Pró-Reitoria de Graduação Presencial.
Coordenação do Curso de Engenharia Mecatrônica.

Artigo apresentado como um dos pré-requisitos para a obtenção do grau de bacharel em Engenharia Mecatrônica, 2021/02.

DESENVOLVIMENTO DE UM SISTEMA IOT SERVERLESS PARA A INDÚSTRIA 4.0

Gustavo de Souza Cruz (gustavo.cruz@souunit.com.br); Raphael de Souza Pereira (raphael.souza99@souunit.com.br); Profº Me. Roberto Felipe Andrade Menezes (rmenezeseng@gmail.com)

RESUMO

A instalação e monitoramento da infraestrutura dos servidores, que possibilitam o controle e leitura de sensores e atuadores, caracteriza um dos principais consumidores de recursos na criação de uma indústria de baixo a alto porte. Além do espaço necessário para comportar esta infraestrutura há ainda o consumo de energia, monitoramento de backups e a velocidade do sistema. A computação em nuvem remove todos esses processos e ainda entrega características como escalabilidade automática por demanda. Para a Indústria 4.0 e seus dispositivos modelados para a Internet das Coisas torna-se natural o controle e leitura dos dados com a facilidade de acesso global e a segurança de ter o sistema sempre operante através da gerência automática dos servidores por parte da plataforma em nuvem e não na indústria. Com isso, este artigo tem como objetivo apresentar uma proposta de como é possível ter acesso a um sistema de controle industrial através da utilização de um banco de dados NoSQL com Firebase e da computação em nuvem. Os resultados demonstraram que, com a implementação de um modelo Serverless dentro da indústria, a central de operações pode voltar a sua atenção para funções mais importantes dentro de uma empresa, ao invés de se preocuparem com o gerenciamento e a operação de servidores, já que eles não serão mais gerenciados por eles, e sim pelos provedores da nuvem.

Palavras-chave: *Indústria 4.0. Internet das Coisas. Computação em Nuvem. NoSQL. Firebase.*

ABSTRACT

The installation and monitoring of server infrastructure, which enable the control and reading of sensors and actuators, characterizes one of the main consumers of resources in the creation of a small to large-scale industry. In addition to the space needed to support this infrastructure, there is also energy consumption, backup monitoring and system speed. Cloud computing removes all of these processes and still delivers features like automatic scalability on demand. For Industry 4.0 and its devices modeled for the Internet of Things, it is natural to control and read data with the ease of global access and the security of having the system always operational through the automatic management of servers by the cloud platform and not in the industry. Thus, this article aims to present a proposal on how it is possible to have access to an industrial control system through the use of a NoSQL database with Firebase and cloud computing. The results showed that, with the implementation of a Serverless model within the industry, the operations center can turn its attention to more important functions within a company, instead of being concerned with the management and operation of servers, as they will no longer be managed by them, but by cloud providers.

Keywords*: Industry 4.0. Internet of Things. Cloud computing. NoSQL. Firebase.*

1 INTRODUÇÃO

A crescente necessidade de se produzir mais e melhor tem levado o mundo industrial a passar por várias revoluções. A primeira Revolução Industrial teve início na Inglaterra no século XVIII. Essa se caracterizou por diversas descobertas que favoreceram a expansão das indústrias, como por exemplo: as invenções do tear mecânico e da máquina a vapor, que resultaram na mecanização dos processos. Agora, cerca de três séculos depois, é a Indústria 4.0 que promete gerar um impacto profundo na sociedade ao introduzir soluções tecnológicas sofisticadas, entre elas, podemos citar algumas que servem de pilar para o desenvolvimento da indústria 4.0, como a Internet das coisas (*Internet of Things* - IoT) e a Computação em Nuvem.

A Indústria 4.0, também chamada de Quarta Revolução Industrial, se destaca pela evolução na comunicação entre sistemas e equipamentos, disponibilizando para os usuários diversas informações úteis para o gerenciamento e aperfeiçoamento dos sistemas de produção. É possível monitorar e controlar todas as ferramentas da produção, melhorando a produtividade e qualidade, através de um melhor ajuste das máquinas (MCKINSEY, 2015). Com isso, pode-se dizer que a Indústria 4.0 tem um impacto significativo na produtividade, pois aumenta a eficiência do uso de recursos e no desenvolvimento de produtos em larga escala.

Segundo Magrani (2018), a IoT surge como a ideia de conectar qualquer dispositivo que gere informações e possa se conectar a um serviço de nuvem. No âmbito industrial, a IoT une máquinas inteligentes e análise avançada de dados para desenvolver sistemas mais rápidos e eficientes, que são capazes de monitorar, coletar, mudar, analisar e entregar dados e informações fundamentais para que as indústrias tomem decisões rápidas e certeiras. No Brasil e no mundo já estão sendo implementadas aplicações como controle de estoque, controle de máquinas, otimização de operações, acompanhamento em tempo real da entrega de matéria-prima, monitoramento de equipamentos, dentre várias outras.

O objetivo deste trabalho é propor o desenvolvimento de um sistema IoT *Serverless* para a Indústria 4.0. Esse estudo visa também demonstrar que, com a implementação de um modelo Serverless dentro da indústria no lugar de um modelo tradicional, organizações de todos os tamanhos e em todos os setores da indústria poderiam se beneficiar, já que todos possuem a mesma preocupação em comum: a redução de gastos desnecessários. Dentre esses gastos desnecessários, além do consumo de energia e o de matéria prima, estão os recursos de hardware que ficam ociosos na maior parte do tempo, o que não acontece nas arquiteturas serverless, já que esses recursos são provisionados pelo provedor do serviço.

Este trabalho está dividido em 5 seções, começando por esta introdução, que tem por objetivo apresentar o trabalho, logo em seguida na seção de fundamentação teórica são apresentados os principais conceitos necessários para o entendimento do trabalho. Após a apresentação destes conceitos, está a seção de metodologia, onde são apresentadas as técnicas e ferramentas utilizadas na execução do trabalho, assim como o fluxo a ser seguido. Os resultados obtidos são demonstrados e analisados na seção de resultados e discussões. Finalizando, a seção de conclusão apresenta as conclusões obtidas com a realização do presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

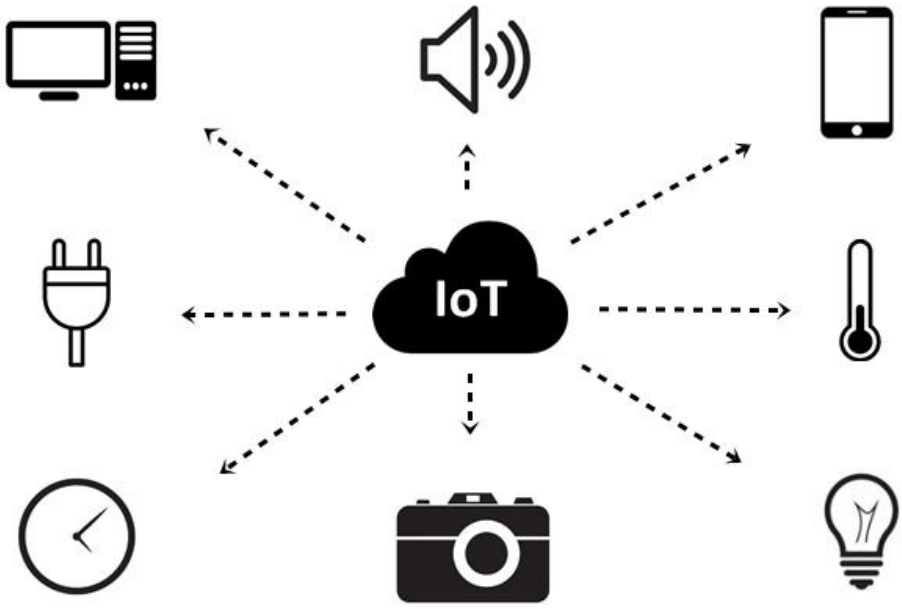
Neste capítulo serão apresentados os conceitos básicos necessários para o completo entendimento do trabalho desenvolvido, contemplando a definição das principais áreas de estudo, como a IoT, a Computação em Nuvem e o Banco de

dados.

2.1 Indústria 4.0 e IoT

Segundo Magrani (2018), a IoT pode ser entendida como um ambiente de objetos físicos interconectados com a internet por meio de sensores pequenos e embutidos. Essa interconexão pode ser feita através de tecnologias como Wi-fi (*Wireless fidelity* - Fidelidade sem fio) e Bluetooth. Esses objetos físicos aliados a dispositivos de computação podem facilitar o processo de transmissão e recebimento de dados em tempo real, já que são capazes de interagir entre si, bem como compartilhar dados e tomar decisões de maneira autônoma. Na Figura 1 é mostrada uma ilustração do esquema da IoT.

Figura 1: Ilustração do esquema da IoT



Fonte: Autoria própria.

A tecnologia IoT se popularizou através dos avanços da computação e áreas correlacionadas, como microeletrônica, comunicação e sensoriamento, tendo recebido bastante atenção da indústria, devido ao seu potencial de uso nas mais diversas áreas das atividades humanas. Outro fator importante que incentivou a ampla difusão da IoT nas indústrias, foi devido ao fato dos aplicativos baseados nessa tecnologia serem ainda menores, mais baratos e com um consumo de energia ainda mais baixo (FACCIONI FILHO, 2016). A aplicação da IoT nas indústrias acelera o amadurecimento da Quarta revolução Industrial, permitindo assim que com a sua utilização, as indústrias se tornem cada vez mais inteligentes.

