UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

JAYLTON ALENCAR PEREIRA

COMUNICAÇÃO BIDIRECIONAL ENTRE CLP E APLICATIVO MÓVEL PARA AUTOMAÇÃO DE PROCESSOS INDUSTRIAIS

Recife
2023

JAYLTON ALENCAR PEREIRA

COMUNICAÇÃO BIDIRECIONAL ENTRE CLP E APLICATIVO MÓVEL PARA AUTOMAÇÃO DE PROCESSOS INDUSTRIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Douglas Contente Pimentel Barbosa

Recife
2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Pereira, Jaylton Alencar.
Comunicação bidirecional entre CLP e aplicativo móvel para automação de processos industriais / Jaylton Alencar Pereira. - Recife, 2023.
61 p. : il.

Orientador(a): Douglas Contente Pimentel Barbosa
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia de Controle e Automação - Bacharelado, 2023.
Inclui referências, apêndices.

1. CLP. 2. Comunicação. 3. Dispositivo móvel. 4. Industria 4.0. 5. Protocolo S7. I. Barbosa, Douglas Contente Pimentel. (Orientação). II. Título.

620 CDD (22.ed.)

JAYLTON ALENCAR PEREIRA

COMUNICAÇÃO BIDIRECIONAL ENTRE CLP E APLICATIVO MÓVEL PARA AUTOMAÇÃO DE PROCESSOS INDUSTRIAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Aprovado em: 03/05/2023.

BANCA EXAMINADORA

_____

Prof. Dr. Douglas Contente Pimentel Barbosa (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Herbert Albérico de Sá Leitão (Examinador Interno)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Primeiramente, quero agradecer à minha família por sempre me incentivar a estudar e seguir meus objetivos. Sem o apoio de vocês, não teria chegado tão longe. Agradeço a paciência e compreensão nos momentos em que precisei me dedicar ao trabalho acadêmico.

Também gostaria de agradecer aos meus colegas de classe, que estiveram comigo durante todo o processo, compartilhando conhecimento, experiências e muitas risadas. Vocês tornaram essa jornada ainda mais especial.

Por fim, quero agradecer aos meus professores, que me guiaram e me incentivaram a buscar o melhor de mim. O conhecimento que adquiri com vocês foi fundamental para a elaboração deste trabalho e, certamente, será de grande valor em minha carreira. A todos vocês, meu muito obrigado. Este trabalho não seria possível sem a ajuda e o apoio de cada um de vocês.

RESUMO

A Indústria 4.0 tem trazido a necessidade de monitorar e aperfeiçoar os processos industriais por meio de diversas técnicas e ferramentas que permitem a comunicação das fábricas com o meio externo pela internet. Tendo isso em vista, este trabalho tem como objetivo realizar a comunicação de um Controlador Lógico Programável (CLP) com um dispositivo móvel. A ideia é criar uma interface para que o usuário possa enviar comandos e receber informações em tempo real do CLP, tornando o processo de monitoramento e controle mais eficiente e acessível. Através do uso do protocolo S7, foi possível estabelecer uma conexão entre os dispositivos e enviar e receber informações em tempo real. Dessa forma, este trabalho demonstra a viabilidade e a importância da comunicação entre um CLP e um aplicativo móvel para o monitoramento e controle de processos industriais, oferecendo uma solução tecnológica que pode otimizar e simplificar os processos de produção.

Palavras-chave: CLP; Comunicação; Dispositivo móvel; Industria 4.0; Protocolo S7.

ABSTRACT

Industry 4.0 has brought about the need to monitor and improve industrial processes through various techniques and tools that enable factories to communicate with the external world over the internet. This work aims to establish communication between a Programmable Logic Controller (PLC) and a mobile device. The idea is to create an interface that allows the user to send commands and receive real-time information from the PLC, making the monitoring and control process more efficient and accessible. Using the S7 protocol, it was possible to establish a connection between the devices and send and receive real-time information. Thus, this work demonstrates the feasibility and importance of communication between a PLC and a mobile application for monitoring and controlling industrial processes, offering a technological solution that can optimize and simplify production processes.

Keywords: PLC; Communication; Mobile device; Industry 4.0; S7 protocol.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
AWP	*Automation Web Programming*
CLP	Controlador Lógico Programável
CSS	*Cascading Style Sheets*
CPS	*Cyber-Physical System*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IA	Inteligência Artificial
IEC	*Internacional Electrotecnical Commission*
IHM	Interface Homem Máquina
IIoT	*Industrial Internet of Things*
IoT	*Internet of Things*
IP	*Internet Protocol*
JSON	*JavaScript Object Notation*
NPM	*Node Package Manager*
OPC	*Open Platform Communications*
OSI	*Open System Interconnection*
SSL	*Secure Sockets Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Com o crescente avanço tecnológico em diversos setores, a indústria tem se tornado cada vez mais conectada, surgindo a chamada Indústria 4.0, o que tem imposto a necessidade de monitorar e aperfeiçoar os processos industriais. A Indústria 4.0 trouxe diversas inovações para o setor industrial, sendo uma delas a Internet das Coisas Industrial (IIoT). Essa tecnologia possibilita a criação de um ambiente altamente conectado, em que dispositivos e sistemas são integrados para otimizar os processos produtivos. Nesse contexto, a telemetria também desempenha um papel importante, permitindo o monitoramento remoto de equipamentos e a coleta de dados em tempo real.

Nesse sentido, a conexão entre o Controlador Lógico Programável (CLP) e um aplicativo para celular pode trazer inúmeros benefícios para o setor industrial, proporcionando maior controle e eficiência nos processos produtivos. Este trabalho tem como objetivo apresentar uma solução para essa conexão, utilizando as tecnologias Node.js, Firebase e Ionic para realizar a comunicação entre o CLP e o aplicativo móvel. Com isso, espera-se contribuir para a aplicação prática da IIoT na indústria, proporcionando maior competitividade e eficiência às empresas.

1.1 Objetivo geral

Desenvolver uma aplicação que possibilite a comunicação entre um controlador lógico programável e um aplicativo para celular. E contribuir para a implementação da indústria 4.0 em empresas que possuem processos industriais.

1.2 Objetivos específicos

Os objetivos específicos deste trabalho consistem em:

- Realizar um estudo sobre a Indústria 4.0, IIoT e telemetria.
- Estudar as tecnologias envolvidas na comunicação entre um CLP e um aplicativo para celular.

- Realizar a comunicação entre o CLP e um computador que esteja na mesma rede.
- Armazenar as informações obtidas do CLP em um servidor remoto.
- Desenvolver um aplicativo para celular que permita visualizar e alteração os dados do CLP.
- Realizar testes e comparar a solução desenvolvida com outras propostas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais para o entendimento do desenvolvimento de uma ferramenta de comunicação entre um CLP e um aplicativo para celular. Serão abordados conceitos da Indústria 4.0, Internet Industrial das Coisas (IIoT) e telemetria, e a importância da conexão entre o CLP e o aplicativo móvel. Além disso, serão apresentadas as principais tecnologias utilizadas no desenvolvimento deste trabalho, tais como o Node.js, Firebase e Ionic. O objetivo deste capítulo é fornecer uma base teórica sólida para o desenvolvimento e entendimento da ferramenta proposta.

2.1 A indústria 4.0

A Indústria 4.0, também conhecida como a Quarta Revolução Industrial, refere-se à tendência atual de automação e troca de dados na manufatura e em outras indústrias. É caracterizada pela integração de tecnologias avançadas, como a Internet das Coisas (IoT), inteligência artificial (IA) e computação em nuvem nos processos de fabricação (IBM, 2023). Essa integração permite maior eficiência, rastreamento e monitoramento aprimorados da produção e a capacidade de fazer ajustes em tempo real para melhorar a qualidade e reduzir o desperdício. No geral, a Indústria 4.0 visa criar um ambiente de manufatura mais conectado e flexível, com o objetivo de aumentar a produtividade e a competitividade.

A IoT refere-se à conexão de dispositivos do cotidiano, como eletrodomésticos, veículos e equipamentos industriais, à internet. Esses dispositivos são equipados com sensores, processadores e recursos de comunicação, que permitem coletar, transmitir e receber dados. Isso permite que os dispositivos interajam entre si e com outros sistemas e sejam monitorados e controlados remotamente.

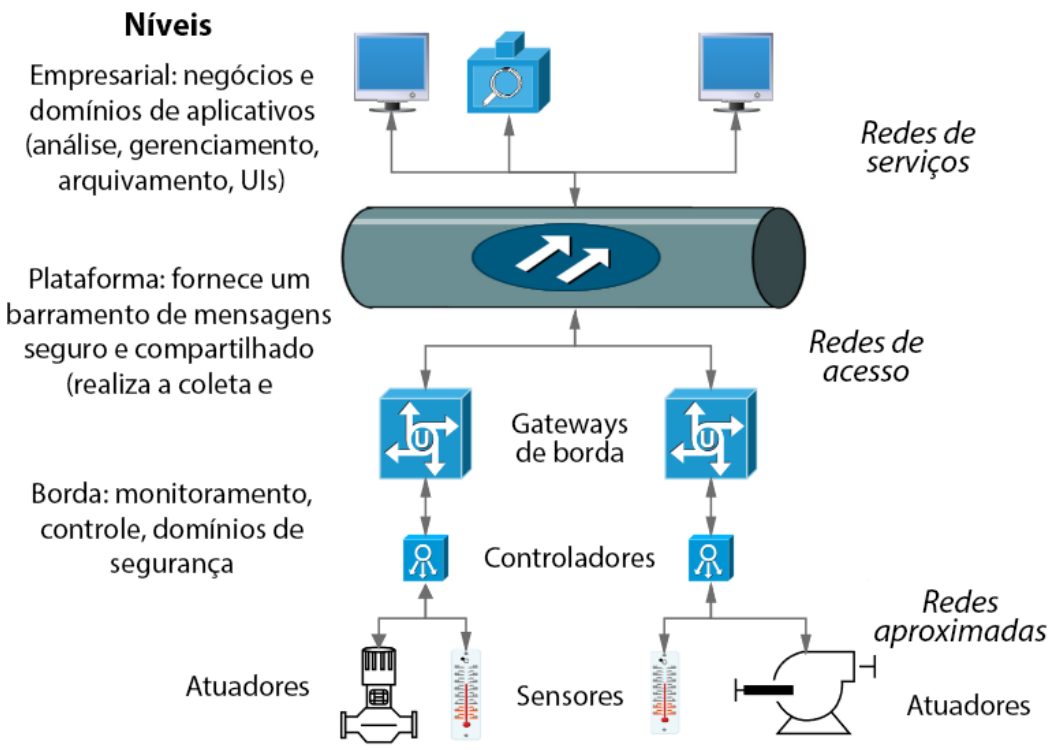
A Internet Industrial das Coisas (IIoT) é um ramo da Internet das Coisas (IoT) que se concentra na aplicação da tecnologia IoT em ambientes industriais. Refere-se ao uso de dispositivos conectados, sensores e outras tecnologias para coletar e transmitir dados de sistemas e equipamentos industriais para um local central para monitoramento, análise e controle.

Embora existam algumas semelhanças, esses conceitos não podem ser substituídos um pelo outro. IoT é mais centrado no consumidor, com comunicações geralmente classificadas como máquina para usuário, enquanto IIoT é focado na indústria, com comunicações máquina a máquina mais críticas, com requisitos mais rigorosos em termos de confiabilidade, segurança e privacidade. A indústria 4.0 é uma extensão do conceito de IoT, com a fusão desse paradigma com a ideia de *Cyber-Physical System* (CPS), e esta fusão tem o objetivo de melhorar a eficiência da produção por meio de serviços inteligentes em fábricas inteligentes (SISINNI, SAIFULLAH, *et al.*, 2018). CPS estendem objetos físicos do mundo real interconectando-os e fornecendo suas descrições digitais, permitindo a criação de serviços inovadores em todo o ciclo de vida do produto.

A arquitetura usada na IIoT pode variar de acordo com o tamanho, quantidade de informações trafegadas e de processo da produção, podendo ter cinco camadas: camada de detecção, camada de acesso, camada de rede, camada intermediária e camada de aplicação (SISINNI, SAIFULLAH, *et al.*, 2018). Como também pode ter apenas três camadas, sendo elas, camada de detecção, camada de rede e camada de aplicação. Onde a camada de detecção é a borda que define o domínio no qual os componentes IIoT interagem uns com os outros. Assim, ela consiste em sensores, controladores e atuadores interconectados por redes locais. A camada de rede conecta a rede local às redes maiores do nível de plataforma, fornecendo cobertura global. Por fim, a plataforma usa a rede de serviço para estabelecer links com o nível empresarial que implementa aplicativos específicos do domínio e fornece interfaces de usuário final. A arquitetura de três camadas pode ser visualizada na figura 1.

Um dos principais benefícios da IIoT é a telemetria, que permite o monitoramento e controle em tempo real dos processos industriais. Isso pode melhorar a eficiência, reduzir o desperdício e melhorar a qualidade geral do produto. Por exemplo, a IIoT pode ser usada para monitorar e controlar a temperatura de um processo de fabricação para garantir que o produto seja produzido na temperatura ideal. Além disso, a IIoT pode ser usada para detectar e diagnosticar falhas de equipamentos, permitindo manutenção imediata e reduzindo o tempo de inatividade.

Figura 1: Exemplo de arquitetura de três camadas



Fonte: Adaptado de (SISINNI, SAIFULLAH, *et al.*, 2018)

A telemetria também pode ser usada para coletar dados sobre o desempenho e uso da máquina, que podem ser usados para melhorar o desempenho geral e a eficiência da máquina. Esses dados também podem ser usados para prever quando uma máquina precisará de manutenção, permitindo uma manutenção proativa em vez de uma manutenção reativa.

Em resumo, a adoção da IIoT e da telemetria na indústria 4.0 está mudando a forma como as empresas operam e produzem. Com essas tecnologias, as empresas podem monitorar e controlar seus processos de produção em tempo real, identificar problemas rapidamente e tomar medidas para resolvê-los. Além disso, as empresas também podem ter uma visão mais ampla de seus processos de produção, permitindo que elas façam previsões e tomem decisões mais informadas.

