serem realizados no trabalho.

No segundo capítulo é apresentado as tecnologias utilizadas para o desenvolvimento do projeto. O terceiro capítulo especifica as metodologias para o desenvolvimento, assim como as etapas de sua implementação. O quarto capítulo descreve os resultados obtidos com o projeto. O quinto capítulo apresenta as conclusões obtidas nesse trabalho.

2 TECNOLOGIAS RELACIONADAS

2.1 PROTOCOLO DE COMUNICAÇÃO

Os protocolos de comunicação são fundamentais para o bom funcionamento dos sistemas baseados em IoT (*Internet of Things*). Saber escolher o protocolo que mais se adequa ao seu projeto é um passo importante para a sequência do seu desenvolvimento.

Protocolos de comunicação têm como função estabelecer uma integração entre dispositivos e sistemas, permitindo troca de dados entre eles independente de seus fabricantes e desenvolvedores [2].

2.2 BANCO DE DADOS

Os sistemas de IoT têm por característica um grande volume de aquisição de dados, onde todos os dispositivos conectados ao broker publicam dados que são gerenciados e enviados para outras ferramentas para tratamento. Nesta hora entra em ação os sistemas de gerenciamento de bancos de dados.

Sistema de gerenciamento de banco de dados (SGBD) é uma coleção de dados inter-relacionados a um conjunto de programas para acessar esses dados. Os bancos de dados são projetados para armazenar grandes volumes de dados, possuem ferramentas e mecanismos para uma manipulação fácil e segura destes dados [3].

Rodhen V. [4] mostra que existem diversos tipos de banco de dados atualmente, conforme listado abaixo.

Bancos de dados relacionais: se tornaram dominantes na década de 1980. Os itens em um banco de dados relacional são organizados como um conjunto de tabelas com colunas e linhas. A tecnologia de banco de dados relacional fornece a maneira mais eficiente e flexível de acessar informações estruturadas [4].

Bancos de dados orientados a objetos: nele, as informações são representadas na forma de objetos, como na programação orientada a objetos [4].

Bancos de dados distribuídos: consiste em dois ou mais arquivos localizados em sites diferentes. O banco de dados pode ser armazenado em vários computadores, localizados no mesmo local físico ou espalhados por diferentes redes [4].

Data warehouses: um tipo de banco de dados projetado especificamente para consultas e análises rápidas, como por exemplo, um repositório central de dados, um *data warehouse* [4].

Bancos de Dados NoSQL: ou banco de dados não relacional, permite que dados não estruturados e semiestruturados sejam armazenados e manipulados (em contraste com um banco de dados relacional, que define como todos os dados inseridos no banco de dados devem ser compostos). Os bancos de dados NoSQL se tornaram populares à medida que os aplicativos web se tornaram mais comuns e mais complexos [4].

Bancos de dados gráficos: armazena dados em termos de entidades e os relacionamentos entre entidades [4].

Bancos de dados OLTP: é um banco de dados rápido e analítico projetado para um grande número de transações realizadas por vários usuários [4].

Bancos de séries temporais (TSDB): é um banco otimizado para lidar com dados indexados por data/hora ou séries temporais, são projetados para lidar com grandes volumes de dados, possuem recursos de compactação, indexação e agregação, o que lhes permite consultas e análises rápidas [5].

2.3 DASHBOARDS E ANÁLISE DE DADOS

Dashboards são painéis que exibem os dados do sistema de forma organizada utilizando diversos elementos gráficos de acordo com o tipo de informação que será gerenciada. Os *dashboards* têm como objetivo transformar dados brutos em informações visuais, facilitando assim a compreensão dos resultados obtidos com a coleta e processamento de dados, são ferramentas que apoiam a tomada de decisão em um processo [6].

2.4 SISTEMAS EMBARCADOS

Nos últimos anos, houve uma rápida evolução na eletrônica. Nesse contexto, os sistemas computacionais embarcados surgem cada vez mais compactos, dinâmicos e acessíveis, adaptando-se às mais diversas aplicações. Com isso os sistemas embarcados ganharam importância e aparecem cada vez mais em nosso dia a dia. Segundo Chase (2007), um sistema é considerado embarcado quando é dedicado a uma tarefa específica e interage continuamente com o ambiente por meio de sensores e atuadores. Esse tipo de sistema exige do projetista conhecimentos em programação, sistemas digitais, controle de processos, sistemas em tempo real, e tecnologias de aquisição de dados e atuadores. O termo "embarcado" refere-se à sua capacidade de operar de forma independente de fontes de energia fixas. As principais características incluem a capacidade computacional e a autonomia operacional, com variações dependendo do tipo e aplicação do sistema. [7].

2.5 LINGUAGEM DE PROGRAMAÇÃO C++

A linguagem C++ é uma extensão da linguagem C, que por sua vez tem suas raízes na linguagem B, desenvolvida por Ken Thompson. A linguagem C foi criada nos laboratórios Bell da *American Telephone and Telegraph* (AT&T) entre 1969 e 1973. Bjarne Stroustrup, frequentemente conhecido como o pai do C++, apresentou ao mundo essa linguagem avançada, projetada para lidar com problemas complexos através de uma ampla gama de recursos. Como Stroustrup afirmou, "C++ é complexa porque é poderosa.