A Indústria 4.0, que faz parte da Quarta Revolução Industrial, é a continuação do aperfeiçoamento das máquinas, que começou na Primeira Revolução Industrial. O conceito de indústria 4.0 muda o conceito de produção centralizada para produção descentralizada, e está fortemente ligado a sistemas embarcados e sistemas ciberfísicos. Estes últimos são as tecnologias que permitem interligar o mundo virtual com o mundo físico. Conforme o avanço das tecnologias ao decorrer do anos, a tendência é que as fábricas cada vez mais se adequem ao conceito da Indústria 4.0, tornando-se altamente autônomas e eficientes.

2.2 Modelo OSI

Segundo Torres (2004), para facilitar a interconexão de sistemas de computadores, a ISO (Organização Internacional de Normalização) desenvolveu um modelo de referência chamado OSI (Interconexão de Sistemas Abertos), para que fabricantes pudessem criar protocolos a partir desse modelo.

O Modelo OSI é composto por 7 camadas abstratas, sendo que cada uma delas realiza determinadas funções e se comunicam com as camadas acima e abaixo delas. Essas camadas podem ser vistas como um conjunto de regras que definem como será realizada a operação e a troca das informações entre dois sistemas. As camadas são: Aplicação, Apresentação, Sessão, Transporte, Rede, Dados e Física (TORRES, 2004). Na Figura 2 é mostrada uma ilustração das camadas do modelo de referência OSI.

Figura 2: Camadas do modelo de referência OSI

Fonte: Autoria própria.

A primeira camada do modelo OSI, é a Camada Física. Essa camada é responsável pela transmissão e recepção do fluxo de bits brutos não estruturados através da parte física, como cabos Ethernet e comutadores. Já a segunda camada é chamada de Camada de Dados. Essa camada é responsável por receber os dados do meio físico e realizar uma análise que busca identificar erros e corrigi-los, além de ser responsável também pelo controle do fluxo de transmissão dos dados (TANENBAUM, 2003).

A terceira camada, conhecida como Camada de Rede, é responsável pelo endereçamento dos dispositivos de rede, sendo responsável pela decisão de que caminho físico os dados devem percorrer da origem ao destino, com base nas condições da rede e na prioridade do serviço. A quarta camada é chamada de Camada de Transporte, e é responsável por garantir o envio e o recebimento dos pacotes vindos da Camada de Rede. Ela gerencia o transporte dos pacotes para garantir o sucesso no envio e no recebimento de dados. Protocolos muito comuns dessa camada são os protocolos TCP e UDP (TANENBAUM, 2003).

A camada cinco é chamada de Camada de Sessão, e é responsável por exercer o controle de quando a comunicação entre duas máquinas deve começar ou terminar, além de também oferecer suporte para as sessões. Já a camada seis é conhecida como Camada de Apresentação, e é responsável por fazer a tradução dos dados para que a próxima camada os use, o que inclui a conversão de códigos para caracteres, a conversão e compactação dos dados, além da criptografia desses dados, caso seja

necessário (TANENBAUM, 2003).

A sétima e última camada leva o nome de Camada de Aplicação, e tem como foco ser a porta de entrada da rede, dando aos usuários acesso aos serviços dessa rede, como aplicativos de mensagens instantâneas, servidores de e-mails, navegadores web, entre outras possibilidades. É aqui que ficam os protocolos mais conhecidos, como HTTPS e FTP (TANENBAUM, 2003).

2.3 Microcontrolador

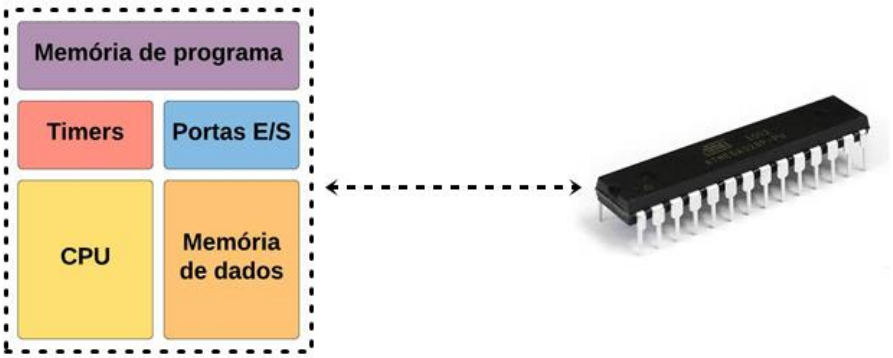
Segundo Kerschbaumer (2018), microcontroladores são circuitos integrados que possuem em seu interior todos os componentes necessários ao seu funcionamento dependendo unicamente da fonte de alimentação externa.

Pode-se dizer que os microcontroladores são computadores de um único chip. Ele é utilizado em sistemas embarcados, e vem com a possibilidade de ser programado para desempenhar tarefas bem específicas, desde sistemas mais complexos, como um sistema de controle de supervisão industrial, até sistemas mais simples, como um sistema de alarme. Essa programação é feita a partir de softwares preparados para fazer a conversão da linguagem de programação utilizada para a linguagem de máquina.

O processador do microcontrolador é a sua parte principal. Ele é responsável por processar as informações de maneira lógica, usando um programa próprio. Além do processador, outro componente que faz o sistema funcionar é a memória de programa. Ela é responsável por guardar instruções ou programas que comandam o microcontrolador. Já as memórias de dados guardam informações temporárias, coletadas durante o uso do microcontrolador, para serem usadas quando for preciso.

Outra parte importante são os temporizadores, já que muitas das funções do microcontrolador devem ser controladas com precisão temporal, é essa parte dele que faz a contagem. As portas de entrada e de saída são as responsáveis por conectar o microcontrolador com as informações que chegam e que saem dele para o mundo, e vice-versa. (KERSCHBAUMER, 2018). Na Figura 3 é mostrada uma ilustração dos componentes de um microcontrolador.

Figura 3: Componentes de um microcontrolador



Fonte: Autoria própria.

Com muita frequência são vistos microcontroladores atrelados a placas de desenvolvimento, que são placas de circuito com vários periféricos, de modo a facilitar o uso de suas funções. Dentre as mais comuns no mercado, encontra-se o Arduino, o ESP e o Raspberry Pi. A grande diferença entre o Arduino e o ESP consiste principalmente no fato de que a grande maioria dos microcontroladores da família ESP têm conexão Wi-Fi. Já em relação ao Raspberry Pi, a grande diferença em

relação aos demais é que ele possui mais memória RAM e de armazenamento, maior velocidade de processamento e a capacidade de realizar múltiplas tarefas ao mesmo tempo (DE OLIVEIRA, 2017). Outro exemplo de uma placa de desenvolvimento é o NodeMCU, que combina o chip ESP8266 ou então o ESP32, com a pilha do protocolo TCP/IP integrada, que permite que o usuário possa implementar o acesso a rede WiFi com qualquer microcontrolador,

2.4 Interface de programação de aplicação

A API (*Application Programming Interface* - Interface de Programação de Aplicação) é uma interface de comunicação que um software oferece para que outros aplicativos acessem os seus recursos, dados e funções, possibilitando assim que eles adaptem as suas funcionalidades da melhor forma a fim de atender às suas necessidades. É possível conectar APIs, assim como criar aplicações que fazem uso dos seus dados ou funcionalidades, usando uma plataforma de integração distribuída que ligue todos os elementos, incluindo diferentes bancos de dados, aplicações nativas em nuvem e dispositivos de IoT (NEUMANN; LARANJEIRO; BERNARDINO, 2018). Na Figura 4 é mostrada uma ilustração da representação do funcionamento de uma API.

Figura 4: Representação do funcionamento de uma API

Fonte: Autoria própria.

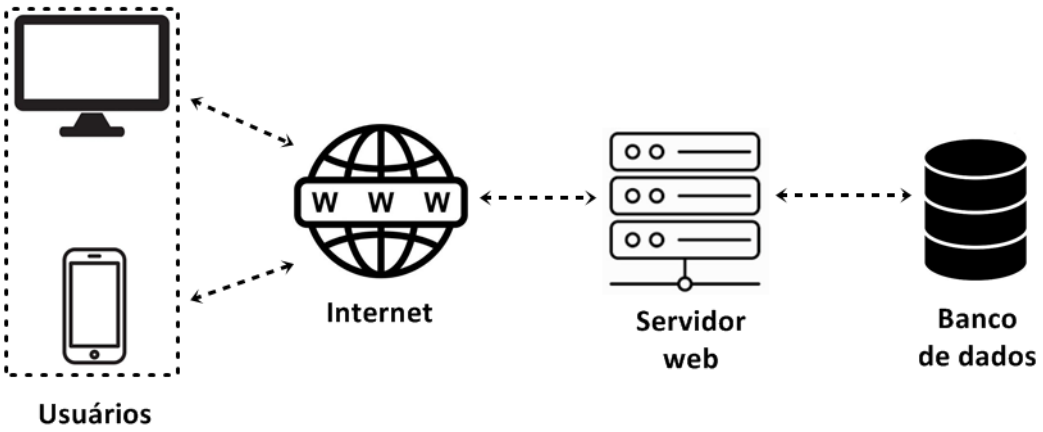
Hoje em dia existe uma enorme quantidade de APIs disponíveis para serem utilizadas nos mais diversos tipos de softwares. Como exemplos de sistemas que oferecem APIs para pagamentos, existe o PagSeguro e o Paypal. Ambos vão oferecer uma infinidade de formas de pagamentos para determinada aplicação. Em relação às APIs de Localização, o Google Maps é o exemplo mais popular. Por meio de seu código original, muitos outros sites e aplicações utilizam os seus mapas em seus serviços. Uma vasta quantidade de redes sociais também oferecem APIs, como Facebook, Twitter e Instagram. Todas elas irão oferecer funcionalidades diversas, como obter informações úteis sobre o usuário e criar opção de login utilizando o perfil da rede social.