2.2 O controlador lógico programável

O Controlador Lógico Programável (CLP) pode ser considerado um computador para o ambiente industrial, o CLP surgiu como uma alternativa aos painéis de relés no controle discreto. Com o avanço da tecnologia o CLP passou a realizar tarefas, cálculos e lógicas mais complexas (FRANCHI e DE CAMARGO, 2008). De acordo com a Internacional Electrotecnical Commission (IEC) o CLP pode ser definido como:

> *"Sistema eletrônico operando digitalmente, projetado para uso em um ambiente industrial, que usa uma memória programável para a armazenagem interna de instruções orientadas para o usuário para implementar funções específicas, tais como lógica, sequencial, temporização, contagem e aritmética, para controlar, através de entradas e saídas digitais ou analógicas, vários tipos de máquinas ou processos. O controlador programável e seus periféricos associados são projetados para serem facilmente integráveis em um sistema de controle industrial e facilmente usados em todas suas funções previstas."*

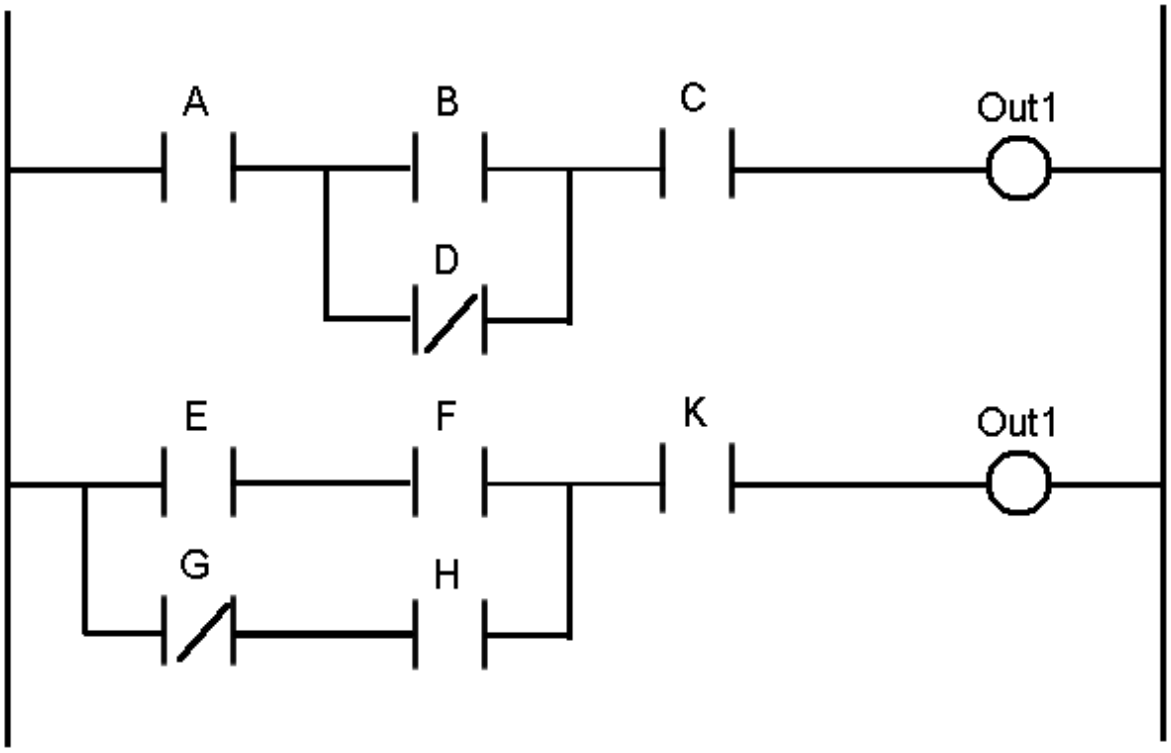
Os CLPs são projetados para resistir a ambientes industriais hostis e geralmente são mais robustos e confiáveis do que os computadores pessoais. Eles são usados em uma ampla gama de aplicações industriais, como: fabricação, embalagem, manuseio de materiais, controle de processos e automação predial.

Atualmente toda planta industrial necessita de algum tipo de controlador para garantir processos seguros e economicamente viáveis. Desde o nível mais simples, em que pode ser utilizado para controlar o motor elétrico de um ventilador para regular a temperatura de uma sala, até um grau de complexidade elevado, controlando a planta de um reator nuclear para produção de energia elétrica. Por ser uma peça tão importante para a indústria, a maioria dos processos de otimização, passam pelo CLP.

Esses controladores geralmente são programados usando uma linguagem de programação especializada, como Lógica Ladder, Diagrama de Blocos Funcionais ou Texto Estruturado. Ladder é a linguagem de programação mais comumente utilizada em CLPs, é uma linguagem gráfica que se assemelha a um esquema

elétrico, é de fácil entendimento e aprendizado, um exemplo dessa linguagem pode ser visto na figura 2.

Figura 2: Diagrama básico da linguagem Ladder



Fonte: (SILVEIRA, 2016)

Os CLPs possuem módulos de entrada e saída (E/S) que se conectam a sensores e atuadores, que são os dispositivos que detectam condições no processo e o controlam. Eles também podem se comunicar com outros dispositivos como computadores, Interface Homem Máquina (IHM) e outros CLPs por meio de protocolos de comunicação como Ethernet, Profinet, Modbus e Profibus.

Os CLPs têm uma ampla gama de recursos, desde o simples controle *on-off* até o complexo controle e monitoramento de processos. Eles também são altamente configuráveis, permitindo que sejam adaptados a aplicações específicas e podem ser facilmente atualizados conforme a tecnologia e os requisitos mudam.

2.3 CLPs da Siemens e os seus protocolos

A Siemens é uma fabricante de CLPs, e sua linha SIMATIC é amplamente utilizada em aplicações industriais em todo o mundo. Esses CLPs estão disponíveis em vários modelos, incluindo as séries S7-300, S7-400, S7-1200 e S7-1500, que oferecem diferentes níveis de desempenho e funcionalidade. Alguns modelos de CLPs Siemens podem ser vistos na figura 3. Esses equipamentos da Siemens são conhecidos por sua robustez e confiabilidade. Eles também oferecem uma ampla gama de módulos de entrada e saída, bem como recursos de comunicação, tornando-os altamente versáteis e adaptáveis a uma variedade de aplicações.

Figura 3: Exemplos de CLPs da Siemens

Fonte: https://www.automation.siemens.com/bilddb/search.aspx

2.3.1 Protocolos de comunicação

Os controladores da Siemens suportam uma variedade de protocolos de rede para comunicação entre diferentes dispositivos e sistemas. Esses protocolos são usados para transmitir dados, sinais de controle e outras informações pela rede.

Alguns dos protocolos de rede mais comumente usados suportados pelos CLPs da Siemens incluem:

- PROFINET: É um protocolo Ethernet industrial desenvolvido pela Siemens, amplamente utilizado em sistemas de controle e automação industrial. O PROFINET suporta comunicação de alta velocidade, controle em tempo real e funções de diagnóstico, tornando-o adequado para uso em ambientes industriais de alto desempenho.
- PROFIBUS: É um protocolo *fieldbus* amplamente utilizado em sistemas de controle e automação industrial. O PROFIBUS é adequado para uso em aplicações de automação de processos e fábricas e suporta uma ampla gama de velocidades de comunicação e tipos de dados.
- MODBUS: Este é um protocolo de comunicação industrial amplamente utilizado que suporta comunicação serial e Ethernet. MODBUS é frequentemente usado para conectar CLPs com outros dispositivos, como sensores, atuadores e displays.
- TCP/IP: Este é o protocolo de comunicação padrão usado para redes na internet, e também é suportado pelos CLPs da Siemens. Ele permite a conexão com outros dispositivos através de redes Ethernet.
- RFC 1006: O protocolo S7 (RFC 1006) permite a conexão de dispositivos Siemens com qualquer parceiro de comunicação. Ele fornece acesso direto à memória.

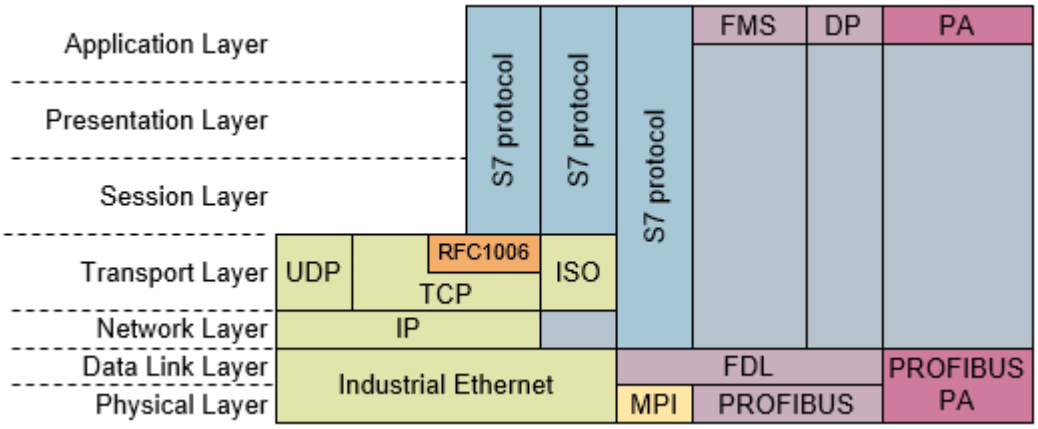
2.3.2 O protocolo Siemens S7 Ethernet

O protocolo Siemens S7 Ethernet (também conhecido como RFC1006) é um protocolo de comunicação usado para conectar CLPs da Siemens a outros dispositivos em redes Ethernet. O protocolo é baseado no protocolo TCP/IP padrão e é utilizado para estabelecer a comunicação entre os CLPs S7 e outros dispositivos, como sistemas IHM, computadores e outros CLPs. Essa comunicação com outros dispositivos é uma das principais vantagens de usar esse protocolo. Isso pode ajudar a melhorar a eficiência e a flexibilidade dos sistemas de automação industrial.

Diferente do protocolo TCP, que não informa o comprimento ou início e fim de uma mensagem, o RFC 1006 especifica informações que devem ser adicionadas na

forma de um cabeçalho aos dados transferidos, fornecendo assim uma transferência orientada a mensagens para aplicativos definidos no protocolo TCP orientado a fluxo de dados (INDUSTRY SUPPORT SIEMENS, 2017). Na maioria das aplicações em tecnologia de automação, é obrigatório trabalhar orientado a mensagens, enviando blocos de mensagens concluídos que podem ser reconhecidos pelo destinatário. Na figura 4 é possível observar a posição do protocolo S7 no modelo OSI, apesar de poder ser utilizado de formas diferentes, neste trabalho o protocolo S7 será utilizado sobre o TCP.

Figura 4: Posição do protocolo S7 no modelo de referência OSI



Fonte: (INDUSTRY SUPPORT SIEMENS, 2019)

O RFC1006 foi projetado para fornecer comunicação rápida e confiável entre dispositivos e suporta transferência de dados cíclica e acíclica. A transferência de dados cíclica é usada para controle e monitoramento em tempo real de variáveis de processo, enquanto a transferência de dados acíclica é usada para dados não críticos no tempo, como configuração e informações de diagnóstico.

As configurações de PUT e GET são importantes em CLPs da Siemens para permitir a comunicação entre o CLP e outros dispositivos externos, como computadores, supervisórios, IHMs e outros controladores. Essas configurações são usadas para transferir dados de entrada e saída entre esses dispositivos, permitindo que as informações sejam compartilhadas e utilizadas para monitorar e controlar processos em tempo real.

O comando PUT é usado para enviar dados do CLP para o dispositivo externo. Esses dados podem incluir informações como valores de tag, como valores de *setpoint* ou estados de máquinas. Por outro lado, o comando GET é usado para receber dados de entrada do dispositivo externo de volta para o controlador. Esses dados podem incluir informações como leituras de sensores, informações de estado do sistema ou atualizações de parâmetros de configuração (SIEMENS, 2020).

2.3.3 O protocolo Modbus

O protocolo Modbus é um protocolo de comunicação usado na automação industrial para conectar dispositivos eletrônicos, como CLPs, sistemas de monitoramento e controle de processos, e outros dispositivos em rede.

O protocolo Modbus opera através de uma comunicação mestre e escravo, em que um mestre envia uma solicitação a um escravo e espera pela resposta. Tal arquitetura confere ao mestre o controle total sobre o fluxo de informações, o que pode ser vantajoso em redes seriais multiponto mais antigas. Ainda que em redes TCP/IP modernas, esse protocolo garanta ao mestre um alto nível de controle sobre o comportamento do escravo, o que pode ser útil em determinados projetos (NI, 2023).

Existem três tipos principais de protocolos Modbus: Modbus RTU, Modbus ASCII e Modbus TCP. O Modbus RTU e o Modbus ASCII são protocolos que operam em nível físico, utilizando sinais elétricos para transmitir dados. Já o Modbus TCP é um protocolo que opera em nível de rede, utilizando o TCP/IP para transmitir dados.

2.4 JavaScript

JavaScript é uma das linguagens de programação mais populares do mundo e é utilizada para desenvolver aplicações em diversas plataformas, como navegadores web, servidores, dispositivos móveis, entre outros. Ela foi criada em 1995 por Brendan Eich, enquanto trabalhava na Netscape, e desde então evoluiu significativamente, se tornando uma linguagem robusta e poderosa.

Uma das principais características do JavaScript é sua capacidade de interagir com o HTML e o CSS, permitindo que os desenvolvedores criem páginas web interativas e dinâmicas. Além disso, o JavaScript também é capaz de executar do lado do servidor, utilizando o Node.js, o que o torna uma opção popular para a criação de aplicações web em tempo real.

Outra característica importante do JavaScript é sua flexibilidade. Ele é uma linguagem de tipagem fraca, ou seja, não é necessário definir o tipo de dado de uma variável antes de utilizá-la. Isso permite que os desenvolvedores escrevam código de forma mais rápida e eficiente. Além disso, o JavaScript é uma linguagem de alto nível, o que significa que os desenvolvedores podem se concentrar em resolver problemas complexos sem precisar se preocupar com os detalhes de baixo nível.

O JavaScript possui uma grande comunidade de desenvolvedores e uma ampla gama de bibliotecas e *frameworks*, que ajudam a tornar o processo de desenvolvimento mais fácil e eficiente. Alguns dos *frameworks* mais populares incluem o React, o Angular e o Vue.js.

2.4.1 TypeScript

TypeScript é uma extensão da linguagem JavaScript, que adiciona recursos de programação orientada a objetos, como tipagem estática, interfaces, classes e herança. A principal característica do TypeScript é a sua tipagem estática, que permite que os desenvolvedores declarem tipos de dados para as variáveis e parâmetros de função. Isso ajuda a prevenir erros de digitação e permite que o compilador faça uma análise mais precisa do código, o que pode levar a erros de tempo de execução reduzidos e a uma melhor manutenção do código.

2.4.2 JSON

JavaScript Object Notation (JSON) é um formato de dados amplamente utilizado na troca de informações entre sistemas e foi inspirado na sintaxe de objetos do JavaScript. O formato JSON utiliza uma estrutura de pares chave-valor, onde cada par é separado por vírgula e cada chave é mapeada a um valor. Os valores

podem ser de diferentes tipos, como texto, números, booleanos, listas e até mesmo outros objetos JSON, permitindo assim uma grande flexibilidade na representação de dados.

Uma das principais vantagens do JSON é a sua compatibilidade com várias linguagens de programação, incluindo JavaScript, Python, Java e muitas outras. Além disso, o formato é amplamente suportado por *Application Programming Interfaces* (APIs) de serviços *web*, tornando-se uma escolha popular para a integração de sistemas.

2.5 HTML e CSS

O *Hypertext Markup Language* (HTML) é a linguagem utilizada para estruturar o conteúdo de uma página *web*. Ele é responsável por definir a hierarquia dos elementos do documento, como cabeçalhos, parágrafos, listas, imagens, entre outros. O HTML é uma linguagem de marcação e não de programação, ou seja, ele não é capaz de realizar operações lógicas e condicionais como uma linguagem de programação.

Já o *Cascading Style Sheets* (CSS) é a linguagem utilizada para estilizar a aparência visual de uma página web. Ele permite que os desenvolvedores definam cores, fontes, tamanhos e posições dos elementos de um site, tornando-o mais atraente e fácil de ler. O CSS também permite que os desenvolvedores criem layouts responsivos, que se adaptam a diferentes tamanhos de tela, desde computadores de mesa até smartphones e tablets.