3 METODOLOGIA E IMPLEMENTAÇÃO

3.1 VISÃO GERAL

O projeto de telemetria visa fornecer dados operacionais das máquinas em tempo real, auxiliando no monitoramento e manutenção das máquinas. Esse projeto é bem estruturado e aproveita tecnologias modernas para monitoramento e análise de dados. A seguir, um detalhamento da arquitetura do sistema e definição de como cada componente contribui para o sistema:

3.2 ARQUITETURA DO PROJETO DE TELEMETRIA

- Máquinas e Sensores: Equipamentos distribuídos por todo o país estão equipados com microcontroladores ESP32. Esses microcontroladores monitoram diversos dados operacionais das máquinas, como corrente, tensão e outros parâmetros importantes. Com base nas leituras de tensão e corrente, o sistema realiza um controle utilizando horímetros, que ajudam a gerenciar e otimizar o plano de manutenção dos equipamentos.

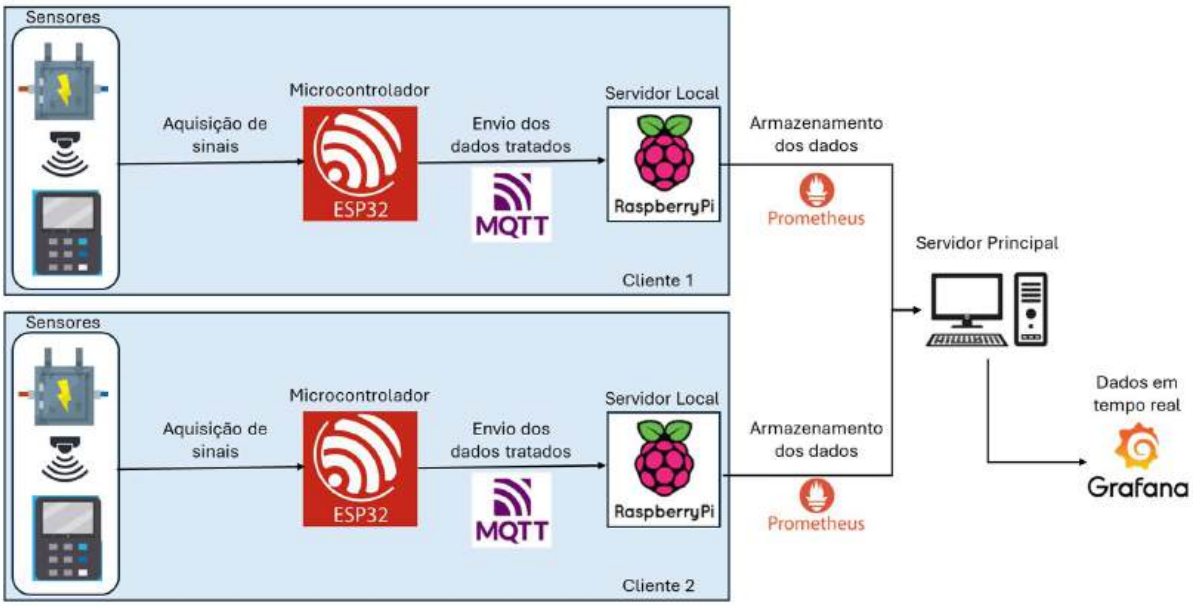
- Comunicação Local: Os dados coletados pelos microcontroladores ESP32 são transmitidos para servidores locais, que são compostos por microcomputadores *Raspberry Pi*. A comunicação entre os microcontroladores e os *Raspberry Pi* é realizada através do protocolo MQTT (*Message Queuing Telemetry Transport*). O MQTT é uma solução eficiente para comunicação em tempo real e é especialmente adequado para redes com largura de banda limitada ou instável.

- Servidores Locais (*Raspberry Pi*): Cada servidor local *Raspberry Pi* recebe e centraliza os dados enviados pelos microcontroladores ESP32. O *Raspberry Pi* pode também realizar um pré-processamento ou filtragem básica dos dados antes de enviá-los para o servidor principal.

- Servidor Principal: Recebe dados de todos os servidores locais. Os dados são armazenados no serviço Prometheus. O Prometheus é um sistema de monitoramento e alerta que armazena dados em um formato de séries temporais, o que é ideal para dados operacionais contínuos.

- Visualização e Monitoramento: O Prometheus expõe os dados armazenados para visualização. O Grafana é usado para criar *dashboards* interativos e personalizáveis, permitindo a visualização e análise dos dados em tempo real através de um navegador *web*.