Como a rede de comunicações mais utilizada é a Internet, a maioria das APIs são projetadas com base em padrões da web. As APIs web possuem uma especificação de protocolo com o objetivo de ajudar a padronizar a troca de informações, esse protocolo é chamado de SOAP (Protocolo Simples de Acesso a Objetos). Outra especificação é o REST (Transferência Representacional de Estado), que diferentemente do SOAP, é um estilo de arquitetura. APIs web que adotam as restrições de arquitetura da REST são chamadas de APIs RESTful. Para que uma API seja considerada RESTful, ela precisa estar em conformidade com algumas restrições de arquitetura, que embora essas restrições possam parecer excessivas, são muito mais simples do que um protocolo prescrito. Por esse motivo, as APIs RESTful estão se tornando cada vez mais comuns do que as APIs SOAP (RICHARDSON, 2018).

2.5 Banco de dados

Segundo Date (2004), um banco de dados é uma coleção de dados operacionais armazenados, sendo usados pelos sistemas de aplicação de uma determinada organização. Um banco de dados é usualmente mantido e acessado por meio de um software conhecido como Sistema Gerenciador de Banco de Dados (SGBD), que permite a definição, a construção e o manejo de um Banco de dados para diversas aplicações. Na Figura 5 é mostrada uma ilustração do esquema do funcionamento de um Banco de dados

Figura 5: Esquema do funcionamento de um Banco de dados



Fonte: Autoria própria.

Hoje em dia existem diversos tipos de bancos de dados, e cada um deles é adequado para atender a uma necessidade. O padrão mais conhecido é o banco de dados relacional, baseado na linguagem SQL (Linguagem de Consulta Estruturada). Ele é muito usado em sistemas de gestão de empresas e em sistemas de relacionamento com o cliente. Isso porque nesse tipo de banco, os dados são estruturados em tabelas, sendo que as linhas e colunas se relacionam. Também existem os chamados bancos de dados não relacionais, que são baseados em uma linguagem NoSQL (Não Somente SQL). Isso quer dizer que os dados não são completamente estruturados em tabelas, como é o caso dos serviços de armazenamento de imagens na web, por exemplo (ELMASRI; NAVATHE, 2016).

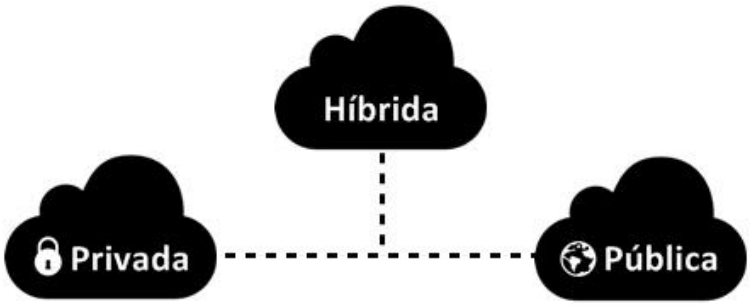
Por ser extremamente importante em uma enorme cadeia de negócios, principalmente aqueles relacionados com a tecnologia, os bancos de dados se difundiram rapidamente no mercado. Com a grande quantidade de dados coletados da IoT, as empresas hoje em dia têm acesso a mais dados do que nunca. Empresas agora podem usar bancos de dados que vão além do armazenamento de dados e de transações básicas para analisar grandes quantidades de dados de vários sistemas. Ao usar bancos de dados e outras ferramentas de computação, as organizações aproveitam dados que coletam para executar funções com mais eficiência e possibilitar uma melhor tomada de decisões.

2.6 Computação em nuvem

A Computação em Nuvem tem como conceito a possibilidade de acessar arquivos e executar diferentes tarefas pela internet, já que os dados já estão todos armazenados em servidores das empresas que prestam o serviço, acessíveis aos usuários pela rede mundial de computadores. Atualmente existem três tipos de Computação em Nuvem: Nuvem Pública, Nuvem Privada e Nuvem Híbrida. Na nuvem pública, os provedores de serviços terceirizados disponibilizam recursos e serviços

para seus clientes via internet. Os principais provedores de nuvem são Amazon AWS, Microsoft Azure e Google Cloud Platform. A nuvem privada também fornece recursos quase semelhantes aos da nuvem pública, mas os dados e serviços são gerenciados pela organização ou por terceiros apenas para a organização do cliente. Já uma nuvem híbrida combina nuvens públicas e privadas ligadas por uma tecnologia que permite que dados e aplicativos sejam compartilhados entre elas (TAURION, 2009). Na Figura 6 é mostrada uma ilustração dos tipos de computação em nuvem.

Figura 6: Tipos de computação em nuvem



Fonte: Autoria própria.

Serverless computing, ou computação sem servidores, é uma evolução da computação em nuvem. O *Serverless* é um modelo de execução onde o provedor de computação em nuvem funciona como o próprio servidor, gerenciando e alocando recursos computacionais. Esse novo modelo serverless de nuvem está se consolidando com rapidez no mercado por oferecer mais flexibilidade e economia do que serviços já consolidados. É uma arquitetura em que a execução de códigos é totalmente gerenciada por um provedor em nuvem, ao contrário do método tradicional de desenvolvimento de aplicativos e a instalação direta em servidores. Como exemplo de um serviço *serverless*, está o Firebase (HENDRICKSON, 2016).

Armazenar dados na Nuvem pode ser vantajoso, pois permite que o usuário proteja os seus dados caso aconteça um evento inesperado. Sendo assim, a criação de uma rotina de backup dos dados críticos na Nuvem é interessante, pois um incêndio ou desastre natural pode arrasar com toda uma empresa. Ter os backups armazenados apenas localmente pode ser inútil na hora da eventual necessidade de recuperação das informações.

3 METODOLOGIA

Neste capítulo são apresentadas as etapas e ferramentas que foram utilizadas no desenvolvimento deste trabalho e como o mesmo foi executado.

A ideia proposta por este trabalho é demonstrar como a IoT pode integrar uma indústria unificando as suas produções. O sistema proposto conta com dispositivos IoT localizados no chão de fábrica da indústria, responsáveis por fazer a captação de informações geradas pelas máquinas presentes na linha de produção.

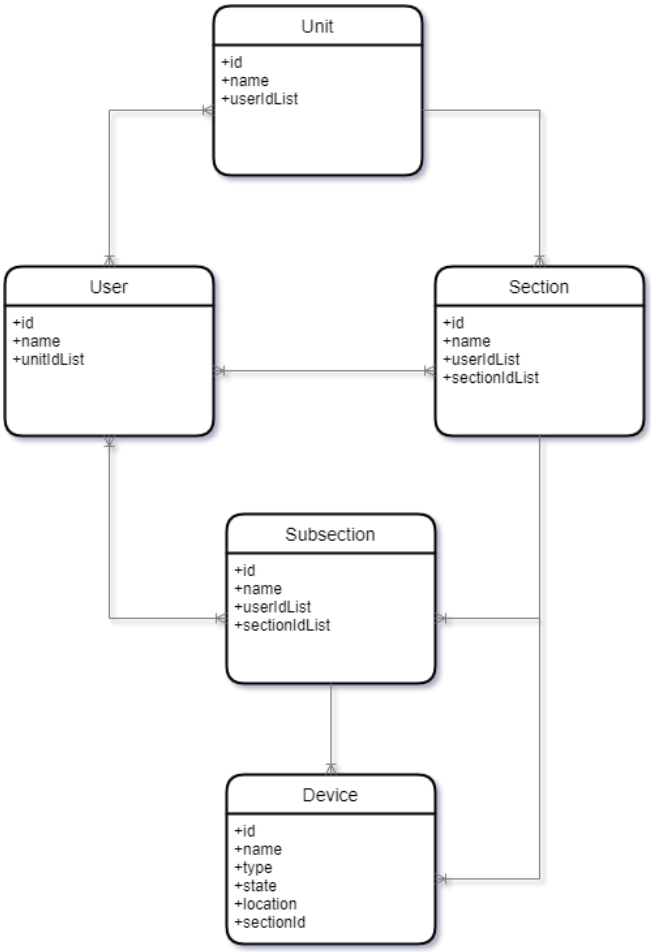
Após realizar a captação de dados, os dispositivos IoT enviam essas informações diretamente ao microcontrolador, onde após esses dados serem gerados, é preciso enviá-los para um banco de dados da central de operações. Para isso, é necessário criar uma API, onde os dados serão processados e posteriormente enviados para um banco de dados, onde precisará ser feita uma requisição de acesso.

3.1 Persistência de dados e segurança

A persistência de dados é feita na nuvem através do Firebase com estrutura de documentos cascateados e segurança de cima para baixo, onde apenas pessoas autorizadas podem acessar o documento atual, mesmo que documentos “pai” já tenham acesso.

A flexibilidade de um banco de dados NoSQL permite o crescimento ao longo do tempo de forma mais natural e centralizada. O acesso é dado adicionando individualmente usuário por usuário e apenas por pessoas com categoria suficiente para permitir acesso ao documento. Esta metodologia é adotada para aumentar o nível de segurança do sistema. Em casos de uma invasão ou vazamento de credenciais, a propagação do acesso será dada mais lenta pela falta de conhecimento do sistema e o processo demorado para adicionar o acesso. O esquema do banco pode ser visto na Figura 7.