HTML e CSS são duas linguagens de marcação e estilização, respectivamente, utilizadas na criação de páginas web. Juntas, elas são responsáveis por definir a estrutura e o visual de um site ou aplicação web.

O HTML e o CSS são linguagens complementares e trabalham juntos para criar páginas web atraentes e funcionais. Eles são compatíveis com diversos navegadores web e plataformas, tornando-os essenciais para o desenvolvimento de sites e aplicações web.

Atualmente, existem diversas ferramentas e frameworks que auxiliam os desenvolvedores no processo de criação de páginas web utilizando HTML e CSS,

como o Bootstrap. Esse *framework* oferece componentes pré-definidos e estilos para tornar o processo de desenvolvimento mais rápido e eficiente.

2.6 O Node.js

Node.js é um ambiente de tempo de execução Javascript multiplataforma e de código aberto que executa o código fora de um navegador da *web*. O Node.js usa um modelo de entrada e saída sem bloqueio e orientado a eventos que o torna leve e eficiente, perfeito para aplicativos em tempo real com uso intensivo de dados executados em dispositivos distribuídos. Node.js é comumente usado para desenvolvimento *web*, aplicativos em tempo real e projetos de IoT. Ele também possui uma grande e ativa comunidade de desenvolvedores que contribui para seu desenvolvimento e mantém uma riqueza de módulos e pacotes de código aberto.

Os pacotes do Node.js, também conhecidos como módulos, são bibliotecas de código reutilizável que contêm funcionalidades específicas que podem ser usadas em diferentes projetos. Eles são distribuídos através do registro do *Node Package Manager* (NPM), que é o gerenciador de pacotes padrão do Node.js. Existem dois tipos de pacotes Node.js: pacotes principais e pacotes de terceiros. Os pacotes principais estão incluídos na distribuição do Node.js e fornecem funcionalidade básica, como acesso ao sistema de arquivos e comunicação de rede. Pacotes de terceiros são desenvolvidos pela comunidade Node.js e podem ser facilmente instalados e usados em um projeto. O uso de pacotes pode economizar muito tempo e esforço para os desenvolvedores, pois eles não precisam criar tudo do zero.

2.7 PHP

O PHP é uma linguagem de programação que permite a criação de sites dinâmicos e interativos. Com o PHP, é possível criar páginas HTML com conteúdo dinâmico, como formulários de cadastro, sistemas de busca, fóruns, entre outros.

A principal característica que difere o PHP do JavaScript é a execução do código no servidor, gerando HTML que é posteriormente enviado ao navegador

(PHP, 2023). O navegador recebe os resultados da execução do script, mas não tem acesso ao código fonte.

Além disso, o PHP possui integração com diversos bancos de dados, o que facilita o armazenamento e a recuperação de informações em sites e sistemas. Entre os bancos de dados suportados pelo PHP, destacam-se o MySQL, PostgreSQL e Oracle.

2.7.1 O Laravel

Laravel é um *framework* de código aberto baseado em PHP para o desenvolvimento de aplicações *web*, lançado em 2011. Uma das características do Laravel é que ele oferece uma grande quantidade de funcionalidades pré-construídas, como autenticação, gerenciamento de banco de dados, roteamento, cache e sessões, que tornam o desenvolvimento de aplicações *web* mais rápido e fácil.

Além disso, o Laravel tem uma arquitetura moderna e escalável, com recursos como *migrations* e modelos eloquentes, que facilitam a manutenção do código e a escalabilidade da aplicação.

2.8 Aplicativos para dispositivos móveis

Os aplicativos móveis estão revolucionando a maneira como as fábricas operam, trazendo maior eficiência e conveniência ao processo de fabricação. Os aplicativos móveis oferecem uma série de benefícios para as fábricas, incluindo monitoramento em tempo real da produção, comunicação entre trabalhadores e rastreamento aprimorado do estoque. Esses benefícios estão ajudando a simplificar as operações, reduzir custos e melhorar a produtividade geral.

2.8.1 O Ionic

O Ionic é uma plataforma de código aberto que fornece uma ampla gama de ferramentas para o desenvolvimento de aplicativos móveis que funcionam em várias

plataformas. O *framework* utiliza o AngularJS e o Apache Cordova para construir aplicativos de alta performance com recursos avançados que rodam em dispositivos iOS e Android.

Um dos grandes benefícios do Ionic é a sua capacidade de criar aplicativos que tenham aparência e funcionalidade consistentes em várias plataformas. Isso é alcançado através do uso de componentes de interface do usuário e estilos que foram especificamente projetados para funcionar com diferentes plataformas móveis, tornando mais fácil para os desenvolvedores criarem aplicativos que pareçam nativos para cada plataforma.

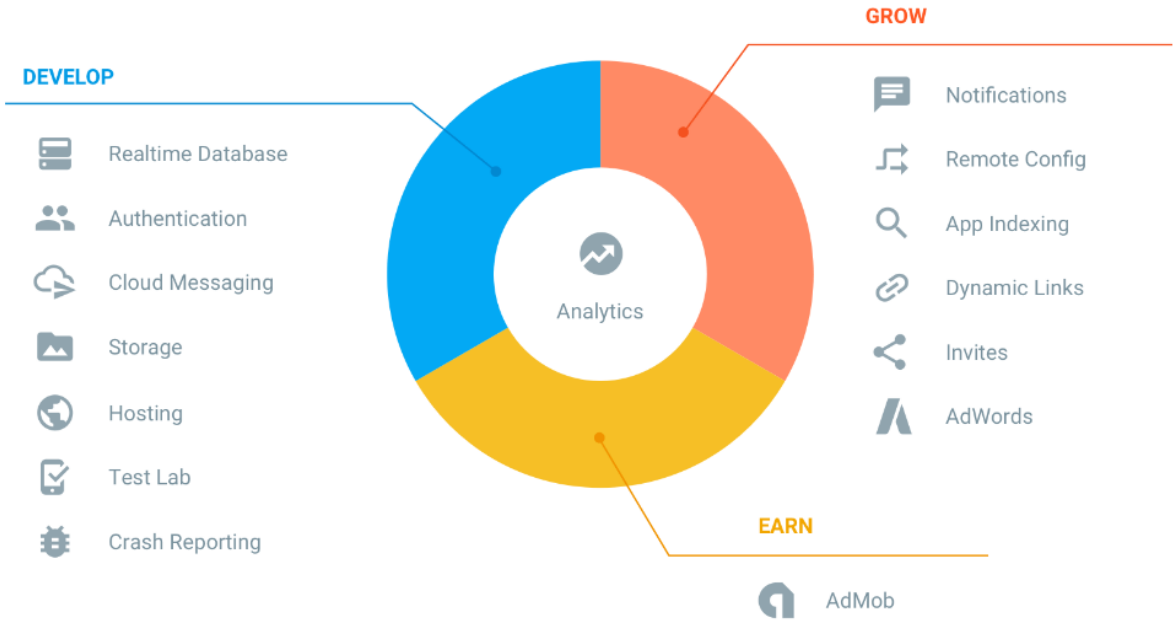
2.9 O Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos móveis e de web oferecida pela Google. Ele fornece uma ampla gama de recursos e ferramentas que facilitam o desenvolvimento e implantação de aplicativos, incluindo o armazenamento de dados em nuvem, autenticação de usuários, análise de dados e muito mais.

Um dos principais benefícios do Firebase é a sua simplicidade e rapidez. Ele fornece uma interface intuitiva que permite aos desenvolvedores criar e gerenciar seus aplicativos de forma fácil e eficiente. Além disso, a plataforma é altamente escalável e permite que os aplicativos sejam adaptados à medida que as necessidades dos usuários evoluem.

O Firebase também oferece uma ampla gama de recursos de armazenamento em nuvem, incluindo o armazenamento de dados em tempo real e o armazenamento de arquivos. Estes recursos permitem que os aplicativos armazenem e acessem facilmente grandes quantidades de dados, sem se preocupar com a infraestrutura subjacente. Algumas das ferramentas disponíveis no Firebase podem ser vistas na figura 5.

Figura 5: Ferramentas disponíveis no Firebase



Fonte: https://codehotfix.com/firebase-completed-guide-for-xamarin-applications/

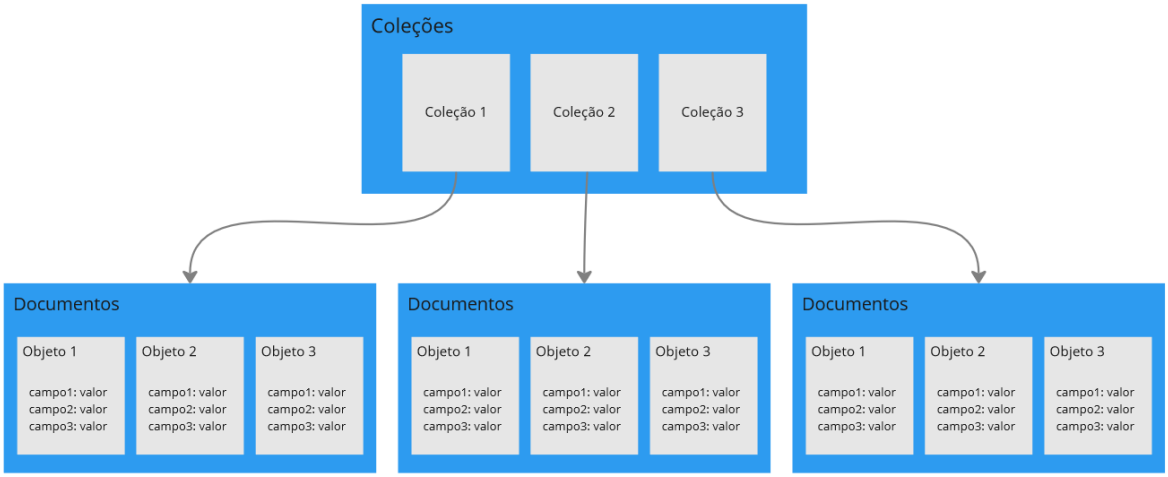
O Firebase Cloud Storage é projetado para fornecer aos desenvolvedores uma maneira fácil de armazenar e acessar arquivos, incluindo imagens, vídeos e outros tipos de mídia, diretamente a partir dos seus aplicativos. Um dos principais benefícios do Firebase Cloud Storage é a sua escalabilidade. Ele permite que os desenvolvedores armazenem e acessem grandes quantidades de dados, mesmo à medida que o número de usuários e o volume de dados crescem. Além disso, a plataforma é projetada para ser altamente segura, garantindo que os dados armazenados sejam protegidos contra acessos não autorizados.

A estrutura de dados do Firebase Cloud Storage é baseada em coleções e objetos ou documentos. As coleções são semelhantes a pastas em um sistema de arquivos. Elas são usadas para agrupar objetos relacionados e torná-los mais fáceis de gerenciar. Cada coleção tem um nome exclusivo e é armazenada no Firebase Cloud Storage como um objeto. Isso significa que as coleções podem ser tratadas como objetos do Firebase Cloud Storage.

Dentro de cada coleção, os objetos são organizados de forma hierárquica, permitindo que os desenvolvedores criem estruturas de dados complexas. Cada objeto contém dados do arquivo e seus metadados, bem como uma chave exclusiva que o identifica dentro de sua coleção. Dentro dos documentos também pode existir

outras coleções, com outros documentos. A figura 6 mostra um exemplo de estrutura dos dados no Firebase.

Figura 6: A estrutura dos dados no Firebase



Fonte: De autoria própria.

Outra vantagem importante é a capacidade de armazenar dados em tempo real. Isso permite que os aplicativos acessem facilmente e atualizem os dados em tempo real, sem precisar de uma conexão constante com um servidor. Além disso, a plataforma oferece recursos de sincronização em tempo real, permitindo que os usuários acessem seus dados de forma rápida e confiável, mesmo em ambientes com baixa largura de banda.

2.10 O Electron

O Electron é uma plataforma de desenvolvimento de aplicativos de desktop baseada em tecnologias web, como Javascript, HTML e CSS. Ele permite que os desenvolvedores criem aplicativos para Windows, MacOS e Linux utilizando as mesmas ferramentas e tecnologias utilizadas na criação de aplicativos web.

Uma das principais vantagens do Electron é a sua facilidade de uso. Como ele utiliza tecnologias web amplamente conhecidas e amplamente disponíveis, os desenvolvedores podem se concentrar em criar aplicativos inovadores sem se

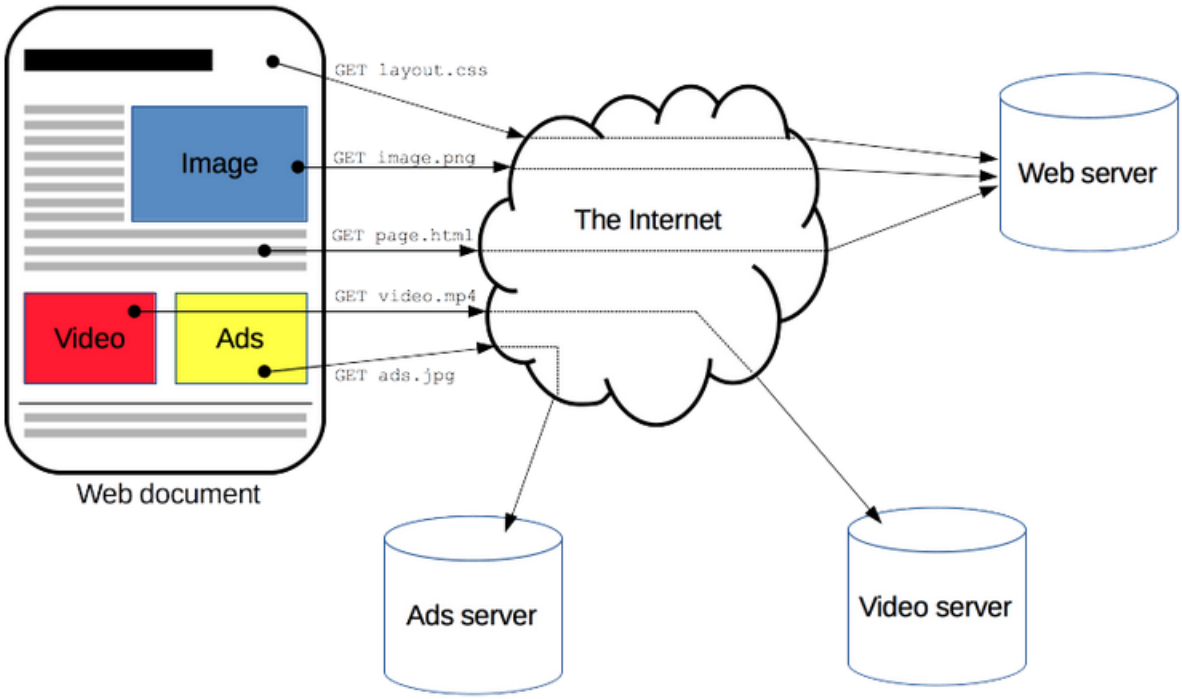
preocupar com as dificuldades técnicas da plataforma. Além disso, o Electron é altamente escalável, permitindo que os aplicativos cresçam e evoluam conforme as necessidades dos usuários mudam.