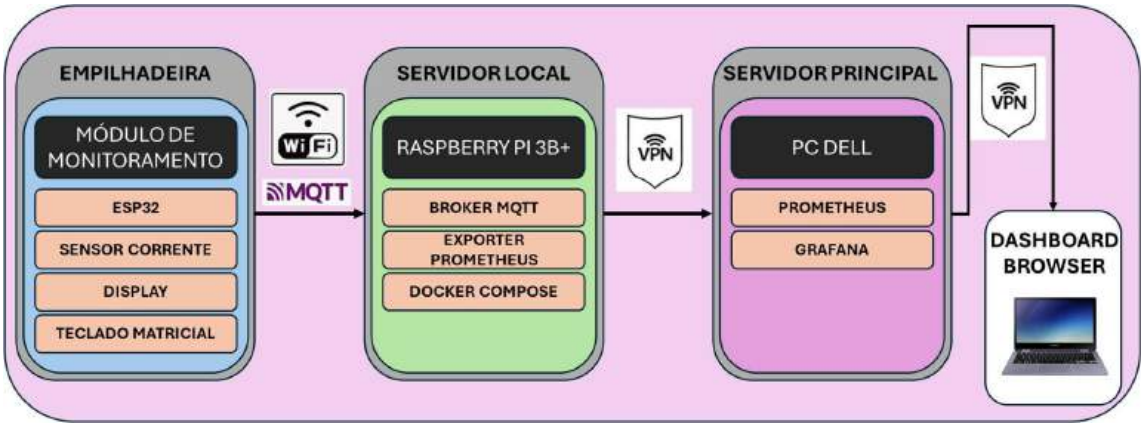
Figura 1 - Arquitetura do sistema.



Fonte: Autoria própria

O conjunto de blocos do sistema que constitui a construção tecnológica dimensionada para o projeto, desde a medição dos sinais e aquisição de informações nas empilhadeiras até o *dashboard* final do usuário, está representado abaixo na Figura 2.

Figura 2 - Diagrama de blocos da "Construção tecnológica da telemetria".



Fonte: Autoria própria

3.2.1 Microcontrolador ESP-32

A ESP32 é uma placa de desenvolvimento de *hardware* aberto que é baseada em um processador *dual-core* de 32 bits e possui 520 KB de memória *flash*. Possui *Wi-Fi* e *Bluetooth* integrados, o que elimina a necessidade de comprar módulos separados. O ESP-32 foi escolhido para o projeto, devido a essa facilidade de integração dos módulos.

Figura 3 - Exemplo do microcontrolador ESP32

Fonte: SmartProjects

3.2.2 Servidor Raspberry Pi 3 B+

O *Raspberry Pi* (ou RPi) é um microcomputador completo integrado em uma única placa. Ele inclui processador, memória RAM e gráficos embutidos, além de portas USB, HDMI, áudio, vídeo composto, e conexões para câmeras e telas LCD. A placa também possui uma interface GPIO com pinos de entrada/saída versáteis. A alimentação é fornecida através de uma porta microUSB, permitindo o uso de fontes de energia comuns para telefones celulares. Para o nosso sistema, foi escolhido o modelo *Raspberry Pi* 3B+, que funciona como o servidor local e opera com o sistema operacional *Debian*.

Figura 4 - Exemplo do microcomputador Easpberry Pi 3B+

Fonte: Amazon

3.2.3 Sensor de Corrente/Tensão

As medições de corrente (A) e tensão (V) das máquinas a serem monitoradas serão realizadas utilizando um resistor de *shunt*, também conhecido como derivador resistivo. Este componente permite a medição precisa da corrente elétrica ao criar uma pequena queda de tensão proporcional à corrente que passa através dele.

3.2.4 Sistema de Preenchimento de Checklist

O *checklist* é um conjunto de perguntas destinadas ao operador da empilhadeira, que inicia o processo realizando sua identificação por meio de um cartão RFID de 815 MHz. O operador responde a perguntas sobre o estado da empilhadeira antes do início de seu turno, garantindo que o equipamento esteja em condições adequadas para uso.

3.2.5 Prometheus - Armazenamento de Dados

Devido ao alto volume de dados e necessidade de velocidade nas consultas para análise, o projeto utilizará o Prometheus, um banco de dados de séries temporais.

O Prometheus foi desenvolvido no *SoundCloud* em 2012, possui código aberto e uma comunidade de desenvolvedores muito ativa. Em 2016 o Prometheu juntou-se à *Cloud Native Computing Foundation*. O Prometheus extrai métricas por meio de *push*, via HTTP (*HiperText Transfer Protocol),* armazena todas as amostras localmente e executa regras sobre os dados para registrar novas séries temporais ou gerar alertas [8].

3.2.6 Grafana - Gestão Visual

Para a construção dos painéis de visualização das informações, o projeto utilizará a ferramenta Grafana. O Grafana é um *software* aberto que permite consultar, alertar e explorar métricas, logs e rastreamentos de forma simples e prática. A ferramenta permite transformar dados de banco de dados de série temporal em gráficos e *dashboards* de visualização. Sua estrutura possui integração direta com o Prometheus [9].

Figura 5 – Exemplo de gráfico no Grafana.