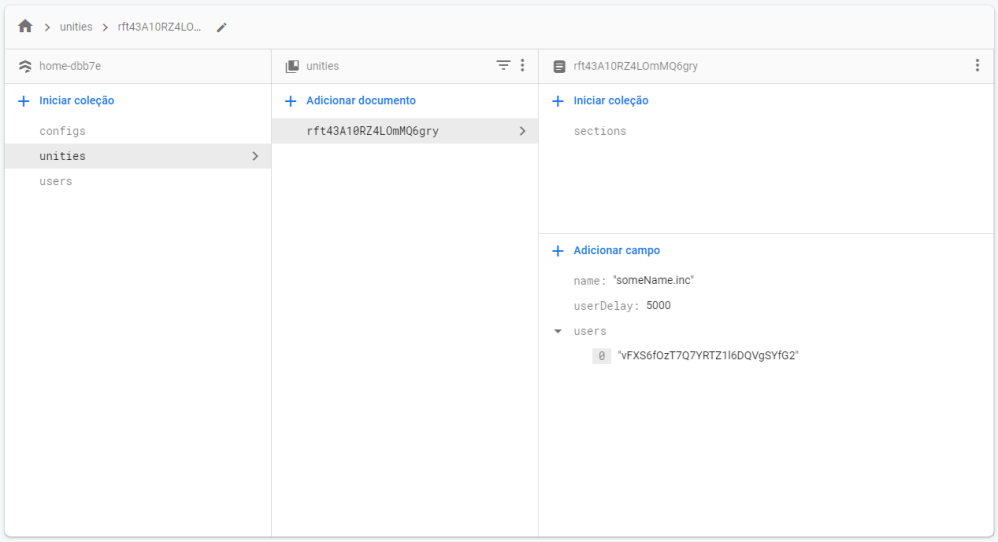
Figura 7: Esquema relacional do banco de dados



Fonte: Autoria própria.

A visualização no banco de dados é demonstrada na Figura 8 com as coleções principais do sistema no console do firestore.

Figura 8: Firestore console



Fonte: Autoria própria.

3.2 Conectividade

Para qualquer ação no banco de dados é necessário estar autenticado, seja o operador na sala de supervisão ou o dispositivo IoT que está lendo/escrevendo os dados do seu sensor na nuvem. O processo de autenticação pode ser visto na Figura 9 e é formado pelo login no sistema com um usuário que pertence à unidade organizacional.

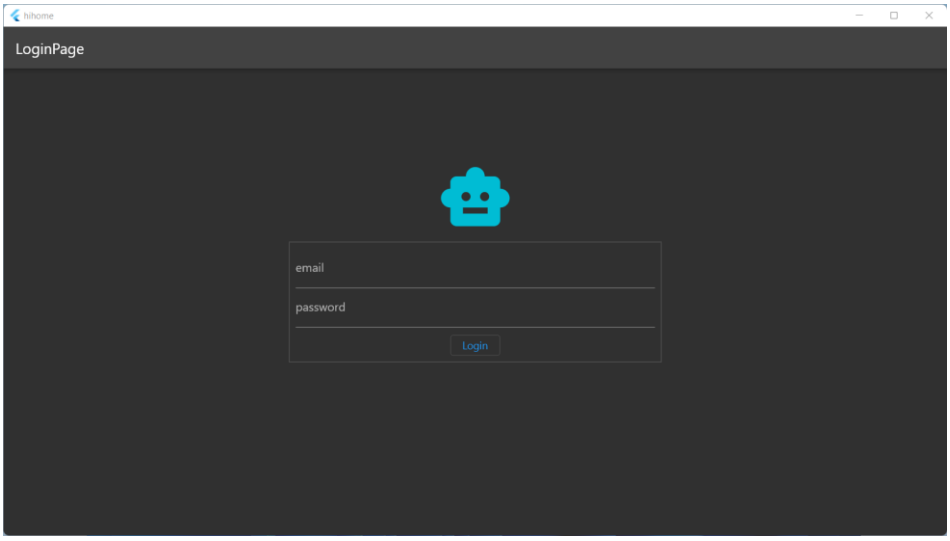
Figura 9: Fluxo de acesso aos dados

Fonte: Autoria própria.

Além do login no sistema, é necessário ter as devidas permissões para o dispositivo ou seção que está tentando acessar. Elas são configuradas através da adição individual dos usuários dentro do documento nos campos de usuários permitidos, seja para leitura, escrita e gerência do documento do dispositivo com os campos de nome, tipo, entre outros.

Para facilitar o uso e agilizar a prototipagem, foi utilizado o sistema do Material Design para o desenvolvimento das telas e ícones do app. É possível ver na Figura 10 a tela de login para o sistema de controle. O sistema de acesso implementado nesta fase do projeto é o de email e senha, mas outras formas de acesso são permitidas pelo firebase.

Figura 10: Tela de Login



Fonte: Autoria própria.

Tanto para o sistema de controle, como para os dispositivos IOT, a autenticação é feita através do sistema JWT com token onde o token é passado em toda requisição e na sua expiração é necessário logar novamente para atualizar o token.

Os dispositivos no campo necessitarão da criação individual ou em grupo de um usuário e suas credenciais de acesso para os mesmos poderem se autenticar no sistema e fazerem as operações de leitura/escrita. É recomendada a criação individual do usuário por medidas de segurança onde o vazamento destas credenciais limitaria o escopo a um único dispositivo com a segurança fragilizada e não toda a fábrica ou seção.

3.3 Middleware

Comparativamente ao transdutor que converte os sinais elétricos para uma rede industrial em uma “língua” específica, pode ser feita a utilização de um middleware para retirar da planta os dados através dos protocolos industriais presentes ou pelo próprio HTTP e entregar aos servidores através do protocolo HTTPS. Este controlador pode ser responsável por um único dispositivo ou malha (DE ALBUQUERQUE; THOMAZINI, 2020). Além de controladores industriais, podem ser middlewares desktops, mini computadores e raspberry pi’s.

Para o exemplo da malha é possível citar como exemplo o gateway SIMATIC IOT2050, visto na Figura 11, da linha de dispositivos conectados à internet da Siemens. Nesse caso é necessário ter o protocolo disponível ao controlador ou transdutor que repassará a informação à nuvem (SIEMENS, 2021).

Figura 11: Gateway SIMATIC IOT2050

Fonte: SIEMENS, 2021.

No projeto foi utilizado um computador, e o meio de comunicação com o dispositivo IOT, no caso o ESP8266, foi utilizado o protocolo HTTP para a comunicação com o microcontrolador em modo de servidor web.

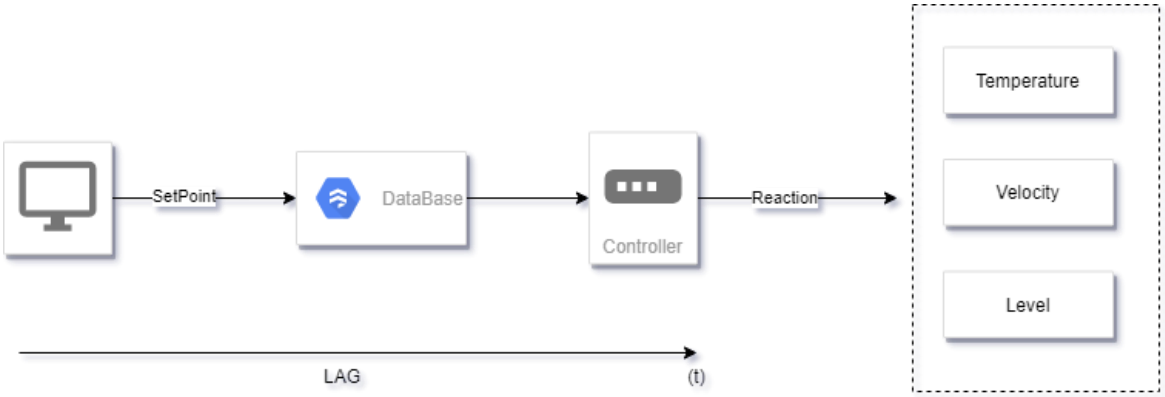
O processo de inicialização do dispositivo IoT é de logar no sistema para receber o token, atualizar o IP do dispositivo no documento na nuvem e inicializar o modo servidor web para receber as alterações através do middleware. Este, por sua vez, está utilizando a linguagem Python com a biblioteca do firestore para verificar as alterações e reagir de forma instantânea pelo canal de comunicação. Este canal atualiza o middleware com o IP atual do dispositivo IoT e então o middleware repassa, através de chamadas HTTP, os novos dados para o dispositivo IoT.

3.4 Variável Controlada

O controle do processo utiliza o valor da variável desejada (Setpoint) na nuvem para guiar os dispositivos na planta. Como por exemplo, é possível citar o valor de temperatura, o estado de um atuador pneumático e a velocidade de uma esteira. O Fluxo de controle pode ser visto na Figura 12.

O controlador em campo por sua vez tem o papel de ler essa variável e atuar no processo, seja ele através de um controle PID, On-Off, etc. Sua eficiência pode ser obtida através da latência, vista na Figura 12. Essa latência, também conhecida como *lag*, é o intervalo de tempo entre o início de uma atividade e o momento em que os efeitos desta se tornam aparentes. Este tempo dependerá da velocidade de conexão, quantidade de dados a serem lidos e o do poder de processamento do controlador ou dispositivo IoT.

Figura 12: Diagrama de lag do sistema



Fonte: Autoria própria.