2.11 HTTP

Hypertext Transfer Protocol (HTTP) é um protocolo de comunicação utilizado na internet para transferir informações de uma aplicação para outra, como por exemplo, de um navegador para um servidor web. Ele foi criado com o objetivo de ser um protocolo padrão para a transferência de informações na *World Wide Web* (WWW) e é amplamente utilizado para acessar páginas *web*, fazer download de arquivos, enviar informações a formulários on-line e para outros propósitos.

O HTTP é baseado em requisições e respostas, ou seja, uma aplicação envia uma requisição para o servidor e o servidor retorna uma resposta. A requisição contém informações sobre o que o cliente deseja, enquanto a resposta contém informações sobre o que o servidor está fornecendo. O HTTP suporta vários métodos de requisição, incluindo GET, POST, PUT, DELETE, entre outros, que determinam a ação que o servidor deve realizar. Na figura 7 está presente alguns exemplos de requisições que são feitas por uma página, é possível perceber que uma mesma página *web* pode fazer requisições HTTP para diferentes servidores.

Figura 7: Requisições do HTTP de uma página *web*



Fonte: (MDN CONTRIBUTORS, 2022)

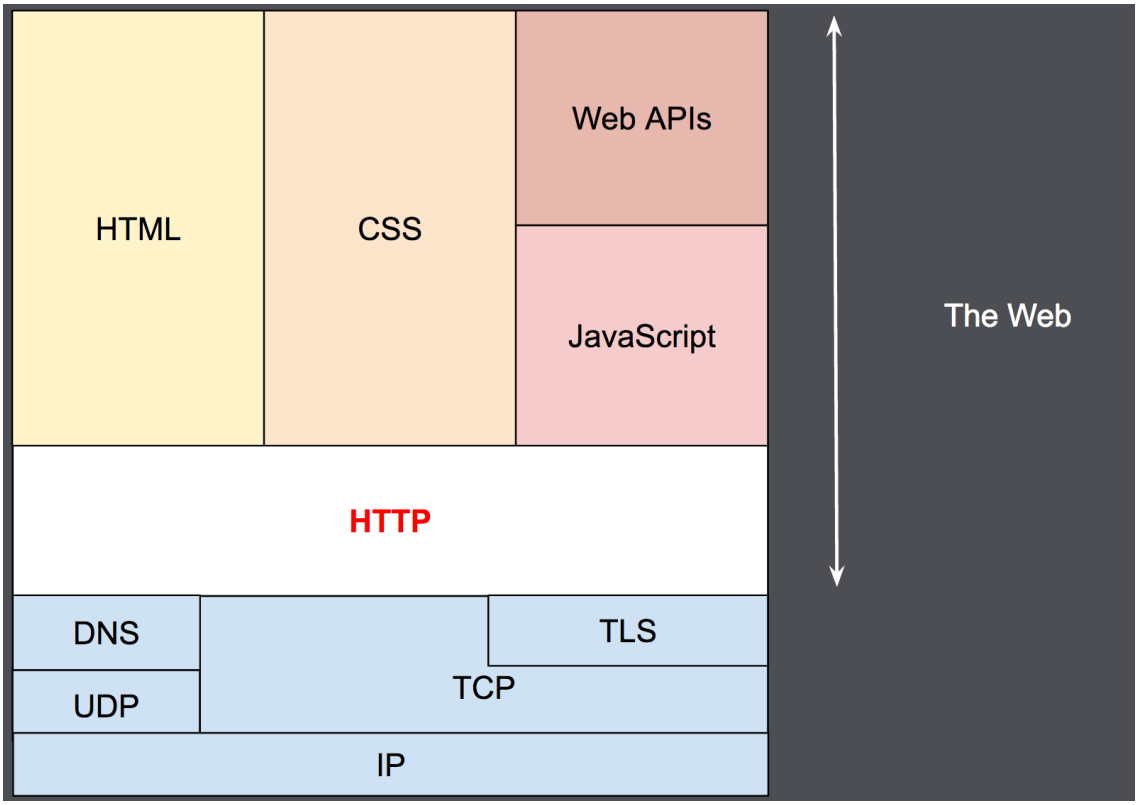
O HTTP utiliza o TCP como seu protocolo de transporte subjacente. Isso significa que quando um cliente solicita uma página da *web* de um servidor, o HTTP envia essa solicitação para o TCP, que se encarrega de dividir os dados em pacotes, enviá-los através da rede e garantir que eles cheguem corretamente ao servidor. Quando o servidor responde com os dados solicitados, o TCP garante que eles sejam entregues ao cliente de maneira confiável. A figura 8 mostra as camadas de uma página *web*, é possível observar que o HTTP está sobre o TCP.

Hyper Text Transfer Protocol Secure (HTTPS) é o protocolo de segurança da web que permite que informações sejam transferidas de maneira segura entre um servidor e um cliente. É uma extensão do protocolo HTTP que adiciona criptografia e autenticação aos dados transferidos.

Ao se conectar a um site através de HTTPS, o cliente e o servidor estabelecem uma conexão segura através do uso de um certificado *Secure Sockets Layer* (SSL) ou *Transport Layer Security* (TLS). Esse certificado criptografa as informações transferidas entre o cliente e o servidor, impedindo que terceiros interceptem ou modifiquem essas informações. Além disso, o certificado também autentica o

servidor, garantindo ao cliente que está se conectando ao site correto e não a um site malicioso disfarçado como o site legítimo.

Figura 8: Camadas de uma página *web*



Fonte: (MDN CONTRIBUTORS, 2022)

2.12 Elipse E3

O Elipse E3 é um software de supervisão e controle de processos industriais desenvolvido pela empresa brasileira Elipse Software. Ele oferece ferramentas avançadas de monitoramento, gerenciamento e análise de dados para diversos setores industriais, incluindo energia, água e saneamento, manufatura, mineração, entre outros.

Uma das principais características do Elipse E3 é a sua interface gráfica intuitiva e personalizável, que permite aos usuários criar e visualizar painéis de controle customizados de acordo com as suas necessidades. Além disso, o *software*

suporta a integração com diversas tecnologias de automação, como CLPs, sensores e atuadores.

O Elipse E3 também conta com uma variedade de recursos avançados de análise de dados, como históricos de eventos, gráficos de tendências e alarmes configuráveis, que permitem aos usuários detectar e solucionar problemas com mais eficiência. Além disso, o *software* oferece suporte à comunicação *Open Platform Communications* (OPC), o que facilita a integração com sistemas de terceiros.

2.12.1 O Elipse Mobile

O Elipse Mobile é uma extensão do software de supervisão e controle de processos industriais Elipse E3, desenvolvido pela empresa brasileira Elipse Software. Ele oferece uma solução móvel para monitoramento e gerenciamento remoto de processos industriais, permitindo que os usuários tenham acesso às informações em tempo real em qualquer lugar.

O aplicativo móvel do Elipse E3 é compatível com dispositivos Android e iOS e oferece uma interface intuitiva e fácil de usar, com recursos avançados de visualização de dados e controle remoto de equipamentos. Com ele, os usuários podem monitorar e controlar os processos industriais de maneira eficiente, independentemente da sua localização.

Uma das principais características do Elipse Mobile é a sua capacidade de sincronização de dados com o Elipse E3, o que significa que os usuários podem acessar os mesmos dados e informações em ambos os sistemas. Além disso, o aplicativo móvel permite que os usuários recebam notificações em tempo real de eventos importantes, como alarmes ou falhas de equipamentos.

2.13 O Node-red

Node-red é uma plataforma de desenvolvimento de aplicativos baseada em Node.js. Ele permite que usuários criem fluxos de dados conectando diversos dispositivos e serviços, tais como sensores, APIs, bancos de dados, entre outros.

O Node-red incorpora um estilo de desenvolvimento de "baixo código", onde os desenvolvedores podem criar rapidamente aplicativos significativos sem ter que escrever muitas linhas de código. O termo "baixo código" foi cunhado pela empresa de pesquisa Forrester Research em um relatório publicado em 2014, mas claramente incorpora um estilo de desenvolvimento que remonta a antes disso (O'LEARY, 2019).

O Node-red oferece uma vasta biblioteca de nós que são blocos de construção para a criação de fluxos de dados. Esses nós podem ser facilmente arrastados e conectados, permitindo que os usuários criem fluxos de dados personalizados de maneira ágil. Além disso, essa ferramenta também conta com uma grande comunidade de desenvolvedores e usuários, que contribuem com novos nós e soluções para problemas específicos.

3 DESENVOLVIMENTO

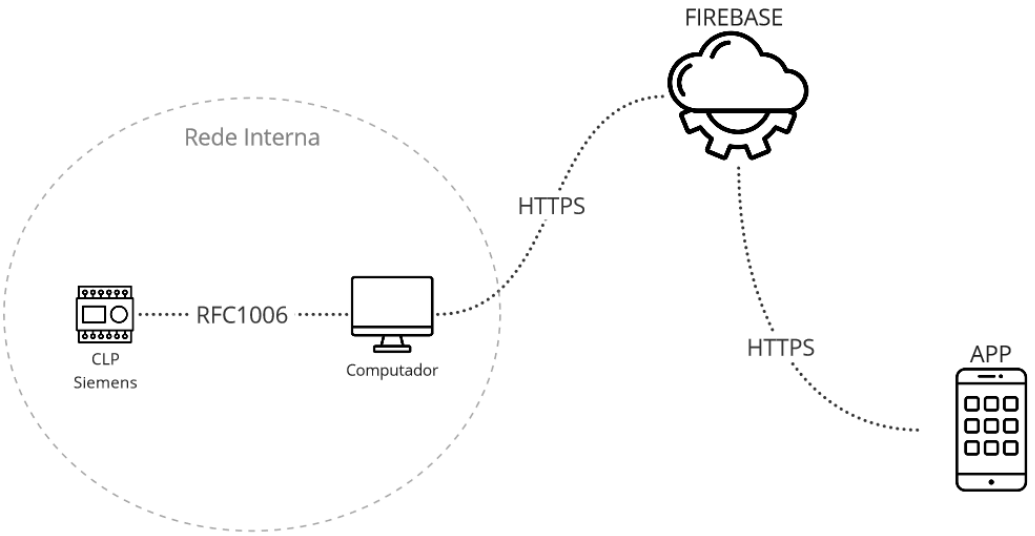
Neste capítulo apresenta-se o método usado para o desenvolvimento deste trabalho. Primeiramente, foi realizada uma pesquisa sobre a Indústria 4.0 e IIoT para obter uma base teórica. Em seguida, foram pesquisadas ferramentas que poderiam ser utilizadas para resolver o problema. Também foram realizados testes e pesquisas sobre a arquitetura do sistema, com o objetivo de definir a disposição dos dispositivos na rede. Após a seleção da arquitetura e das ferramentas adequadas, o sistema e o aplicativo foram desenvolvidos. Por fim, foram realizados testes, análises e comparações com outras propostas para validar a eficácia do sistema.

Pode ser bastante inseguro conectar um CLP diretamente a internet, pois assim a máquina estaria suscetível a vários tipos de ataques. Tendo isso em vista, é necessário o desenvolvimento de uma ferramenta que conecte o CLP a rede externa, isolando o CLP da rede externa. Essa ferramenta possibilita a troca de informações entre o CLP e o servidor. Foi necessário usar um servidor para armazenar as informações. Portanto, através da ferramenta desenvolvida e do servidor ocorre a conexão entre o CLP e o dispositivo móvel.

3.1 Arquitetura desenvolvida

A arquitetura é implementada de forma que, por questões de segurança, o CLP não fica conectado diretamente à internet. Dessa forma, é necessário utilizar um computador na mesma rede do CLP para enviar e receber informações da nuvem. A figura 9 ilustra como as conexões são feitas e quais protocolos são utilizados. As comunicações entre o computador e o servidor, assim como entre o celular e o servidor, são semelhantes, uma vez que ambas têm o mesmo objetivo: receber e enviar informações para o servidor que está em uma rede externa. Assim, independentemente da arquitetura escolhida, a lógica é a mesma e ambas as conexões utilizam o protocolo HTTPS. No entanto, a comunicação entre o CLP e o computador é diferente, ambos os dispositivos estão na mesma rede e o computador precisa buscar informações no CLP e prepará-las para serem enviadas ao servidor.

Figura 9: Representação da arquitetura desenvolvida



Fonte: De autoria própria.

3.2 Arquiteturas e ferramentas testadas

Foram realizados testes com diferentes ferramentas para comunicar o CLP com um aplicativo para celular. Foram levados em consideração aspectos como facilidade de implementação, eficiência e compatibilidade com outras tecnologias.

3.2.1 Elipse E3

Inicialmente, utilizou-se o Elipse E3 como ferramenta para enviar os dados do CLP para o servidor. Como esta ferramenta já possuía um *driver* de comunicação com os CLPs Siemens, foram desenvolvidos dois códigos em VBScript, o primeiro era executado toda vez que o *drive* de comunicação detectava alguma mudança nos valores das tags e tinha como objetivo enviar os valores alterados para o servidor. O segundo algoritmo realizava requisições para o servidor a cada certo intervalo de tempo para verificar se as informações no servidor foram mudadas, após receber a resposta do servidor, os valores eram escritos nas tags através do *drive* de comunicação.

No servidor foi hospedada uma API utilizado o *framework* Laravel. Essa API contava com apenas duas rotas: uma para receber os dados enviados pelo Elipse e

outra para enviar os dados para Elipse quando requisitado. Além disso, foi necessário criar um banco de dados para armazenar esses dados.

Embora tenha funcionado, optou-se por seguir por outra forma, pois a implementação dessa aplicação no Elipse se mostrou um pouco complexa, era necessário adicionar alguns arquivos de código em um sistema supervisório.

3.2.2 Modbus

Também foi feito um teste utilizando o protocolo Modbus para a comunicação entre o CLP e o computador, neste teste foi desenvolvido um código em Javascript que funcionava como um *gateway* entre o Modbus e o HTTPS para enviar as informações para o servidor, mas esta opção foi descartada devido à necessidade de configuração do Modbus no CLP. O controlador se comportava como um cliente e no computador era necessário executar dois códigos, um como servidor Modbus que recebia as informações do Firebase e disponibilizava para os seus clientes e outro como cliente, que tinha como função enviar as informações recebidas do CLP para a nuvem. Apesar de ser descartada, essa opção foi importante para entender o uso do Javascript na comunicação com o CLP.

3.2.3 Node-red

A terceira ferramenta utilizada foi o node-red, que oferece módulos que permitem a comunicação entre o computador e o CLP, bem como o envio de informações para a nuvem, sem a necessidade de nenhum desenvolvimento de código, apenas usando as funções presentes no node-red. No entanto, esta opção também foi descartada, pois os módulos do node-red eram pouco flexíveis e seria um pouco mais complicado realizar a troca de informações entre o servidor e o CLP.

3.3 A ferramenta escolhida

A ferramenta mais adequada para a realização desta tarefa foi o pacote nodeS7, disponível em (PLC PEOPLE, 2022), pacote este que é de código aberto e

pode ser utilizado por qualquer pessoa que obtenha uma cópia deste *software*. O nodeS7 possibilita a realização da comunicação entre o computador e o CLP através do protocolo Siemens S7 Ethernet.