Fonte: Wikipédia

3.2.7 Docker Compose

Docker compose é uma ferramenta para definição e execução de múltiplos *containers* Docker. Com ela é possível configurar todos os parâmetros necessários para executar cada container a partir de um arquivo de definição. o *docker compose* será utilizado para realizar a exportação dos sinais recebidos pelo *broker* MQTT para o Prometeus. Escolher *Docker Compose* para o seu projeto simplifica a definição, gerenciamento e orquestração de múltiplos contêineres, garantindo um ambiente de desenvolvimento consistente e facilmente escalável

3.2.8 MQTT - Protocolo de comunicação

Com a popularização dos sistemas de IoT, surgiram diversos protocolos de comunicação capazes der atender projetos desta natureza, para o desenvolvimento de nosso sistema de telemetria para gestão de frotas escolhemos utilizar o protocolo MQTT (*Message Queue Telemetry Transport*), por ser um protocolo sofisticado, com fácil implantação e alta segurança.

O MQTT foi criado na década de 90 pela IBM (*International Business Machines*) com foco em sistemas supervisórios e de aquisição de dados. É um protocolo que possui um alto desempenho em hardwares simples, pois possui um *payload* pequeno devido sua pouca utilização de banda de rede. No quesito de segurança, o MQTT apresenta mais níveis de serviço e menor complexidade, além de permitir comunicação de 1 para N [10].

O protocolo MQTT utiliza um padrão de troca de mensagens *publish/subscriber* (publicador/subscritor) e utiliza tópicos para identificação das mensagens. O sistema de tópicos lembra o conceito de URI com níveis separados por "/" (O PROTOCOLO). Neste padrão, as informações são enviadas pelo dispositivo ao *broker* através de uma publicação, onde o dispositivo subscreve no *broker*, podendo enviar diversos tópicos a ele [11].

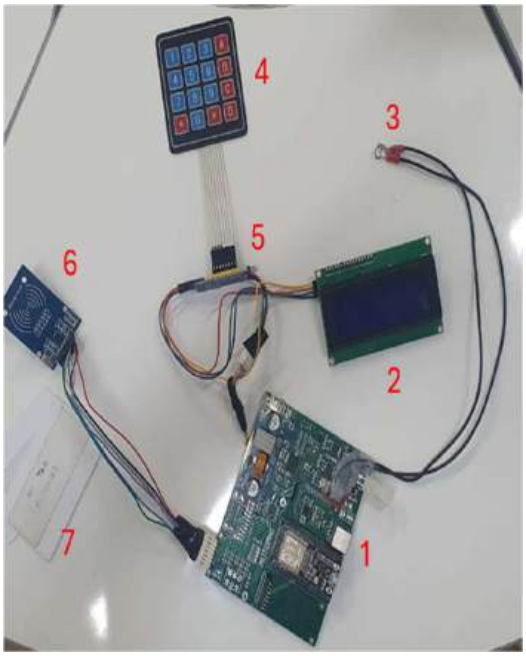
O *broker* funciona como um servidor, responsável por centralizar as comunicações com os dispositivos e gerir todas as publicações e subscrições, ele permite desacoplamento entre as partes comunicantes, diferentemente de outros sistemas cliente/servidor. É possível trabalhar com *brokers* redundantes para tornar o sistema mais robusto [11].

Neste projeto iremos utilizar um *Raspberry Pi*, modelo 3B+ para ser o *broker*, rodando em um sistema operacional derivado do Linux, o *Raspian*.

3.3 ELETRÔNICA EMBARCADA

A Figura 6 e a Tabela I apresentam respectivamente os componentes da eletrônica embarcada utilizada no projeto.

Figura 6 - Componentes da eletrônica embarcada.



Fonte: Autoria própria

Tabela 1 – Relação dos componentes.

Item	Componente	Quantidade
1	Circuito + ESP32 + INA3221	1
2	Display LCD 20x4 com Blacklight Azul	1
3	Conectores para leitura de sinais do shunt	2
4	Teclado Matricial - Membrana 4x4	1
5	Módulo de Expansão I2C 8-bit PCF8574	2
6	Leitor RFID RC522 - 13,56MHz	1
7	Cartão RFID - 13,56MHz	2

Fonte: autoria própria.

3.3.1 Display LCD 20x4 com Backlight Azul

Utilizaremos um *display* 20x4 com luz azul para realizar interface visual com o operador. O *display* auxiliará o operador na visualização dos parâmetros do dispositivo e preenchimento do *checklist.*

3.3.2 Keypad Matricial 4x4

O teclado matricial permitirá que o operador e o mantenedor naveguem pelo sistema operacional do dispositivo de forma intuitiva e eficiente. O teclado matricial oferece vantagens como uma interface compacta, baixo custo, alta durabilidade e simplicidade na implementação para navegação e entrada de dados.

3.3.3 PCF8574

O Circuito integrado PCF8574 é um expansor de 8 bits bidirecionais (8 pinos I/O) que são comandados por I2C (barramento serial). Ele foi adaptado para controlar LCDs de vários tamanhos como 16×2, 16×4, 20×2 ou 20×4 por meio de barramento serial I2C. Dessa forma, com apenas 4 fios (+5v, SDA, SCL e terra) podemos controlar um LCD e o *Keypad*.