3.5 Leitura da variável medida

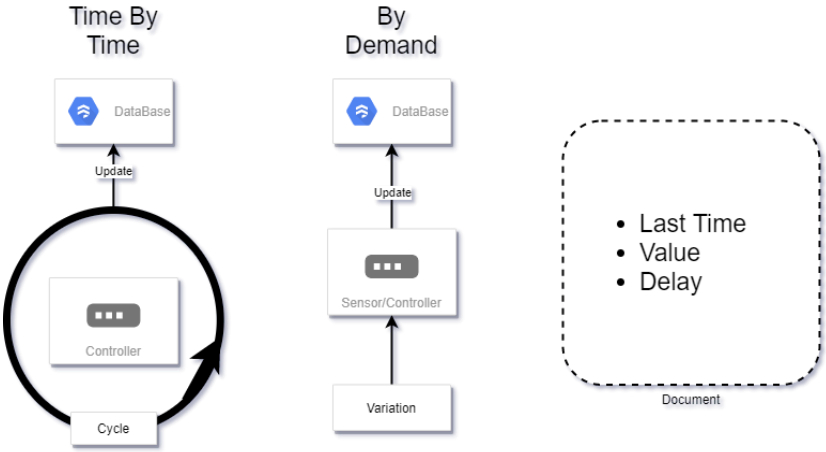
A leitura do processo ocorre através do valor atual no documento do banco de dados que representa a variável medida ou estado. Este por sua vez notifica os usuários conectados do lado da sala de controle e é atualizado através do controlador ou dispositivo IoT. A frequência de atualização pode ser feita de tempos em tempos ou quando houver uma atualização na variável medida.

A escolha do modo de operação pode variar entre as limitações de custos do sistema e uma maior precisão de leitura entre as variações da variável medida, sendo respectivamente a “By Demand” e a “Time by Time”. Os modos de leitura são exibidos na Figura 13.

Outros fatos importantes dentro do documento são:

- Horário da última atualização;
- Valor medido;
- Tempo entre atualizações (para o modo intermitente).

Figura 13: Diagrama de tipos de leitura e atualização



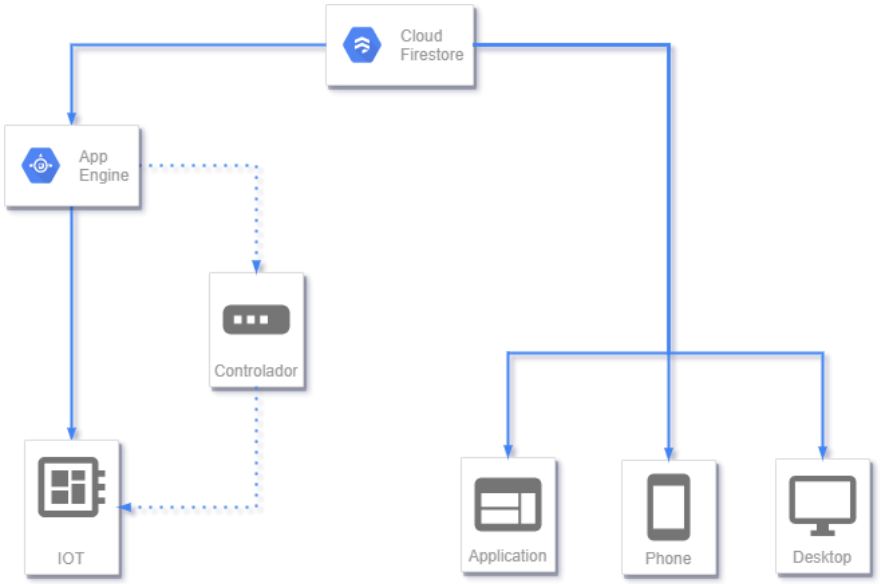
Fonte: Autoria própria.

3.6 Sistema de controle

A leitura e controle do sistema é feita através da alteração do SetPoint e da variável medida no banco de dados, especificamente no documento responsável pelo dispositivo. Estes dados guiam, respectivamente, o sistema de controle com valor da variável medida e o dispositivo com o setpoint que ele deve se adequar.

Como plataforma de controle foi desenvolvido um sistema supervisório com flutter pela sua versatilidade na geração de projetos em android, ios, web e desktop com o mesmo *Codebase* e assim atendendo ao objetivo de acessar o sistema não só por qualquer parte do mundo, mas em várias plataformas. O diagrama de controle pode ser visto na Figura 14 e ele demonstra o fluxo de dados pelo sistema entre o dispositivo e o controle.

Figura 14: Diagrama de controle

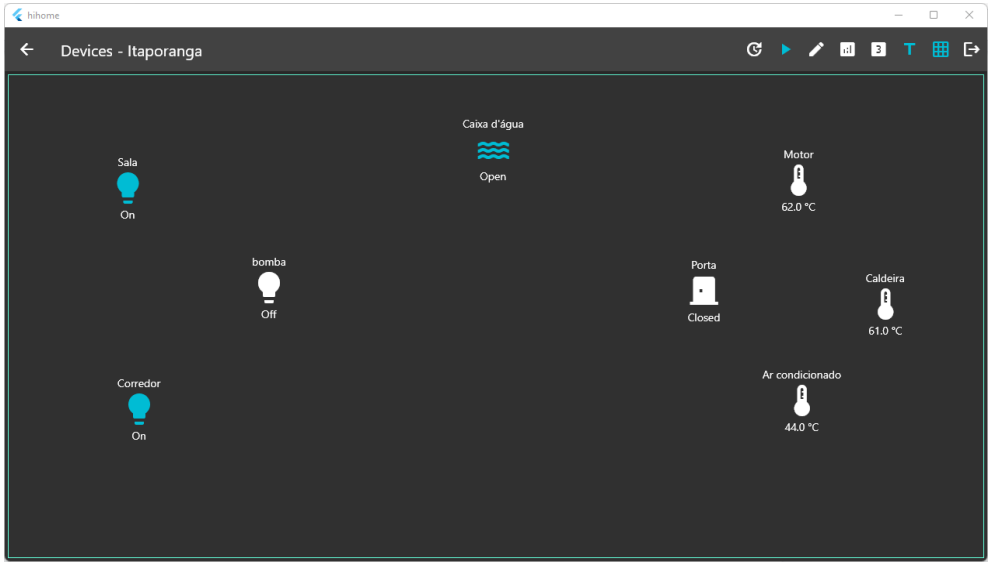


Fonte: Autoria própria.

A visualização do supervisório no modo de dispositivos pode ser vista na Figura 15 e existem alguns elementos na app bar para:

- Modo de visualização: altera entre dispositivos e visualização de seções.
- Modo de edição: alteração do valor, nome e tipo do dispositivo.
- Modo de execução: alteração do valor e visualização dos dados.
- Modo de análise: visualização dos logs do dispositivo escolhido.
- Tamanho dos ícones: o tamanho varia entre 4 opções.
- Exibição do nome do dispositivo.
- Logout: volta a tela de login para alteração do usuário.

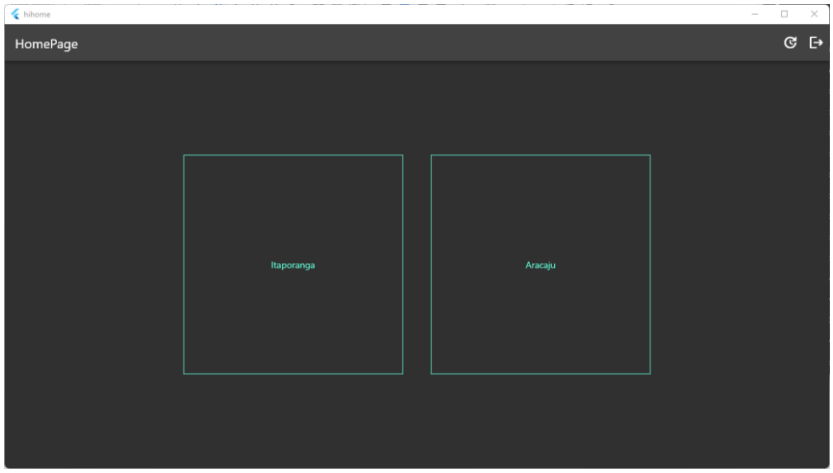
Figura 15: Visualização do supervisório no modo de dispositivos



Fonte: Autoria própria.

Levando em conta o objetivo das unidades possuírem seções e suas seções possuírem outras seções internas, é necessário escolher a seção que quer visualizar para acessá-la. A visualização das seções pode ser vista na Figura 16 e a troca entre seção e controle é feita utilizando o botão na barra superior nas seções seguintes.

Figura 16: Visualização do supervisório no modo seções

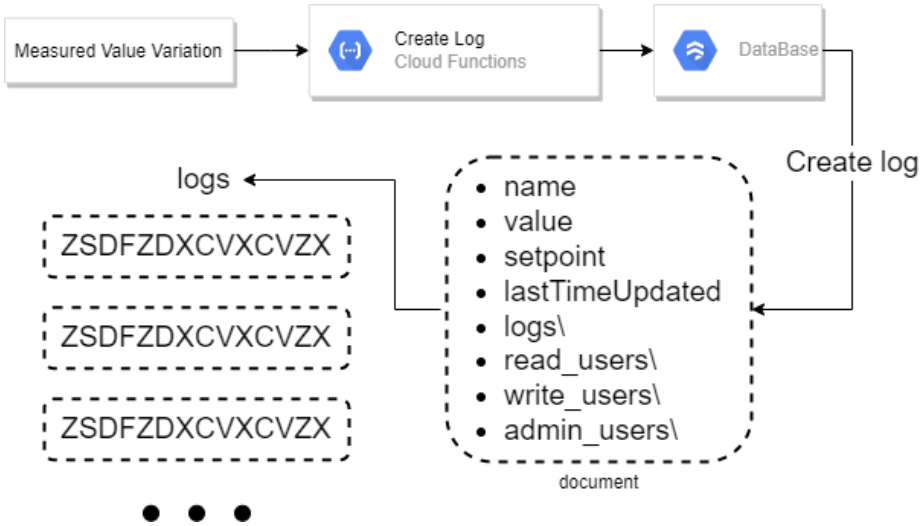


Fonte: Autoria própria.