Com este pacote é possível fazer a leitura e a escrita de dados no CLP, através de um algoritmo em Javascript. Para realizar a comunicação é preciso informar apenas o IP, o *rack* e o *slot* do controlador. Há apenas dois pré-requisitos para a comunicação funcionar, o computador e o CLP precisam estar na mesma rede e as configurações de GET e PUT precisam estar habilitadas no CLP para que o computador consiga ler e modificar as tags. Também é necessário informar as tags que o usuário deseja ler ou escrever, pois assim o pacote sabe exatamente quais dados buscar ou escrever no CLP.

3.3.1 Funções utilizadas

Foram utilizadas as seguintes funções do pacote nodeS7 para realização deste trabalho:

- *initiateConnection*: Inicializa a comunicação entre o computador e o CLP. É preciso informar a porta, o endereço IP, o *rack* e o *slot* do CLP.
- *addItems*: Esta função é usada para informar para o algoritmo quais tags serão acessadas e alteradas,
- *writeItems*: Com essa função é possível alterar os valores das tags no CLP. É preciso informar apenas o endereço da tags e o valor, pode ser usada para escrever mais de uma tag por vez.
- *readAllItems*: Esta função realiza a leitura dos valores de todas aquelas tags que foram adicionadas com a função *addItems.*

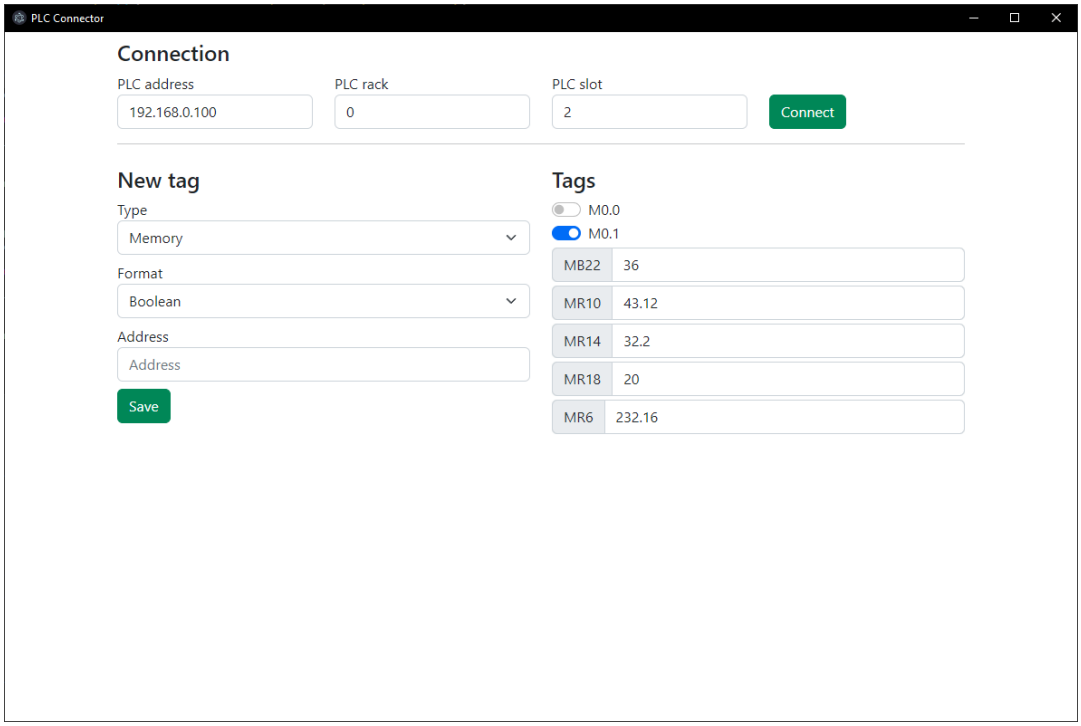
3.4 O Connector

Foi desenvolvido a aplicação para computador, onde era possível através de uma interface gráfica, inserir as informações necessárias para efetuar a comunicação com o CLP. Esta aplicação foi desenvolvida com mais uma ferramenta

baseada em Node.js, o Electron, com a utilização da linguagem de programação Javascript para fazer a lógica do sistema, e utilizando HTML e CSS para desenvolver a interface gráfica da aplicação. Foi dado o nome de Connector para essa aplicação. A interface gráfica da aplicação pode ser vista na figura 10.

O Electron proporciona uma vantagem significativa para o desenvolvimento da aplicação, permitindo que o mesmo código seja utilizado para gerar arquivos executáveis em diferentes sistemas operacionais, não se limitando apenas ao Windows, que foi utilizado no desenvolvimento e testes deste trabalho. Dessa forma, a aplicação pode ser facilmente instalada em computadores de menor capacidade, com sistemas operacionais baseados em Linux, sem apresentar grandes problemas. Tal possibilidade traz benefícios para indústrias que necessitam de soluções eficientes e econômicas em seus processos produtivos.

Figura 10: Interface gráfica do Connector



Fonte: Reprodução

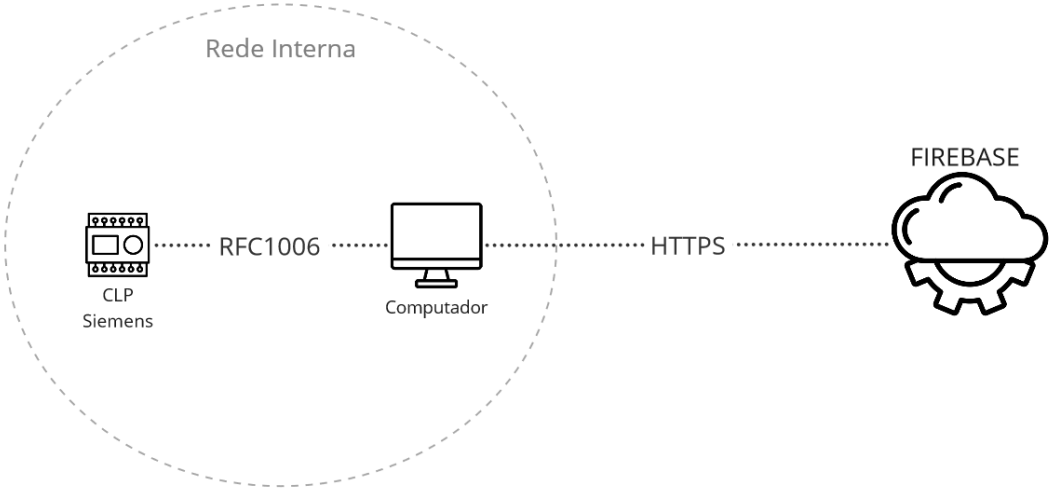
3.4.1 Lógica e comunicação

A aplicação foi desenvolvida de modo que fosse possível ler e escrever tags do tipo memória, no caso dos CLPs da Siemens, as tags do tipo M e ler tags de entrada e saída, tags dos tipos I e Q respectivamente.

Após a comunicação entre o CLP e o computador ser realizada com sucesso, foi necessário se comunicar com algum servidor que armazenasse os dados, o servidor escolhido foi o Firebase, devido a sua facilidade de implementação e a possibilidade de a comunicação ser feita em tempo real, assim a troca de informação entre o CLP e qualquer aplicação externa poderia ser feita de forma rápida e com pouco consumo de banda da rede local. A figura 11 mostrar como os dispositivos estão conectados. O Firebase tem um pacote Node.js que permiti o envio e o recebimento de dados no formato JSON, com o uso deste pacote foi implementado as funções de leitura e de escrita.

Para integrar as comunicações entre o CLP e a aplicação e entre a aplicação e o servidor, foi necessário realizar dois passos. O primeiro passo foi buscar os dados que estavam no Firebase, salvar esses dados localmente e mostrar na interface gráfica, o segundo passo foi possibilitar a alteração desses dados e o envio dos dados alterados de volta para o servidor, dados esses que são as tags do CLP. Também é possível através do sistema, criar tags para serem salvas no servidor.

Figura 11: Disposição dos dispositivos



Fonte: De autoria própria.

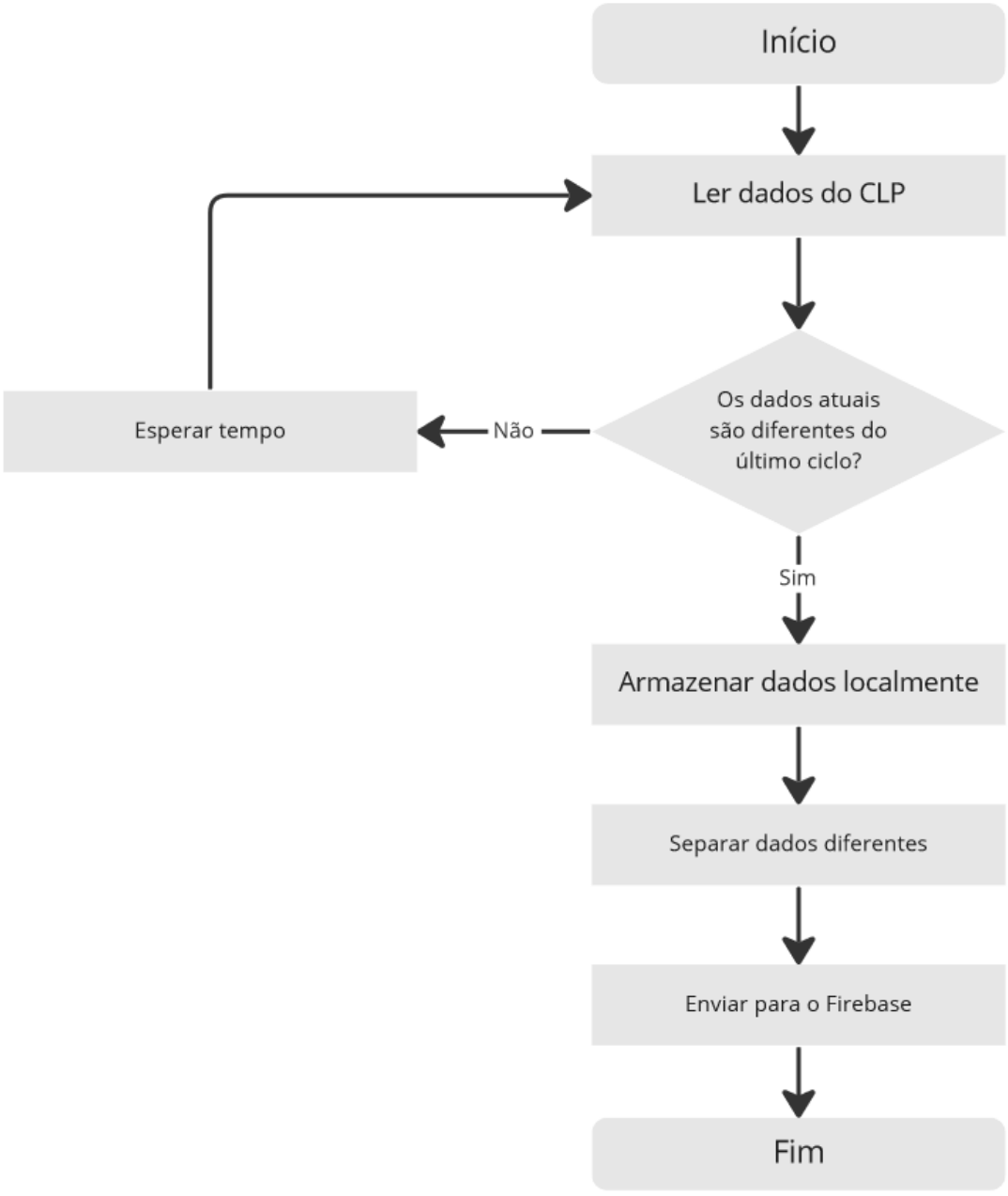
O sistema foi desenvolvido de forma que as tags lidas do CLP sempre tenham seu endereço informado pelo Firebase. Isto significa que, quando o usuário deseja ler o valor de uma certa tag, ele insere o endereço da tag, esta informação é então enviada para o servidor, onde o Firebase retorna a lista de tags para o sistema. Em seguida, é realizada a leitura dos dados atuais do CLP e os dados são salvos no Firebase.

O sistema realiza a leitura das tags a cada intervalo de tempo definido e analisa se esses valores estão diferentes daqueles salvos no servidor, se estiver, é feito o envio dos valores diferentes para o servidor. O diagrama na figura 12 apresenta essa lógica, entende-se como “Início” quando a conexão do Connector com o CLP é realizada com sucesso.

A escrita de tags no CLP é feita com base em um evento que é acionado quando ocorre alguma alteração dos valores no Firebase, quando isso ocorre o código comparar os valores recebido com os valores atuais do controlador, se tiver alguma incoerência, o valor no CLP é alterado pelo novo valor. O diagrama na figura 13 apresenta essa lógica, entende-se como “Início” quando é detectado alguma mudança nos dados do Firebase.

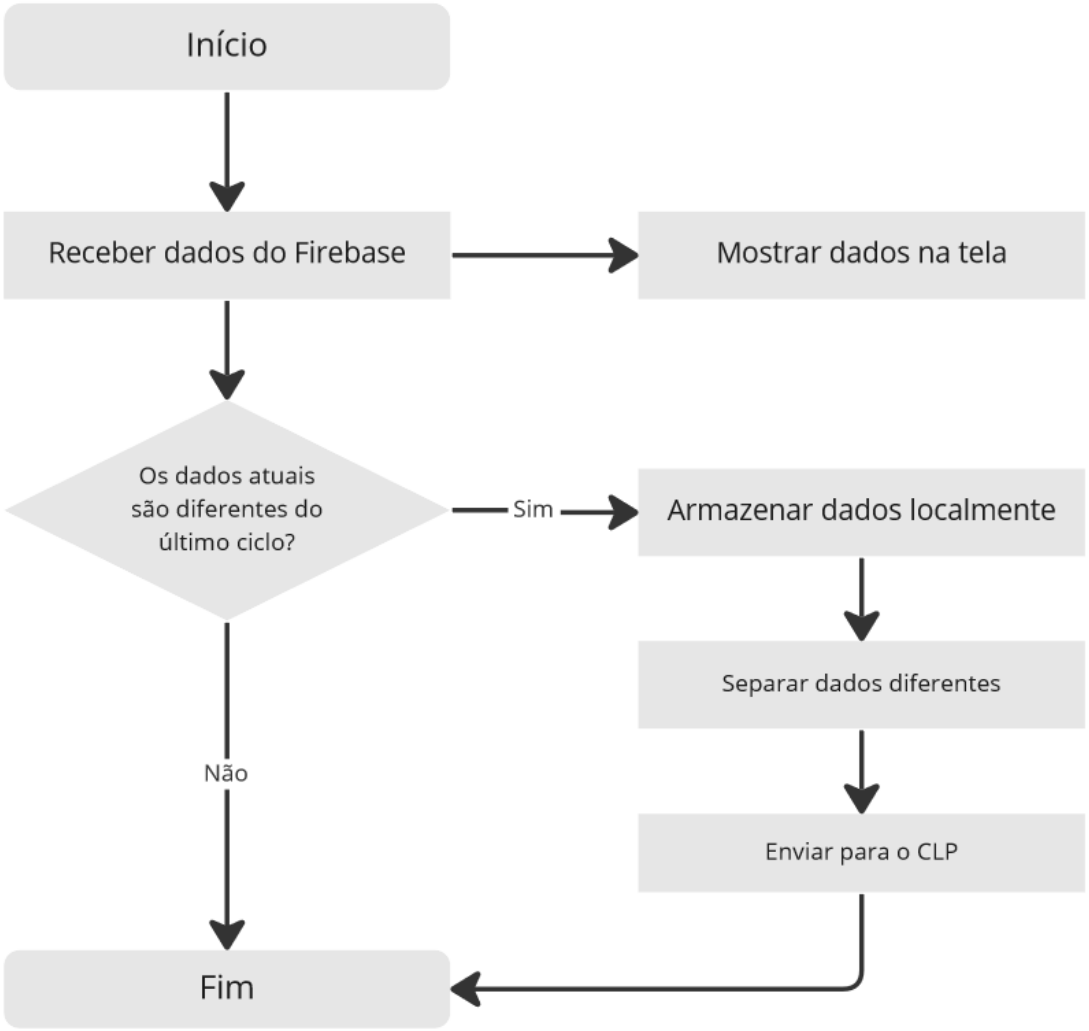
No primeiro ciclo de funcionamento do programa, ou seja, logo após ser realizada a conexão com o CLP, os dados que estão no servidor são inseridos no CLP, então é importante que o controlador não tenha nenhum dado crítico antes da conexão, pois esses dados podem ser alterados.