3.3.4 INA226

O módulo INA226 é um monitor de tensão e corrente de 3 canais e interface digital I2C. O módulo INA226 será responsável por realizar as leituras de corrente e tensão das máquinas através dos sinais enviados pelo resistor de *shunt*.

3.3.5 Resistor de shunt

Os resistores de *shunt* ou derivadores resistivos são utilizados para a medição de correntes de amplitudes geralmente elevadas. São na verdade resistores com baixo coeficiente térmico que ao serem percorridos por uma corrente elétrica (im) geram sobre si uma tensão elétrica (vm). O resistor de shunt será utilizado para medição da corrente e tensão geral do

equipamento que será medido. A medição será realizada pelo microcontrolador ESP 32 através do barramento de comunicação I2C com a utilização do circuito INA266.

Figura 7 - Resistor de shunt em invólucro de proteção.

Fonte: Autoria própria.

4 RESULTADOS

O desenvolvimento do sistema de telemetria para empilhadeiras atingiu com sucesso seus objetivos, proporcionando leituras precisas dos equipamentos e aproveitando a arquitetura projetada para garantir a entrega dos dados em tempo real. A seguir, serão apresentados os resultados obtidos, detalhando a construção e o desempenho de cada componente desenvolvido, bem como suas respectivas respostas.

4.1 PAINEL DO OPERADOR (PO)

Um painel do operador foi desenvolvido com o objetivo de centralizar o microcontrolador e os módulos de comunicação do sistema. Essa abordagem resultou em um sistema compacto, que protege a eletrônica embarcada e proporciona uma interface de usuário mais agradável e prática.

Figura 8 - Painel do operador.

Fonte: autoria própria

O sistema do painel do operador é responsável por medir as leituras de corrente e tensão da empilhadeira, coletar as respostas do *checklist* preenchido pelo operador e identificar o usuário que está utilizando o equipamento.

Todas essas funções foram testadas e apresentaram resultados positivos. A seguir, são apresentados os resultados obtidos em cada teste das funcionalidades do painel do operador.

4.1.1 Leituras de tensão e corrente

O sensor de corrente e tensão utilizado no projeto foi o resistor de *shunt*, a escolha deste tipo de medição se deu por conta da precisão obtida em sua leitura, como também pela facilidade de instalação que foi construída, criando um conceito *plug-and-play* para o sistema.

Para realizar essas leituras, foi necessária a comunicação entre o microcontrolador e o módulo INA226 por meio do barramento de comunicação I2C.

Figura 9 - Configuração dO INA226 e parametrização de resistor shunt.

```
// ------------------------------------------------------------------
// -----INA setup-----
void ina226_setup(){
  INA.init();
  INA.setAverage(AVERAGE_4);
  INA.setConversionTime(CONV_TIME_1100);
  INA.setResistorRange(SHUNT_RESISTENCE, 300.0);
  INA.waitUntilConversionCompleted();
}
```

Fonte: autoria própria

O circuito INA226, em conjunto com o resistor de *shunt*, funcionou conforme o esperado ao utilizar o barramento de comunicação I2C. Ele realizou com precisão a leitura de corrente e tensão transmitidas pelo resistor de shunt. As leituras foram aferidas com a utilização de um multímetro digital *Fluke*, modelo 17B.

Figura 10 - Leituras realizadas durante teste aplicado em transpaleteira.

Fonte: autoria própria.

4.1.2 Configuração display + teclado

A adição do *display* LCD e do teclado foi feita para permitir a interface do usuário com o sistema. Dessa forma, o usuário pode realizar o login no sistema e preencher o *checklist* com os detalhes da máquina antes do início de sua operação.

Para possibilitar a comunicação entre o microcontrolador, o teclado e o display, foi necessário utilizar o módulo PCF8574 via barramento de comunicação I2C.

Figura 11 - Configuração de comunicação do teclado e display no barramento I2C.

```
40  // ---defines---
41  #define SHUNT_RESISTENCE  0.75
42  char keypad[19] = "123A456B789C*0#DNF";
43
44  // ------------------------------------------------------------
45  //----I²C Adresses------
46  #define LCDADRESS 0x27
47  #define INAADRESS 0x40
48  #define KeyAdress 0x21
49
50  // ------------------------------------------------------------
51  //----pins------
52  #define   SS        22
53  #define   RST       5
54  #define   SCK       14
55  #define   MISO      2
56  #define   MOSI      15
57  #define   SDA       32
58  #define   SCL       33
```

Fonte: autoria própria

Os circuitos PCF8574 funcionaram conforme o esperado ao utilizar o barramento de comunicação I2C, gerenciando corretamente a troca de sinais com os dispositivos conectados, como o display e o teclado.