3.7 Logs

A criação de logs do sistema foi desenvolvida através das Functions do Firebase. Toda alteração no valor atual do dispositivo dispara a criação de um documento novo dentro de uma coleção “logs” mantendo assim o objetivo de dados centralizados. A estrutura do caminho dos logs pode ser vista na Figura 17, onde os logs estão dentro do próprio documento.

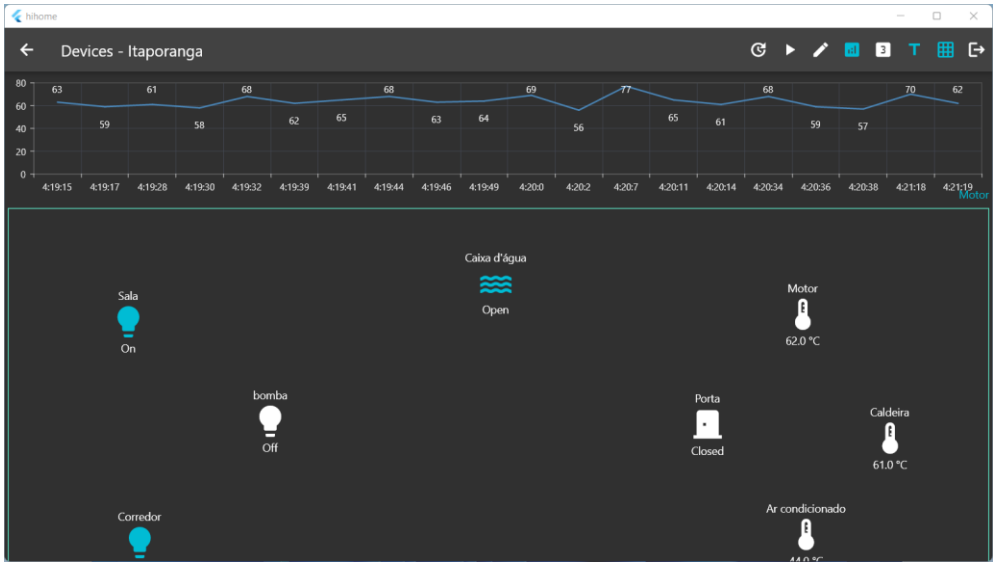
Figura 17: Function de criação de logs



Fonte: Autoria própria.

A visualização dos logs, vista na Figura 18, pode ser acessada clicando no modo de análise e escolhendo o dispositivo. Os últimos 20 logs são mostrados no topo da tela.

Figura 18: Visualização de logs



Fonte: Autoria própria.

3.8 Controle automático pelo lado do usuário

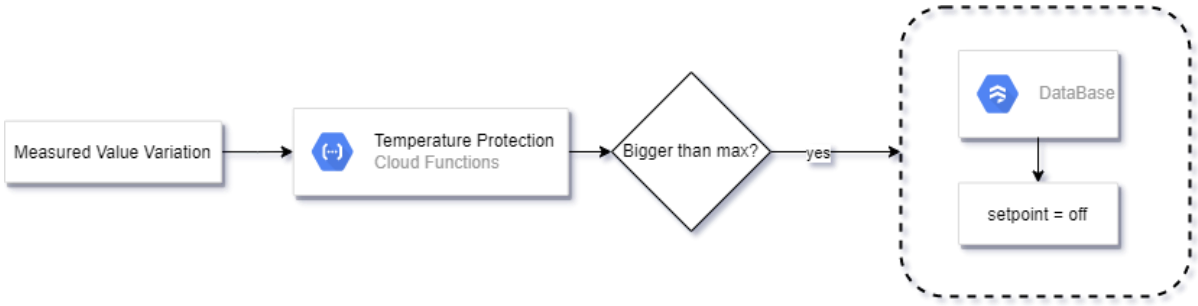
Além do controle automático implementado na planta com CLPs e outros dispositivos inteligentes, é possível automatizar ações através das Functions do Firebase. Como exemplo a proteção de motor com superaquecimento e no verão. Na planta os sensores e relés térmicos estão com as configurações inadequadas para o novo cenário.

A primeira opção geralmente seria enviar um operador para a planta, considerando seu custo, dificuldade de manutenção e perigo operacional. Outra opção seria a utilização dos dados na nuvem para o controle.

O controle automático pela nuvem pode ser dado por dois principais caminhos. O primeiro seria configurar o sistema para o valor da temperatura máxima ser guardado na nuvem e o sistema de desligamento ler constantemente este dado que

pode ser facilmente alterado. O segundo principal caminho, demonstrado na Figura 19, para principalmente quando a infraestrutura do caminho 1 não foi implementada no dispositivo IOT é o seu setpoint ser alterado automaticamente por um Function configurada para checar em toda variação de temperatura (valor atual do sensor) e desligar o motor, através da alteração do setpoint no documento da nuvem, caso passe da temperatura máxima.

Figura 19: Diagrama da Function para temperatura

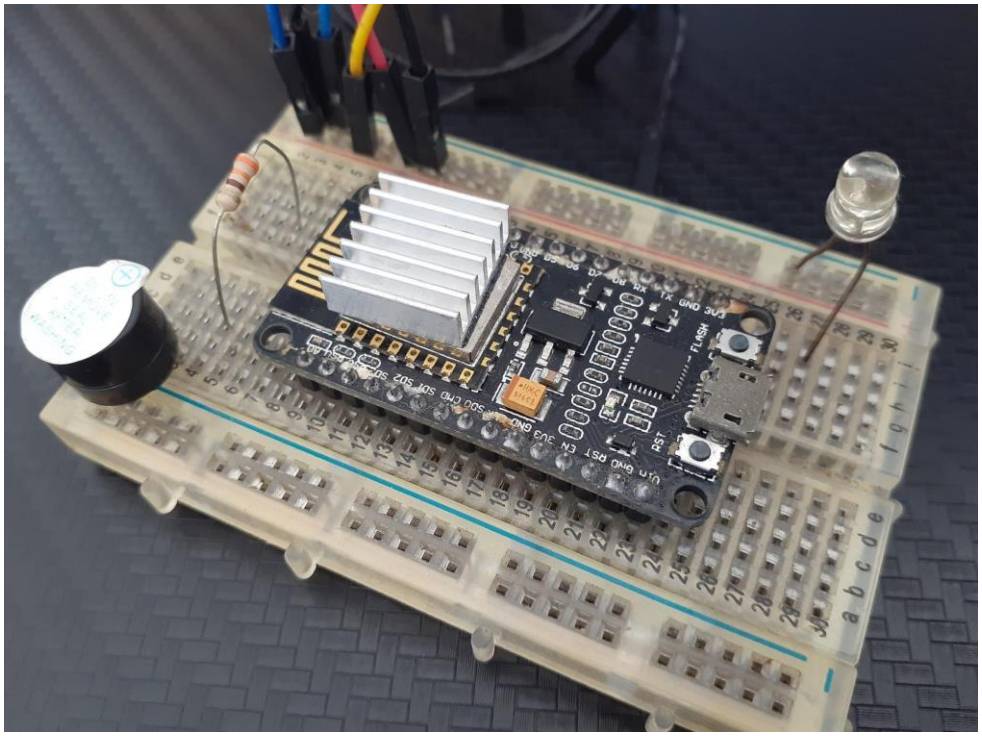


Fonte: Autoria própria.

3.9 Microcontrolador

O processo de desenvolvimento do sistema do microcontrolador foi feito em duas etapas, sendo elas os testes de código e de uso. A primeira etapa foi feita utilizando o microcontrolador NODEMCU, visto na Figura 20, pela facilidade de escrita do código, visto que o mesmo é da mesma familia ESP de microcontroladores, além de funcionar com a mesma programação que o ESP8266, variando apenas suas configurações na hora da gravação. Nele foram feitos os testes de conexão, lógica e velocidade de leitura.