Figura 12: Diagrama da lógica do envio de dados do CLP para o Firebase



Fonte: De autoria própria.

Figura 13: Diagrama da lógica do recebimento de dados



Fonte: De autoria própria.

3.4.2 A estrutura de dados

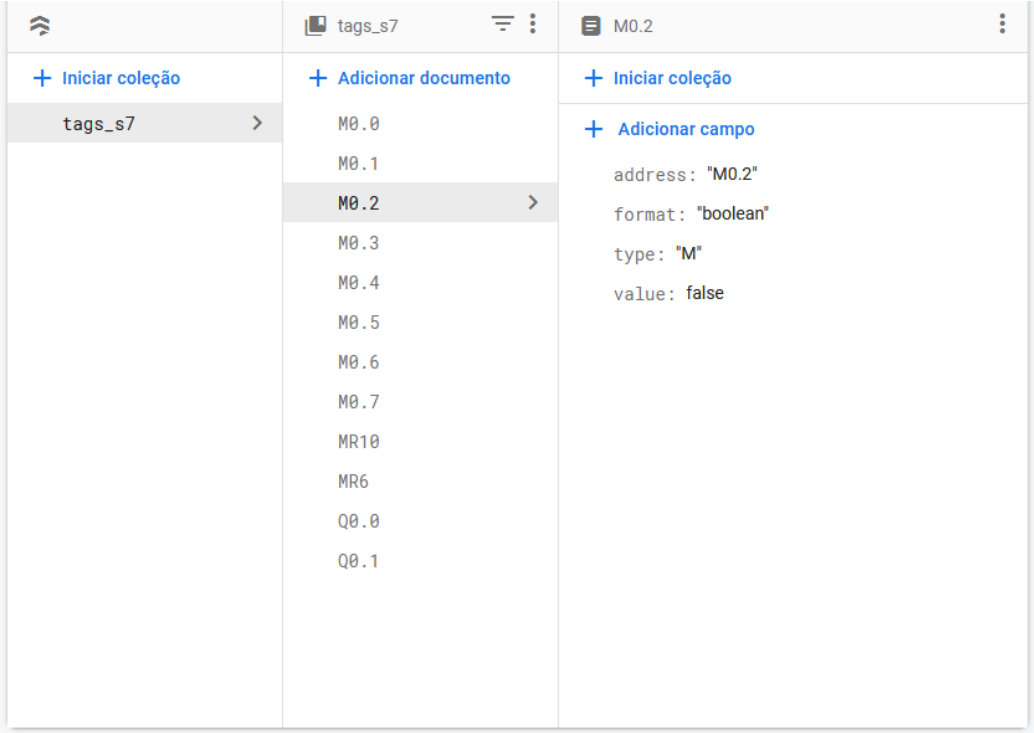
Foi criado uma coleção no Firebase com o nome *tags_s7* para armazenar as informações trocadas entre os dispositivos da arquitetura, dentro dessa coleção se encontra os documentos com as tags do CLP. Para facilitar a identificação dos documentos, foi dado como nome desses documentos o endereço da sua respectiva tag, isso facilita a leitura da tag e a alteração, vide figura 14.

Cada objeto foi estruturado com os seguintes campos:

- Address: O endereço da tag no CLP, deve ser inserido no formato *string*.

- Format: O formato do dado, pode ser do tipo booleano e do tipo decimal. Deve ser inserido como *string*.
- Type: Indica qual é o tipo da tag, pode ser do tipo memória (M), saída (Q) ou entrada (I). Também deve ser inserido como *string*.
- Value: O valor atual da tag, dependendo do formato da tag, esse campo pode ser numérico ou booleano.

Figura 14: A estrutura dos dados do sistema no Firebase



Fonte: Reprodução

3.5 Aplicativo para celular

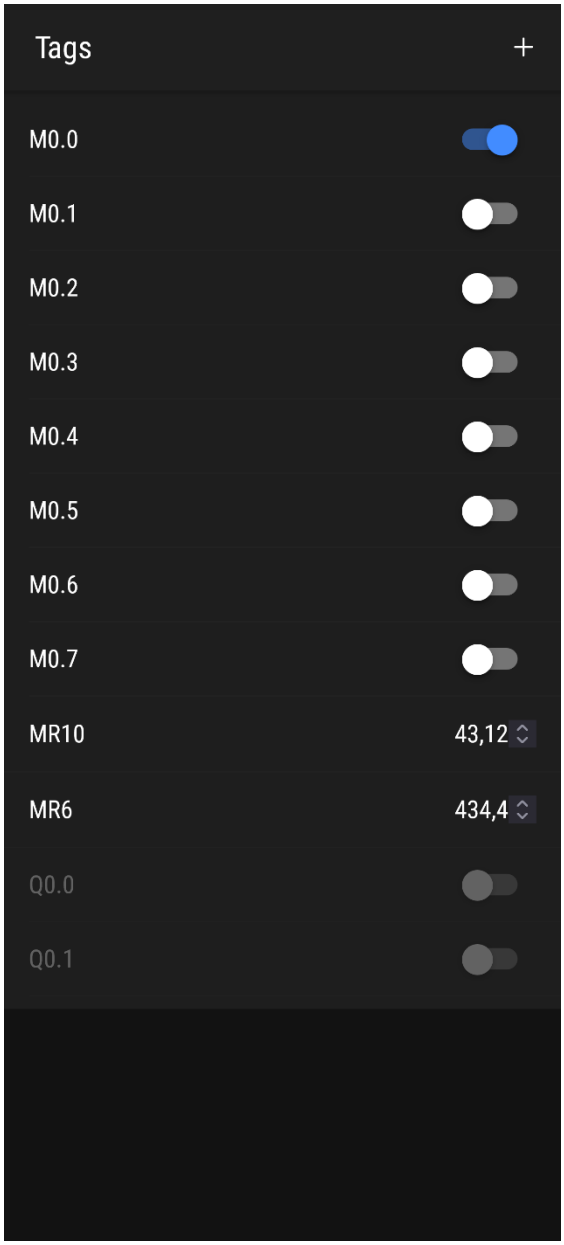
Para proporcionar o acesso aos dados de forma remota foi desenvolvido um aplicativo para celular, que não necessariamente precisa estar na mesma rede do CLP, esse aplicativo consome as informações diretamente do Firebase, e assim é possível ler e escrever informações, enviando o dado para o servidor que automaticamente será enviado para o CLP.

Este aplicativo foi desenvolvido com o Ionic, usando a linguagem de programação TypeScript. Por padrão, o Ionic gera vários módulos e arquivos que facilitam a criação e a compilação do aplicativo, esse foi o principal motivo para a escolha dessa ferramenta no desenvolvimento do APP. Assim, apenas foi necessário criar a comunicação com o servidor, usando o mesmo pacote já citado anteriormente, que possibilita a comunicação entre o Firebase e o Node.js. Usando TypeScript e HTML foi desenvolvido uma tela com a lista de tags, possibilitando também a alteração e a criação de tags pelo usuário. Da mesma forma do Connector, quando algum dado é alterado no servidor, automaticamente a alteração é recebida pelo aplicativo e mostrada na tela.

A figura 15 mostra a tela inicial do aplicativo, onde pode ser observado o estado atual das tags, como também pode ser feita a alteração dos seus valores, mas isso só é permitido para as tags do tipo memória, não é permitido para tags de entrada ou saída. Na figura 16 é mostrado o formulário onde é possível salvar uma nova tag, para isso é preciso informar o tipo, o formato e o endereço da tag no CLP. Também foi desenvolvido uma função para deletar tags, para isso, bastar arrastar o item para a esquerda e irá aparecer um ícone de lixeira, que quando pressionado acontece a deleção.

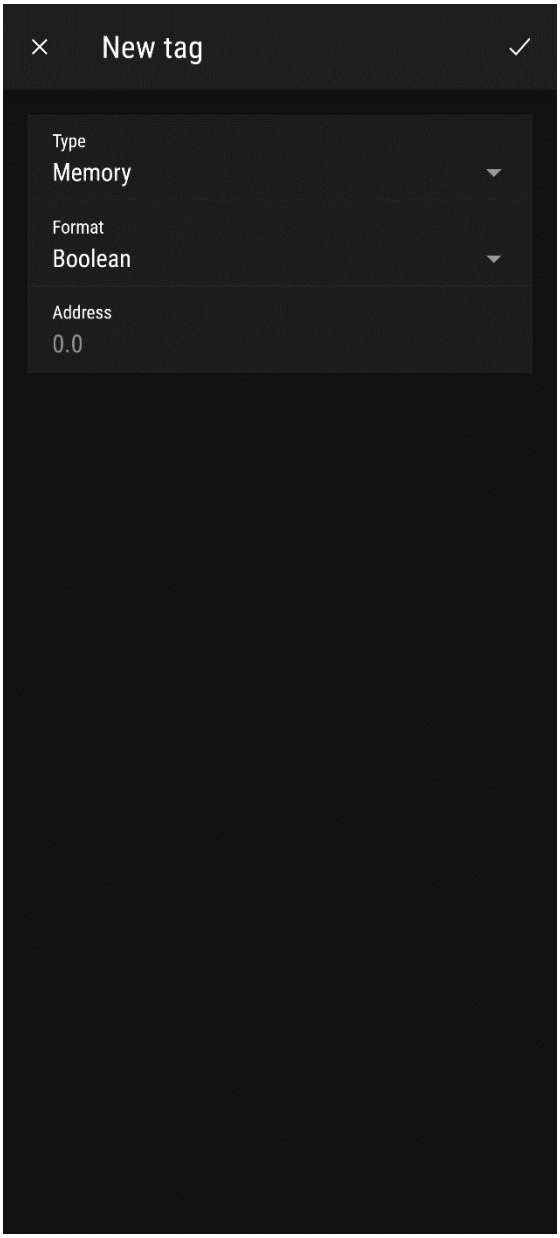
Por fim, a utilização do Firebase como banco de dados em tempo real facilitou a criação da API, permitindo que a comunicação entre o CLP e o aplicativo fosse realizada de forma simples e segura. A tecnologia também forneceu recursos para garantir a confiabilidade e escalabilidade do sistema. Assim, a comunicação entre o CLP e o aplicativo foi realizada com sucesso, demonstrando que a utilização de tecnologias modernas como Node.js, Ionic e Firebase podem trazer inúmeras vantagens na criação de sistemas complexos.

Figura 15: Tela inicial do aplicativo



Fonte: Reprodução

Figura 16: Tela para criação de novas tags

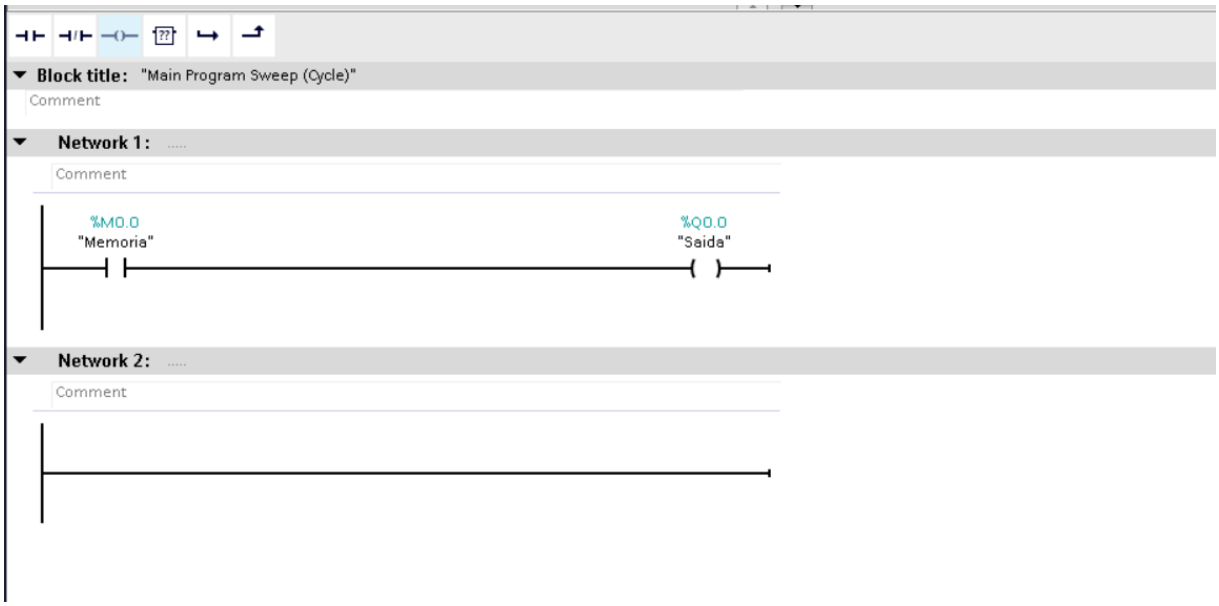


Fonte: Reprodução

3.6 Testes realizados

Foram feitos testes com dois tipos de controladores em ambiente de simulação, com S7-300 e com o S7-1200, para os dois equipamentos a comunicação entre o CLP e o aplicativo funcionou perfeitamente. Para esse teste foi criada uma lógica simples em Ladder onde uma tag do tipo memória alimentava uma saída, como pode ser visto na figura 17. O software Tia Portal foi usado para programar o CLP e o PLCSIM da Siemens foi usado para simular os controladores.