4.1.3 Configuração das telas de checklist

O *checklist* tem o objetivo de estabelecer uma conexão entre o usuário e o sistema, exigindo que o operador de empilhadeira responda a questões relacionadas ao estado da máquina antes do início da operação. O preenchimento do *checklist* é crucial tanto para o operador quanto para o sistema, pois registra a identidade do usuário e o estado da máquina no momento do início da operação. Caso seja detectada alguma avaria no equipamento, o sistema registrará essa informação.

Figura 12 - Navegação do checklist em andamento.

Fonte: autoria própria

Todas as telas do *checklist* funcionaram corretamente, enviando as respostas através da arquitetura do sistema até o *dashboard* de gestão.

Figura 13 - Trecho do código da tela apresentada na figura 11,

```
1123    void bateria(){
1135
1136        if (key != 'N') {
1137          vTaskDelay(50);
1138          if (key == '1') {
1139            client.publish(topic_B, "True");
1140            lcd.clear();
1141            lcd.setCursor(6, 1);
1142            lcd.print("CONCLUIDO");
1143            lcd.setCursor(2, 2);
1144            lcd.print("MAQUINA LIBERADA");
1145            vTaskDelay(1000);
1146            opnav = true;
1147            telafinal();
1148          } else if (key == '2') {
1149            client.publish(topic_B, "False");
1150            lcd.clear();
1151            lcd.setCursor(6, 1);
1152            lcd.print("CONCLUIDO");
1153            lcd.setCursor(2, 2);
1154            lcd.print("MAQUINA LIBERADA");
1155            vTaskDelay(1000);
1156            opnav = true;
1157            telafinal();
1158          }
1159        }
```

Fonte: autoria própria

4.1.4 Leitor RFID

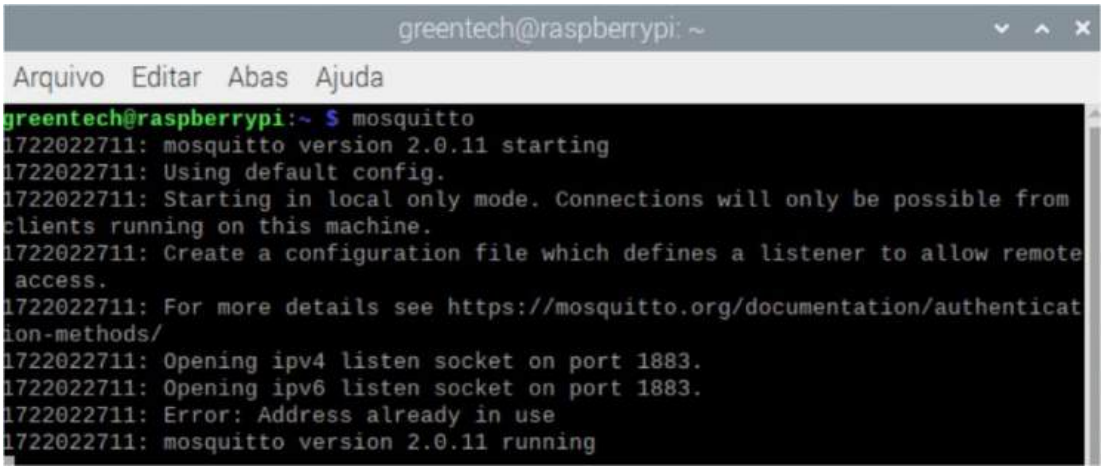
O leitor RFID foi selecionado para identificar o usuário por meio da leitura de cartões. Cada operador terá seu próprio cartão e deverá realizar o login no sistema antes de iniciar a utilização da empilhadeira. O módulo leitor de RFID funcionou conforme o esperado, utilizando o barramento de comunicação SPI, e realizou com precisão a leitura dos cartões RFID operando na frequência de 815 MHz para identificar o operador.

4.2 SERVIDORES

4.2.1 Servidor local

O *Raspberry Pi* foi configurado com sistema operacional *Debian*, Mosquitto *Broker* MQTT, *Docker Compose* e *Prometheus Exporter*.

Figura 14 - Tela de status do broker Mosquitto em funcionamento.