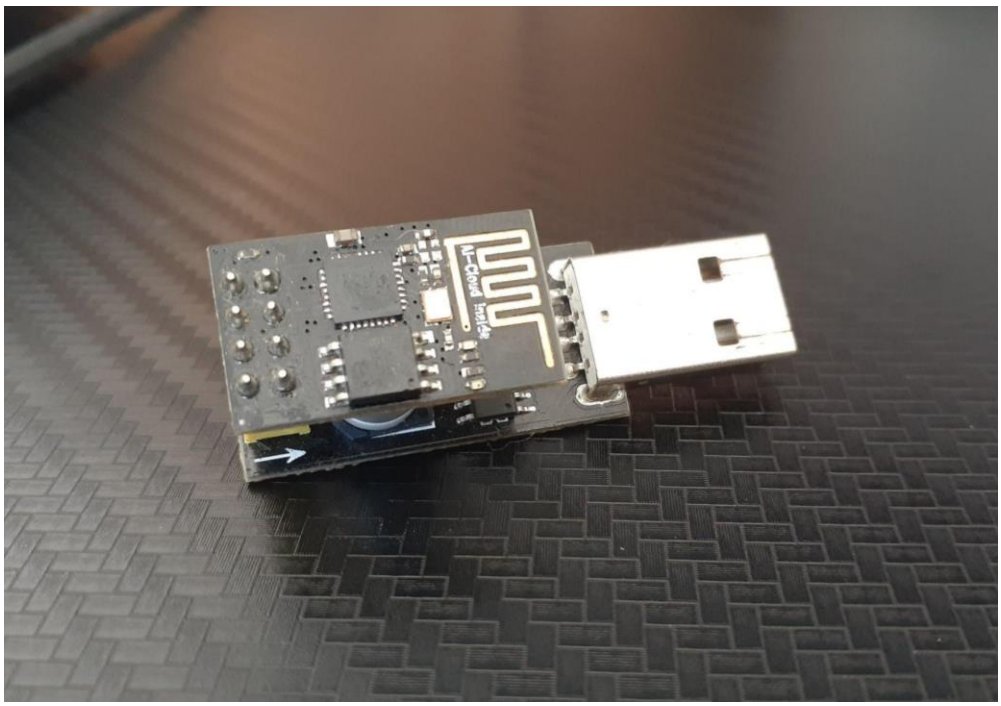
Figura 20: Placa NODEMCU



Fonte: Acervo pessoal.

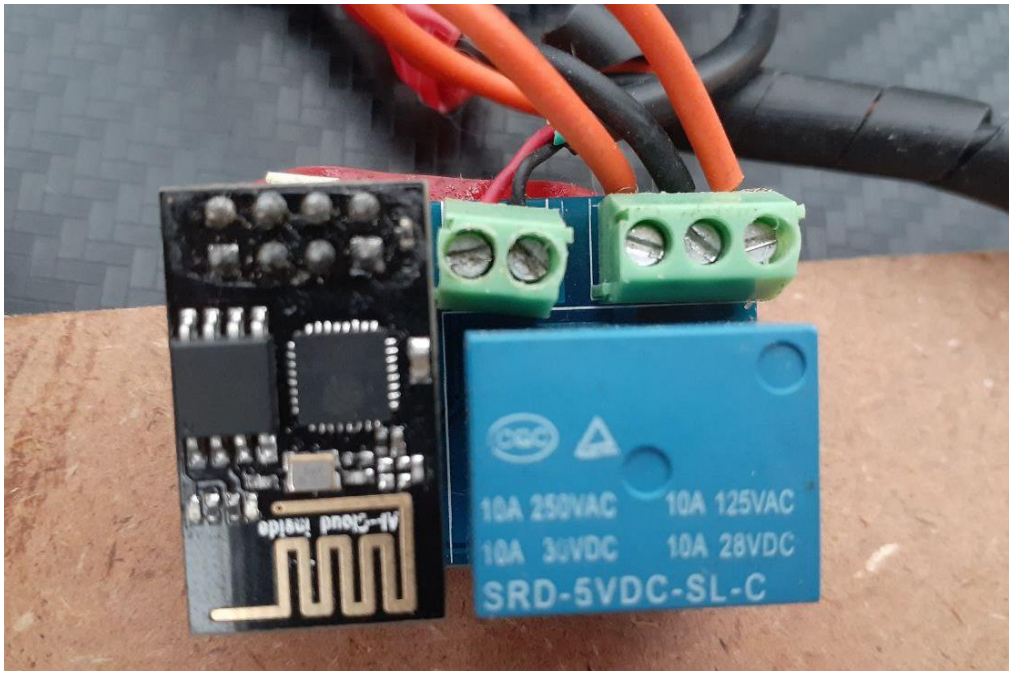
Ao final do desenvolvimento, o microcontrolador ESP8266 (Figura 21) foi implementado junto com um módulo relé para o acionamento de cargas (Figura 22), formando o protótipo de controle de uma lâmpada.

Figura 21: ESP8266 com seu gravador USB



Fonte: Acervo pessoal.

Figura 22: Módulo Relé com ESP8266



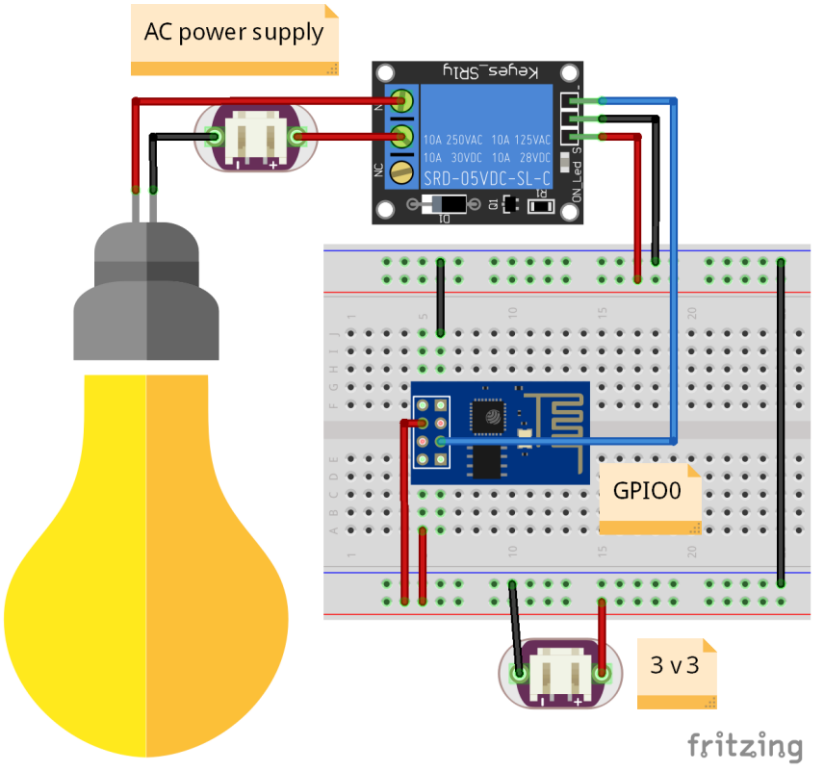
Fonte: Acervo pessoal.

A gravação no microcontrolador é feita com o gravador visto na Figura 23 junto de sua modificação para inicializar o ESP8266 em modo de programação através da adição de um botão de curto entre o GND e a GPI. O esquema eletrônico desenvolvido pode ser visto na Figura 24.

Figura 23: Modificação do gravador USB

Fonte: Acervo pessoal.

Figura 24: Esquema eletrônico utilizado no protótipo

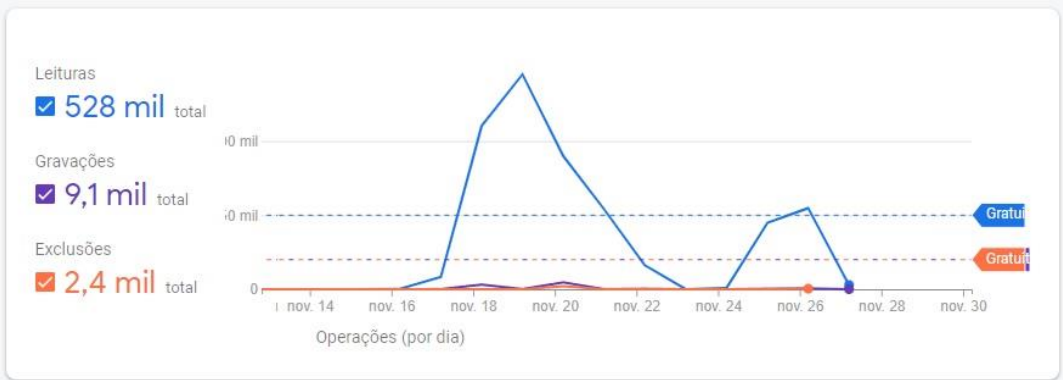


Fonte: Autoria própria.

A leitura do banco de dados foi feita através da API do Firestore em ciclos chamados através do protocolo HTTPS ao banco de dados para ter as informações mais atualizadas do setpoint. Antes de ler as informações do documento na nuvem é necessário autenticar o dispositivo IOT através de um email e senha com acesso criado para dispositivo (Ex: iot123@database.com) e assim receber um token que será usado para as futuras leituras e ações no banco. Em caso de expiração do token é feito o login novamente no sistema. Um gráfico de utilização, visto na Figura 25,

ilustra os testes do projeto entre os dias 16 e 27 de novembro. O uso normal de um dispositivo durante 24 horas foi em média de 50 mil leituras.

Figura 25: Utilização do Banco de Dados Firebase Firestore



Fonte: Autoria própria.

O retorno do Firestore é um texto no formato JSON (*JavaScript Object Notation* - Notação de Objeto em JavaScript) e nele é retirada as informações do estado atual do documento. O microcontrolador obtém esses valores e atua para o caso de acionamento, desacionamento ou mudança de uma variável analógica. Para o caso de estados compostos como um PID, esses valores poderiam estar em diferentes campos ou unidos com um separado escolhido. O JSON pode ser visto na Figura 26.

Figura 26: JSON recebido pelo microcontrolador


```
{
    "fields": {
        "name": {
            "stringValue": "Office"
        },
        "point": {
            "mapValue": {
                "fields": {
                    "y": {
                        "doubleValue": 0.22801089666079929
                    },
                    "x": {
                        "doubleValue": 0.10073343912760478
                    }
                }
            }
        },
        "id": {
            "stringValue": "xC8UGLSYT8z2pxwKaAeY"
        },
        "type": {
            "stringValue": "lamp"
        },
        "value": {
            "stringValue": "off"
        }
    }
}
```


Fonte: Autoria própria.