Figura 17: Lógica Ladder usada para testes



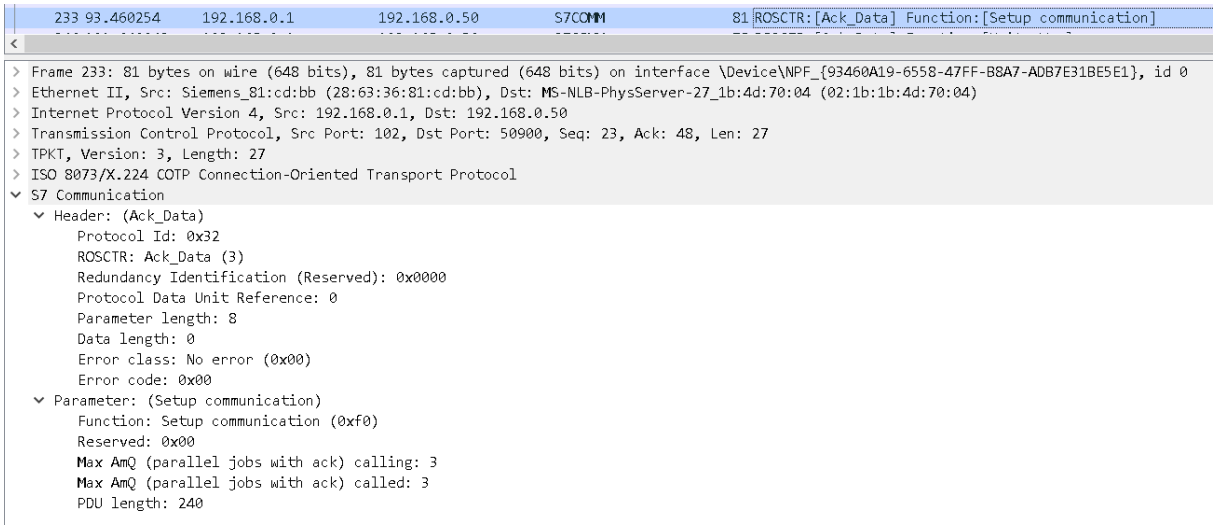
Fonte: Reprodução

Também foram realizados testes com a CPU física S7-1214C DC/DC/DC no laboratório, para esse teste foi usado a mesma lógica do teste anterior. A aplicação funcionou como esperado nesse teste.

Durante o desenvolvimento de um projeto de comunicação entre um CLP e um computador, foram realizados testes para avaliar a integridade da comunicação entre os dispositivos. Para tal, utilizou-se o *software* Wireshark para analisar os dados trafegados entre os dispositivos. Para isso foi usado o filtro S7comm que retorna apenas os registros que são feitos no protocolo S7. Comparando a figura 18 com a figura 4, é possível identificar as camadas do modelo OSI, como a camada de

rede, que na imagem é possível identificar como “*Internet Protocol Version 4*”, ou seja, IPv4, que tem informações importantes como o IP de destino e de origem. Na camada de transporte está o TCP, que tem a informação da porta usado pelo dispositivo de destino e a porta de origem.

Figura 18: Estabelecendo comunicação com o CLP

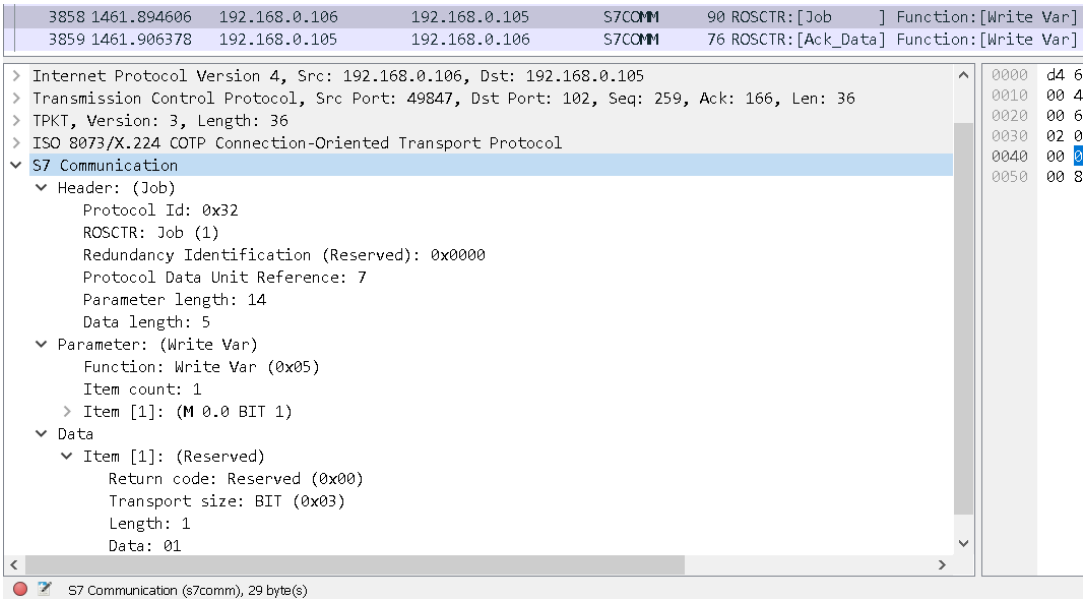


Fonte: Reprodução

Com base na figura 19, é possível identificar dois registros no Wireshark durante o processo de escrita no CLP. O primeiro registro corresponde ao envio da requisição de escrita do computador para o CLP, enquanto o segundo registro corresponde à resposta. Conforme ilustrado na figura, a requisição de escrita inclui o endereço da tag e o valor a ser escrito.

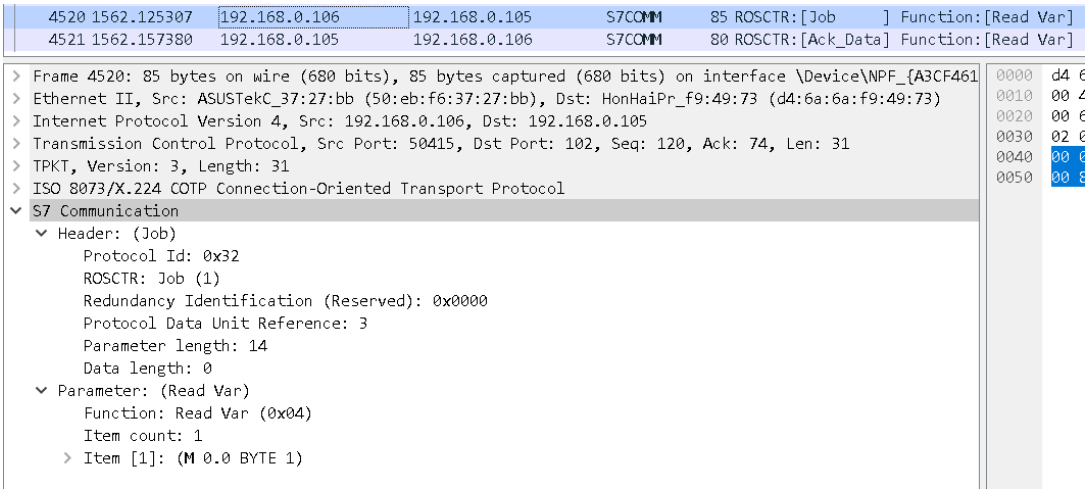
Além disso, durante o teste de leitura de tags do CLP, também foi necessário enviar o endereço da tag que seria lida, como pode ser visto na figura 20. Essa informação é fundamental para que o software de comunicação possa identificar a tag correta e enviar os dados para o computador.

Figura 19: Escrevendo dados no CLP



Fonte: Reprodução

Figura 20: Lendo dados do CLP



Fonte: Reprodução

4 COMPARAÇÃO COM OUTRAS FERRAMENTAS

Uma vantagem da proposta realizada nesse trabalho em relação ao Elipse Mobile é a compatibilidade com outros sistemas. O Elipse Mobile não permite a troca de informações com outros sistemas, como por exemplo, um *site*. Outra vantagem é que a proposta não precisa de nenhum outro sistema para se conectar com o CLP, ou seja, está diretamente conectado. Já o Elipse Mobile precisa de um sistema supervisório desenvolvido no Elipse E3 para fazer a conexão com o CLP. Uma vantagem do Elipse Mobile é que pode ser usado com diferentes tipos de controladores e não apenas controladores da Siemens.

Em (DIAZ, RIVERA, *et al.*, 2021), os dados do CLP são enviados para o Firebase. Em seguida, o aplicativo móvel lê esses dados do banco de dados e exibe as informações relevantes para o usuário em tempo real, assim como a aplicação desenvolvida neste trabalho. O artigo usa uma ferramenta da Siemens chamada *Automation Web Programming* (AWP), para disponibilizar os dados do CLP em um servidor *web* local. Um Raspberry Pi conectado à rede local é utilizado para buscar os dados no servidor local e enviá-los para o Firebase. No entanto, a comunicação é unidirecional e não é possível enviar dados do Firebase para o CLP, o que é uma desvantagem.

5 CONCLUSÃO E PROPOSTA DE CONTINUAÇÃO

É possível concluir que a realização da conexão e troca de dados entre um CLP e um aplicativo para celular é uma solução viável para monitoração e controle remoto de equipamentos industriais. Através da integração de tecnologias como o Node.js, Ionic e Firebase, foi possível desenvolver uma aplicação que permite a comunicação em tempo real entre o CLP e o aplicativo, possibilitando a visualização de informações e o envio de comandos de forma rápida e segura. A utilização de uma interface amigável e intuitiva tornou a experiência do usuário mais eficiente e agradável. Dessa forma, pode-se concluir que a aplicação desenvolvida apresenta uma boa solução para o monitoramento e controle remoto de equipamentos industriais.

Devido a escalabilidade de aplicações feita com Node.js, o sistema desenvolvido pode ser integrado a outras ferramentas, como sistemas SCADA, mesmo que esses sistemas funcionem com outros protocolos de rede ou outras linguagens de programação, pois é possível adicionar um novo modulo na aplicação para realizar a comunicação com diferentes ferramentas, como por exemplo o Modbus, que foi usado para testes neste trabalho.

O Firebase também pode ser usado para se comunicar com outras ferramentas, como sites ou API, a maioria das linguagens usadas para programação *web* têm pacotes para conexão com o Firebase, assim a informação que foi retirada do CLP e salva no Firebase poderia ser enviada para qualquer sistema.

Sugere-se como trabalho futuro adicionar no sistema a conexão com CLP da Rockwell, que pode seguir a mesma arquitetura desenvolvida neste trabalho, pois existe um pacote no Node.js para realizar a comunicação com CLPs da Rockwell. Por ter sido desenvolvido pelo mesmo grupo, este pacote é bem parecido com o modulo usado neste trabalho, ou seja, não será preciso realizar grandes mudanças na arquitetura, apenas algumas alterações no código e na lógica do Connector. Como também será necessário alterar um pouco a estrutura de dados no Firebase, que atualmente está preparada para receber dados de apenas um CLP, e assim futuramente pode receber dados de mais de um CLP, independente do fabricante.

REFERÊNCIAS

ABOUT npm. **npm Docs**. Disponível em: <https://docs.npmjs.com/about-npm>. Acesso em: 6 Fevereiro 2023.

AYDOGMUS, Zafer; AYDOGMUS, Omur. A Web-Based Remote Access Laboratory Using SCADA. **IEEE Transactions on Education**, Fevereiro 2009. 126-132.

BLEIGH, Michael. HTTP/2 comes to Firebase Hosting. **The Firebase Blog**, 10 fev. 2023. Disponível em: <https://firebase.blog/posts/2016/09/http2-comes-to-firebase-hosting>.

DIAZ, Alexander et al. A Development of a Mobile Application for Monitoring Siemens S7-1200 Controller Variables Through Firebase. **Smart Innovation, Systems and Technologies**, 01 Janeiro 2021.

ELECTRON. Introduction | Electron. **ElectronJS**. Disponível em: <https://www.electronjs.org/docs/latest/>. Acesso em: 18 jan. 2023.

FRANCHI, Claiton M.; DE CAMARGO, Valter L. A. **Controladores Lógicos Programáveis:** Sistemas Discretos.

GOOGLE. Armazenamento em nuvem para Firebase. **Firebase**, 31 jan. 2022. Disponível em: <https://firebase.google.com/docs/storage?hl=pt-br>. Acesso em: 2023 jan. 14.

IBM. What is Industry 4.0 and how does it work? **IBM**, 2023. Disponível em: <https://www.ibm.com/topics/industry-4-0>. Acesso em: 20 Janeiro 2023.

INDUSTRY SUPPORT SIEMENS. What is RFC1006 and what do I need this service for? **Industry Support Siemens**, 2017. Disponível em: <https://support.industry.siemens.com/cs/document/15048962/what-is-rfc1006-and-what-do-i-need-this-service-for-?dti=0&lc=en-BR>. Acesso em: 28 fev. 2023.

INDUSTRY SUPPORT SIEMENS. What properties, advantages and special features does the S7 protocol offer? **Industry Support Siemens**, 2019. Disponível em: <https://support.industry.siemens.com/cs/document/26483647/what-properties-advantages-and-special-features-does-the-s7-protocol-offer-?dti=0&dl=en&lc=pt-BR>. Acesso em: 16 Março 2023.

IPCOMM GMBH. IPCOMM, Protocols: S7 Protocol (RFC 1006). **IPCOMM GmbH**. Disponível em: <https://www.ipcomm.de/protocol/S7ISOTCP/en/sheet.html>. Acesso em: 28 jan. 2023.

MDN CONTRIBUTORS. HTTP. **MDN Web Docs**, 02 nov. 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP>. Acesso em: 2022 out. 12.

MDN CONTRIBUTORS. Uma visão geral do HTTP. **MDN Web Docs**, 2022. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Acesso em: 15 mar. 2023.

NI. Introduction to Modbus using LabVIEW. **NI**, 2023. Disponível em: <https://www.ni.com/pt-br/shop/labview/introduction-to-modbus-using-labview.html>. Acesso em: 04 maio 2023.

O'LEARY, Nick. Low-code Development with Node-RED. **IBM**, 2019. Disponível em: <https://developer.ibm.com/blogs/introducing-node-red-version-1-0/>. Acesso em: 04 maio 2023.

OTTO, Andreas; HELLMANN, Klas. IEC 61131: A general overview and emerging trends. **IEEE Industrial Electronics Magazine**, 11 Dezembro 2009.

PHP. O que é o PHP? **PHP**, 2023. Disponível em: <https://www.php.net/manual/pt_BR/intro-whatis.php>. Acesso em: 04 maio 2023.

PIGAN, Raimond; METTER, Mark. **Automating with PROFINET:** Industrial Communication Based on Industrial Ethernet.

PLC PEOPLE. Node.JS library for communication to Siemens S7 PLCs. **GitHub**, 2022. Disponível em: <https://github.com/plcpeople/nodeS7>. Acesso em: 09 Janeiro 2023.

RADICH, Quinn ; BATCHELOR, Drew ; SATRAN, Michael. Using VBScript - Win32 apps. **Microsoft**, 30 out. 2022. Disponível em: <https://learn.microsoft.com/en-us/windows/win32/lwef/using-vbscript>.

SIEMENS. S7 Communication with PUT/GET. **Siemens**, 2020. Disponível em: <https://cache.industry.siemens.com/dl/files/115/82212115/att_1039294/v2/82212115_S7_communication_Sequencer_TIA_Portal_en.pdf>. Acesso em: 02 mar. 2023.

SILVEIRA, Cristiano B. Como Funciona a Linguagem LADDER - Citisystems. **Citisystems**, 25 jul. 2016. Disponível em: <https://www.citisystems.com.br/linguagem-ladder/>. Acesso em: 07 Outubro 2022.

SISINNI, Emiliano et al. Industrial Internet of Things: Challenges, Opportunities, and Directions. **IEEE Transactions on Industrial Informatics**, 02 Julho 2018.

WEAVER, Alfred C. Internet-based factory monitoring. **IEEE Transactions on Education**, Agosto 2002.