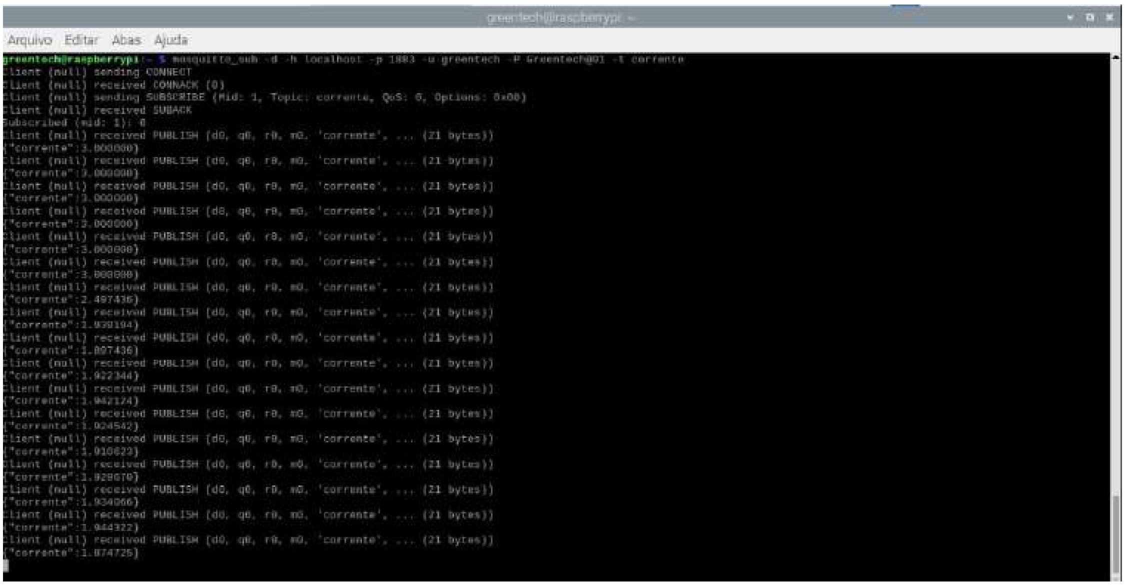
Fonte: autoria própria

Figura 15 - Configuração do broker MQTT.

Fonte: autoria própria

A conexão Wi-Fi entre o ESP32 e o Raspberry Pi funcionou, porém demonstrou uma limitação de cobertura do sinal. O envio de mensagens entre o ESP32 e o *Raspberry Pi* via MQTT ocorreu com sucesso e atendeu às expectativas.

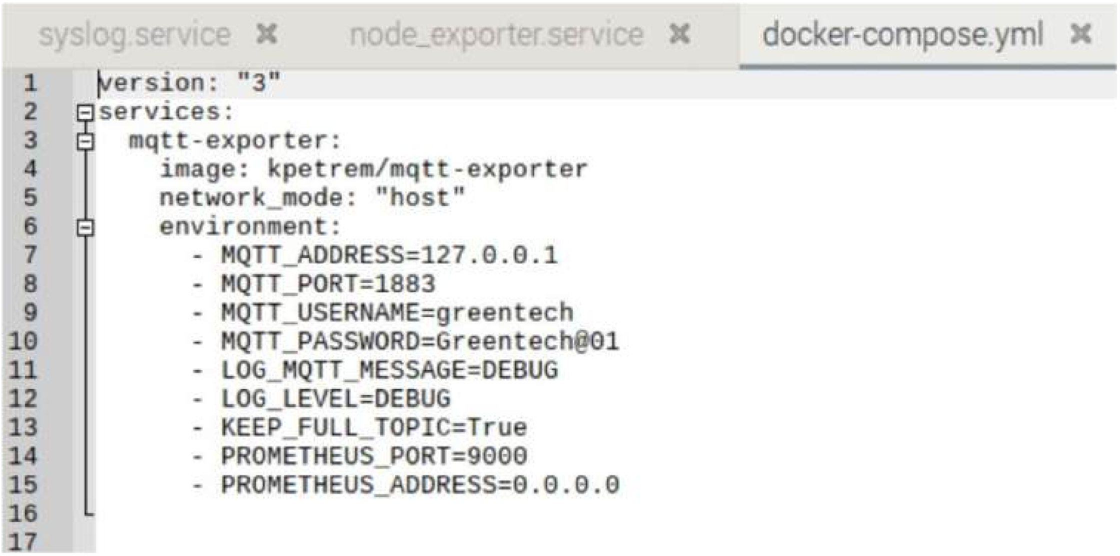
Figura 16 - Leituras de corrente representadas no broker durante testes.



Fonte: autoria própria

O *Docker Compose* e o *Prometheus Exporter* funcionaram corretamente e conseguiram enviar as mensagens do *Broker* para Prometeus, instalado no servidor principal.

Figura 17 - Parametrização do docker compose.

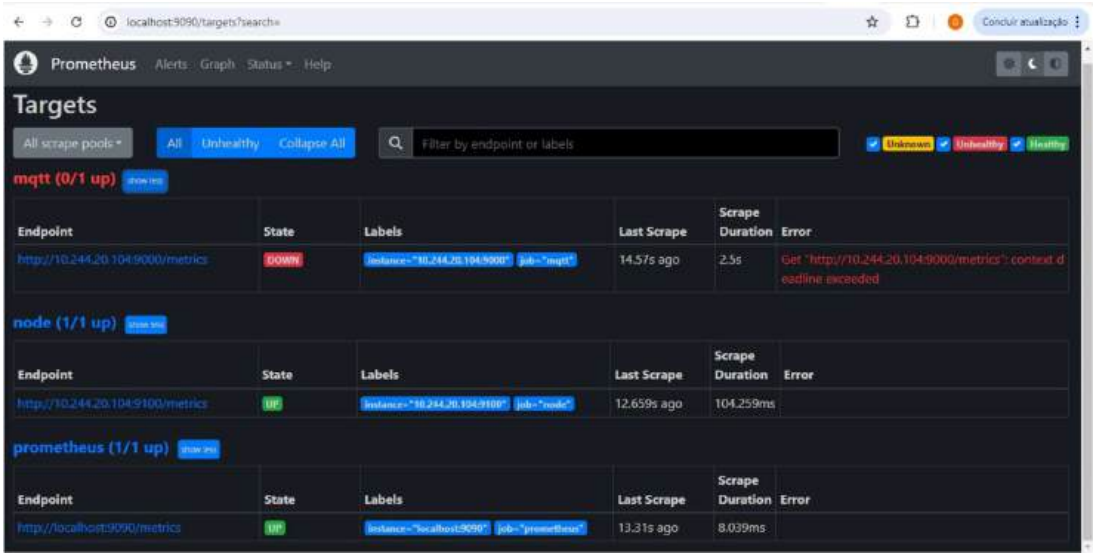


Fonte: autoria própria

4.2.2 Servidor principal

No servidor principal, todas as informações foram recebidas corretamente. O serviço Prometheus funcionou como esperado e se mostrou eficiente para a aplicação.

Figura 18 - Serviço do Prometheus rodando e link de conexão com o servidor local.



Fonte: autoria própria

A integração com o sistema de *dashboards* do Grafana ocorreu conforme projetado. Com todos os elementos devidamente configurados e conectados, foi possível visualizar as variáveis de operação das empilhadeiras em tempo real nos *dashboards* do Grafana.

Figura 19 - Dashboard no Grafana durante os testes e simulações.

Fonte: autoria própria

5 CONCLUSÃO

Neste trabalho, foi possível explorar e compreender a aplicação de tecnologias e métodos atuais em sistemas de telemetria, desde o *hardware* disponível no mercado até as configurações gráficas que permitem ao usuário monitorar os do sistema. O uso do ESP32, *Raspberry Pi*, Prometheus e Grafana demonstrou ser eficaz para a aplicação proposta, proporcionando uma solução de baixo custo e fácil integração.

Embora o sistema desenvolvido atenda de maneira satisfatória aplicações em pequenos ambientes, a sua aplicabilidade em grandes galpões e centros de distribuição apresenta desafios relacionados à cobertura e à estabilidade da conexão WiFi.

Para futuros trabalhos, pode-se aprimorar o projeto e torná-lo mais adequado a ambientes de grande escala com a adoção de tecnologias como LoRa ou WiFi HaLow. Essas tecnologias podem fornecer uma comunicação mais robusta e de longo alcance, garantindo uma cobertura eficiente e confiável.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] TOTVS – Gestão de Frotas – Disponível em https://www.totvs.com/blog/gestao-para-rotas/gestao-de-frotas/ - Acesso em 05 de abril de 2024.

[2] ALCTEL - O que é o sistema IOT e como funciona na prática? - Disponível em: https://www.alctel.com.br/o-que-e-protocolo-iot-e-como-funciona-na-pratica/ - Acesso em 26 de abril de 2024.

[3] SILVERSCHATS A., KORTH H., SUDARSHAN S., Sistema de banco de dados. 7ª edição. Editora: GEN LTC, 21 de agosto de 2020.

[4] RODHEN V., BANCO DE DADOS: Características e Importância para o BI e Segurança da Informação., 2022. Disponível em: https://timr.com.br/banco-de-dados-caracteristicas-e-importancia-para-o-bi-e-seguranca-da-informacao/. Acesso em: 27 de abril de 2024.

[5] SPASOJEVIC A., - O que é um banco de dados de mséries temporais?, 2023. DIsponível em: https://www.phoenixnap.pt/gloss%C3%A1rio/o-que-%C3%A9-um-banco-de-dados-de-s%C3%A9ries-temporais. Acesso em: 27/04/2024.

[6] NITEO LEARNING, 2024 - Dashboards do zero: Aprenda as melhores práticas para visualizar dados de negócios w manter um time analítico, 2024. Disponível em: https://niteolearning.com/blog/o-que-e-dashboard-tipos-como-funciona/. Acesso em: 28 de abril de 2024.

[7] CHASE O., SISTEMAS EMBARCADOS 2007 - Disponível em: https://www.maxpezzin.com.br/aulas/6_EAC_Sistemas_Embarcados/1_SE_Introducao.pdf. Acesso em: 25/07/2024.

[8] PROMETHEU, - OVERVIEW, 2024. Disponível em: https://prometheus.io/docs/introduction/overview/. Acesso em 27/04/2024.

[9] GRAFANA LABS, 2024 - About Grafana. Disponível em: https://grafana.com/docs/grafana/latest/introduction/. Acessado em 28 de abril de 2024.

[10] BARROS, M. MQTT — Protocolos para IoT. In: EMBARCADOS. , 2015. Disponível em: https://www.embarcados.com.br/mqtt-protocolos-para-iot/. Acesso em 25 de abril de 2024.

[11] O PROTOCOLO MQTT: leve e simples: perfeito para IoT e sistemas embarcados., 2018. Disponível em: https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/ trabalhos-vf/mqtt/. Acesso em: 25 de abril de 2024.

Assinatura do Orientador