APÊNDICES
APENDICE A – CÓDIGO JAVASCRIPT DO CONNECTOR

preload.js > ...

```
// Imports
var nodes7 = require('nodes7');
const { db } = require("./fb");

// Global variables
var conn = new nodes7;
var firstLoad = false;
var CONNECTED = false;
var TAGS = {};

window.onload = () => {
    const $ = require('jquery');
    get();

    $('#connect').on('click', function () {

        // get PLC infos
        let plc_address = $('#plc_address').val();
        let plc_rack = $('#plc_rack').val();
        let plc_slot = $('#plc_slot').val();
        $('#connect').html('Connecting').prop('disabled', true);
        $('#status').addClass('d-none');

        // connection
        conn.initiateConnection({ port: 102, host: plc_address, rack: plc_rack, slot:
        plc_slot, debug: false }, (err) => {
            if (typeof (err) !== "undefined") {
                $('#status').html('Connection failed!')
                $('#status').removeClass('d-none');
                $('#connect').html('Connect').prop('disabled', false);
                return false;
            }

            $('#status').addClass('d-none');
            $('#disconnect').removeClass('d-none');
            $('#connect').html('Connected').prop('disabled', true);

            get();
            CONNECTED = true;

            //read items
            setInterval(() => {
                if (firstLoad && CONNECTED) {
                    conn.readAllItems(valuesReady);
                }
            }, 500);

        });
    });

```

```
50      // disconnect
51      $('#disconnect').on('click', function () {
52          conn.dropConnection();
53          CONNECTED = false;
54          $('#connect').html('Connect').prop('disabled', false);
55          $('#disconnect').addClass('d-none');
56      })
57
58      // change tag - boolean
59      $(document.body).on('change', '.bool-tag', function () {
60          const type = $(this).data('type');
61          if (type === 'M') {
62              const id = $(this).data('id');
63              let payload = {};
64              payload[id] = this.checked;
65
66              post(payload);
67          }
68      });
69
70      // change tag - numeric
71      $(document.body).on('change', '.number-tag', function () {
72          const type = $(this).data('type');
73          if (type === 'M') {
74              const val = $(this).val();
75              const id = $(this).data('id');
76              let payload = { [id]: id.includes('R') ? parseFloat(val) : parseInt(val) };
77
78              post(payload);
79          }
80      });
81
82      // register new tag
83      $(document.body).on('submit', '#new-tag-form', function (e) {
84          e.preventDefault();
85          $('#address').removeClass('is-invalid')
86          const format = $('#format').val();
87          let address = $('#address').val();
88          let value;
89
90          //Validate forms
91          if (format === 'boolean') {
92              const userKeyRegExp = /^\d+\.[0-7]{1}?$/;
93              const valid = userKeyRegExp.test(address);
94              if (!valid) { // validate bit address
95                  $('#address').addClass('is-invalid')
96                  return false;
97              }
98              value = false;
99          } else {
100             const userKeyRegExp = /^[B,R]\d+?$/;
101             const valid = userKeyRegExp.test(address);
102             if (!valid) { // validate word address
103                 $('#address').addClass('is-invalid')
104                 return false;
105             }
106             value = 0;
107         }
108
109         address = $('#type').val() + address;
110         let payload = {};
111         payload[address] = value;
112
113         post(payload);
114         $('#address').val('');
115     });
116
```

```
117     /*
118     * Functions
119     */
120     //get data from firebase
121     function get() {
122         const doc = db.collection('tags_s7');
123         $('.tag-list').html('Loading...');
124         doc.onSnapshot(docSnapshot => {//getting realtime changes
125             $('.tag-list').html('Loading...');
126             const tag_list = docSnapshot.docs.map((doc) => ({
127                 id: doc.id,
128                 ...doc.data(),
129             }));
130             let object = {};
131             if (tag_list.length === 0) {
132                 firstLoad = true;
133             }
134             let tagsHtml = '';
135 
136             //render TAGS
137             tag_list.forEach(element => {
138                 object[element.address.replace('D', 'R')] = element.value;
139                 if (element.format === "boolean") {
140                     tagsHtml += `
141                         <div class="form-check form-switch">
142                             <input class="form-check-input bool-tag" value="1" $
                                {element.type !== 'M' ? 'disabled' : ''} type="checkbox" $
                                {element.value ? 'checked' : ''} data-type="${element.type}
                                " data-id="${element.id}" id="checkbox_${element.id}">
143                             <label class="form-check-label" for="checkbox_${element.id}
                                ">${element.id}</label>
144                         </div>
145                     `;
146                 } else {
147                     tagsHtml += `
148                         <div class="input-group mt-1">
149                             <span class="input-group-text">${element.address}</span>
150                             <input type="text" class="form-control number-tag" $
                                {element.type !== 'M' ? 'disabled' : ''} value="${Math.round
                                (element.value * 1000) / 1000}" id="input_${element.id}"
                                data-type="${element.type}" data-id="${element.id}"/>
151                         </div>
152                     `;
153                 }
154             });
155             $('.tag-list').html(tagsHtml);
156 
157             conn.removeItems(); //remove tags
158             var tags_label = Object.keys(object);
159             conn.addItems(tags_label);// add tags
160 
161             var changedTags = getChangedTags(object);
162             var labels = Object.keys(changedTags);
163             var values = Object.values(changedTags);
164 
165             if (CONNECTED && labels.length > 0 && values.length > 0) {
166                 conn.writeItems(labels, values, valuesWritten);// write tags
167             }
168             console.log('changedTags :>> ', changedTags);
169 
170         }, err => {
171             console.log(`Encountered error: ${err}`);
172         });
173     }
```

```
117     /*
118     * Functions
119     */
120     //get data from firebase
121     function get() {
122         const doc = db.collection('tags_s7');
123         $('.tag-list').html('Loading...');
124         doc.onSnapshot(docSnapshot => {//getting realtime changes
125             $('.tag-list').html('Loading...');
126             const tag_list = docSnapshot.docs.map((doc) => ({
127                 id: doc.id,
128                 ...doc.data(),
129             }));
130             let object = {};
131             if (tag_list.length === 0) {
132                 firstLoad = true;
133             }
134             let tagsHtml = '';
135 
136             //render TAGS
137             tag_list.forEach(element => {
138                 object[element.address.replace('D', 'R')] = element.value;
139                 if (element.format === "boolean") {
140                     tagsHtml += `
141                         <div class="form-check form-switch">
142                             <input class="form-check-input bool-tag" value="1" $
                                {element.type !== 'M' ? 'disabled' : ''} type="checkbox" $
                                {element.value ? 'checked' : ''} data-type="${element.type}
                                " data-id="${element.id}" id="checkbox_${element.id}">
143                             <label class="form-check-label" for="checkbox_${element.id}
                                ">${element.id}</label>
144                         </div>
145                     `;
146                 } else {
147                     tagsHtml += `
148                         <div class="input-group mt-1">
149                             <span class="input-group-text">${element.address}</span>
150                             <input type="text" class="form-control number-tag" $
                                {element.type !== 'M' ? 'disabled' : ''} value="${Math.round
                                (element.value * 1000) / 1000}" id="input_${element.id}"
                                data-type="${element.type}" data-id="${element.id}"/>
151                         </div>
152                     `;
153                 }
154             });
155             $('.tag-list').html(tagsHtml);
156 
157             conn.removeItems(); //remove tags
158             var tags_label = Object.keys(object);
159             conn.addItems(tags_label);// add tags
160 
161             var changedTags = getChangedTags(object);
162             var labels = Object.keys(changedTags);
163             var values = Object.values(changedTags);
164 
165             if (CONNECTED && labels.length > 0 && values.length > 0) {
166                 conn.writeItems(labels, values, valuesWritten);// write tags
167             }
168             console.log('changedTags :>> ', changedTags);
169 
170         }, err => {
171             console.log(`Encountered error: ${err}`);
172         });
173     }
```

APENDICE B – CÓDIGO HTML DO CONNECTOR

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>PLC Connector</title>
    <link rel="stylesheet" href="./node_modules/bootstrap/dist/css/bootstrap.min.css" />
  </head>
  <body>
    <div class="container">
        <div class="row">
            <div class="col-12 my-2 header">
                <div class="row">
                    <h4>Connection</h4>
                    <div class="col-md-3">
                        <label for="">PLC address</label>
                        <input type="text" class="form-control" id="plc_address"
                        value="192.168.0.100" placeholder="PLC address">
                    </div>
                    <div class="col-md-3">
                        <label for="">PLC rack</label>
                        <input type="text" class="form-control" id="plc_rack" value="0"
                        placeholder="PLC rack">
                    </div>
                    <div class="col-md-3">
                        <label for="">PLC slot</label>
                        <input type="text" class="form-control" id="plc_slot" value="2"
                        placeholder="PLC slot">
                    </div>
                    <div class="col-md-3 d-flex align-items-end pt-1">
                        <button class="btn btn-success" id="connect">Connect</button>
                        <button class="btn btn-danger d-none ms-2"
                        id="disconnect">Disconnect</button>
                    </div>
                </div>
                <hr>
                <div class="alert alert-danger d-none" id="status" role="alert"></div>
            </div>
            <div class="col-6">
                <h4>New tag</h4>
                <form id="new-tag-form">
                    <div class="mt-2">
                        <label for="">Type</label>
                        <select name="" id="type" required class="form-select">
                            <option value="I">Input</option>
                            <option selected value="M">Memory</option>
                            <option value="Q">Output</option>
                        </select>
                    </div>
                    <div class="mt-2">
                        <label for="">Format</label>
                        <select id="format" required class="form-select">
                            <option selected value="boolean">Boolean</option>
                            <option value="number">Number</option>
                        </select>
                    </div>
                    <div class="mt-2">
                        <label for="">Address</label>
                        <input type="text" required class="form-control" id="address"
                        placeholder="Address">
                    </div>
                    <button type="submit" class="btn btn-success mt-2" id="save">Save</
                    button>

                </form>
            </div>
            <div class="col-6">
                <h4>Tags</h4>
                <div class="tag-list"></div>
            </div>
        </div>
    </div>
  </body>
</html>
```

APENDICE C – CÓDIGO TYPESCRIPT DO APLICATIVO

```typescript
import { Component, OnInit, ViewChild } from '@angular/core';
import { Firestore } from "../firestore.class";
import { AngularFirestore } from '@angular/fire/compat/firestore';
import { IonModal, ToastController } from '@ionic/angular';

interface Tag {
  address: string,
  format: string,
  value: number | boolean,
  type: string
}

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage extends Firestore implements OnInit {
  @ViewChild(IonModal) modal: IonModal;
  newTag: Tag = {
    address: '',
    type: 'M',
    format: 'boolean',
    value: false,
  }
  tags: Tag[] = [];
  constructor(db: AngularFirestore, private toastController: ToastController) {
    super(db);
  }

  ngOnInit() {
    this.getAll()
      .subscribe((tags: any) => {
        tags.forEach((element: any) => {
          if (element.format === 'number' && element.address.includes('R')) {
            element.value = Math.round(element.value * 1000) / 1000;
          }
        });
        this.tags = tags;
      });
  }

  async presentToast(message: string) {
    const toast = await this.toastController.create({
      message: message,
      duration: 3000,
      buttons: [
        {
          text: 'Dismiss',
          role: 'cancel',
        }
      ]
    });

    await toast.present();

  }
```

```
58
59    onChange(item: any, event: any) {
60      if (item.type === 'M') {
61        this.setItem({ ...item, value: item.format === 'number' && item.address.includes('D') ? parseInt
62      }
63    }
64
65    cancel() {
66      this.modal.dismiss(null, 'cancel');
67    }
68
69    confirm(): any {
70      if (!this.newTag.address) {
71        this.presentToast('Insert tag address');
72        return false;
73      }
74      if (this.newTag.format == 'boolean') {
75        const userKeyRegExp = /^\d+\.[0-7]{1}?$/;
76        const valid = userKeyRegExp.test(this.newTag.address);
77        if (!valid) { // validate bit address
78          this.presentToast('Invalid address');
79          return false;
80        }
81        this.newTag.value = false;
82      } else {
83        const userKeyRegExp = /^[B,R]\d+?$/;
84        const valid = userKeyRegExp.test(this.newTag.address);
85        if (!valid) { // validate word address
86          this.presentToast('Invalid address');
87          return false;
88        }
89        this.newTag.value = 0;
90      }
91      this.setItem({ ...this.newTag, address: this.newTag.type + this.newTag.address }).then(() => {
92        this.newTag = { // reset newTag
93          address: '',
94          type: 'M',
95          format: 'boolean',
96          value: 0,
97        }
98        this.modal.dismiss('', 'confirm');
99      });
100   }
101
102   onDelete(id: string) {
103     this.delete(id);
104   }
105 }
106
```

APENDICE D – CÓDIGO HTML DO APLICATIVO

src > app > home > home.page.html > ion-header

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Tags
    </ion-title>
    <ion-buttons slot="end">
      <ion-button id="open-modal" expand="block">
        <ion-icon name="add"></ion-icon>
      </ion-button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>

<ion-content [fullscreen]="false">

  <ion-list>
    <ion-item-sliding *ngFor="let tag of tags; index as i">
      <ion-item>
        <ion-label>{{tag.address}}</ion-label>
        <ion-toggle *ngIf="tag.format === 'boolean'" [(ngModel)]="tag.value" [disabled]="tag.type !==
          (ngModelChange)="onChange(tag,$event)"></ion-toggle>
        <ion-input type="number" style="text-align: right;" *ngIf="tag.format === 'number'" [(ngModel)
          [disabled]="tag.type !== 'M'" (ngModelChange)="onChange(tag,$event)" debounce="2000"></ion-i
      </ion-item>
      <ion-item-options>
        <ion-item-option (click)="onDelete(tag.address)" color="danger">
          <ion-icon slot="icon-only" name="trash"></ion-icon>
        </ion-item-option>
      </ion-item-options>
    </ion-item-sliding>
  </ion-list>

</ion-content>
<ion-modal trigger="open-modal">
  <ng-template>
    <ion-header>
      <ion-toolbar>
        <ion-buttons slot="start">
          <ion-button (click)="cancel()">
            <ion-icon name="close-outline"></ion-icon>
          </ion-button>
        </ion-buttons>
        <ion-title>New tag</ion-title>
        <ion-buttons slot="end">
          <ion-button (click)="confirm()" [strong]="true">
            <ion-icon name="checkmark-outline"></ion-icon>
          </ion-button>
        </ion-buttons>
      </ion-toolbar>
    </ion-header>
    <ion-content class="ion-padding">
      <ion-item>
        <ion-label position="stacked">Type</ion-label>
        <ion-select placeholder="Select type" [(ngModel)]="newTag.type">
          <ion-select-option value="I">Input</ion-select-option>
          <ion-select-option value="M">Memory</ion-select-option>
          <ion-select-option value="Q">Output</ion-select-option>
        </ion-select>
      </ion-item>
      <ion-item>
        <ion-label position="stacked">Format</ion-label>
        <ion-select placeholder="Select format" [(ngModel)]="newTag.format">
          <ion-select-option value="boolean">Boolean</ion-select-option>
          <ion-select-option value="number">Number</ion-select-option>
        </ion-select>
      </ion-item>
      <ion-item>
        <ion-label position="stacked">Address</ion-label>
        <ion-input type="text" [placeholder]=" newTag.format === 'boolean' ? '0.0' : 'B2'"
          [(ngModel)]="newTag.address"></ion-input>
      </ion-item>
    </ion-content>
  </ng-template>
</ion-modal>
```