4 RESULTADOS E DISCUSSÕES

A partir do vídeo que pode ser acessado através do QR Code que é ilustrado na Figura 1 do Apêndice, no final deste trabalho, é possível visualizar uma demonstração da aplicação do sistema em funcionamento em tempo real.

No vídeo disponibilizado é possível visualizar o modelo de desenvolvimento IoT *Serverless* para Indústria em funcionamento, onde após coletar as informações geradas pelos dispositivos IoT, que a fim de simular um ambiente de fábrica, esses dados foram gerados de forma aleatória, o provedor de nuvem Google Cloud Platform faz o armazenamento desses dados através da plataforma do Firebase, ao qual pode

ser acessado através de uma requisição HTTPS, que por sua vez pode ser efetuada através de uma API programada pelo framework Flutter.

A interface Web é uma das formas que a central de operações pode estar realizando a supervisão de determinada operação em tempo real. Além da parte supervisória, a central de operações também possui a capacidade de controlar as máquinas através do comando de desligar e ligar. Outra aplicabilidade presente nessa mesma interface, é a capacidade do operador de realizar o controle de temperatura dessa mesma máquina.

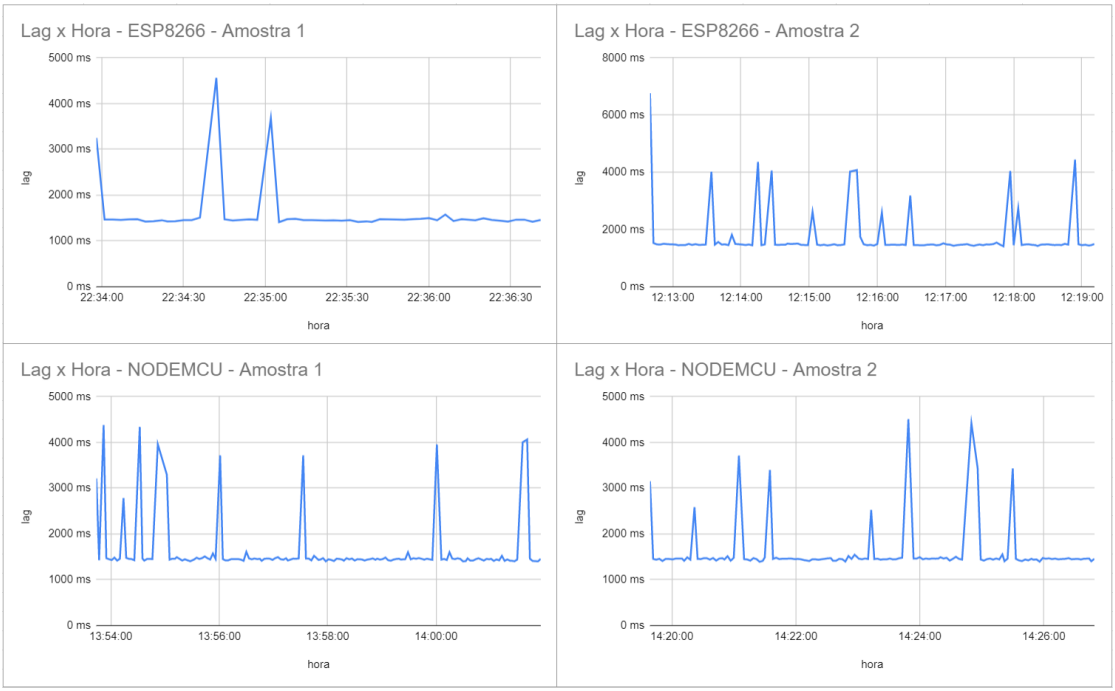
Com a implementação de um modelo *Serverless* dentro da indústria, a central de operações pode voltar a sua atenção para funções mais importantes dentro da empresa, ao invés de se preocuparem com o gerenciamento e a operação de servidores, já que eles não serão mais gerenciados por eles, e sim pelos provedores da nuvem. Com isso, é possível apontar uma alternativa para a migração de um modelo tradicional para um modelo *Serverless* na indústria, de modo que as organizações terceirizam o servidor e as tarefas de gerenciamento de infraestrutura associadas a terceiros, fazendo assim com que mais recursos possam ser gerados e alocados em outras áreas.

Através do sincronismo de dados fornecido pelo firebase foi possível ter a atualização dos valores na nuvem quase que de forma instantânea entre operadores que estão utilizando o supervisório. Já para o microcontrolador houve a limitação de sincronismo pelo tempo de processamento.

Os dados de tempo de leitura para os dois microcontroladores usados no projeto, vistos na Figura 27 e Tabela 01, mostram que o tempo médio para ler os dados na nuvem e atualizar as saídas do microcontrolador foi de 1465 milissegundos. Em termos práticos de utilização, o lag do sistema é menor que o máximo, pois o tempo de propagação dos dados após atualizar as informações na nuvem e o microcontrolador recebê-las está entre o ciclo atual de leitura e a próxima, então para 70% das vezes o tempo prático de propagação será menor do que 1025 milissegundos.

Alguns dos fatores para esse tempo de propagação são a velocidade de processamento, localização do servidor da nuvem, qualidade do sinal wi-fi e qualidade da antena dos microcontroladores. Em adição, ambos dispositivos foram produzidos há alguns anos, então não possuem a velocidade de processamento e qualidade de conexão dos novos dispositivos IoT disponíveis no mercado.

Figura 27: Amostras de leitura do ESP8266 e NODEMCU



Fonte: Autoria própria.

Tabela 01: Média de tempo do ESP8266 e NODEMCU

Tempos de Leitura				
Dia	Quantidade de leituras	Tempo de teste	Média	Microcontrolador
25/11/2021	52	3 min	1459 ms	ESP8266
25/11/2021	122	7 min	1471 ms	
26/11/2021	155	8 min	1575 ms	NODEMCU
26/11/2021	137	7 min	1456 ms	
Média	130	7 min	1465 ms	Média

Fonte: Autoria própria.

Para comprovar essa limitação pelo lado dos dispositivos IOT foram feitos testes com com um programa Python rodando em um computador conectado ao cabo e utilizando a biblioteca do Firestore. O tempo de leitura médio foi reduzido em 83,22% e o tempo necessário para fazer a mesma quantidade de requisições que os microcontroladores foi reduzido em 91,19% devido à maior estabilidade entre as chamadas da nuvem. O tempo equivalente ao tempo médio de leitura foi de 246 milissegundos. Estas informações podem ser vistas na Tabela 02.

Tabela 02: Comparação entre os microcontroladores e um programa Python utilizando a API REST

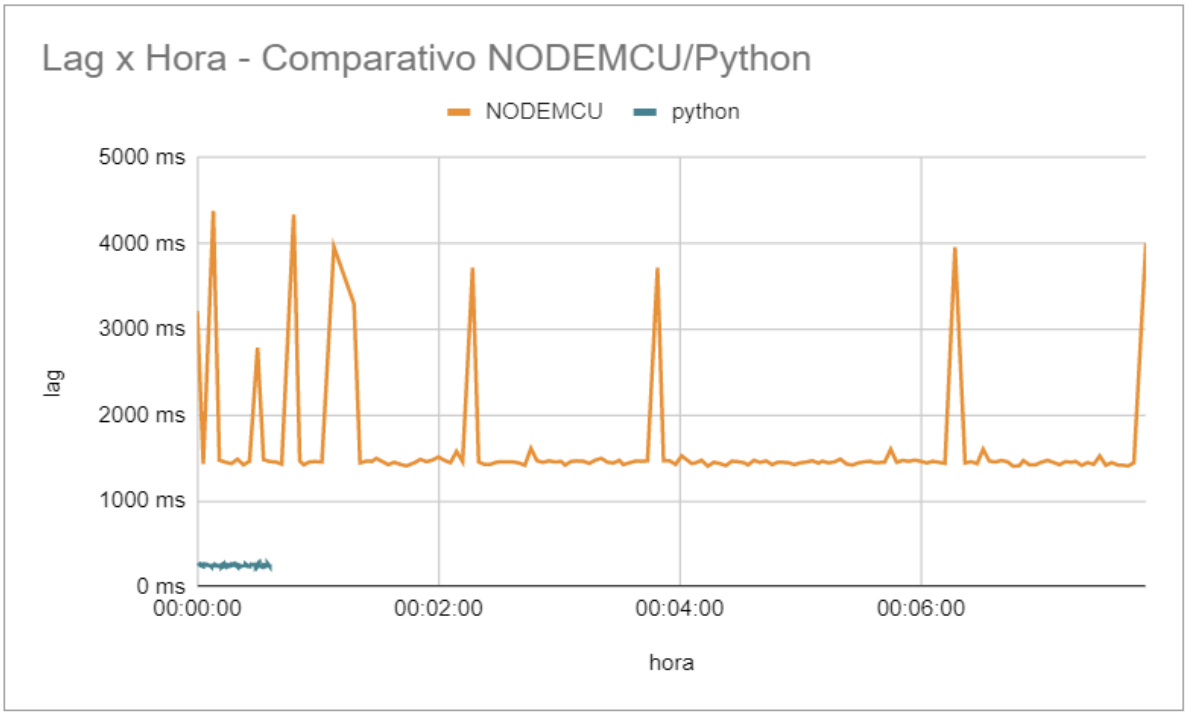
Tempos de Leitura - Comparação				
Dia	Quantidade de leituras	Tempo de teste	Média	Microcontrolador
26/11/2021	130	7 min	1465 ms	ESP/NODEMCU
26/11/2021	150	37 s	246 ms	Python-Desktop
Redução		91,19%	83,22%	

Fonte: Autoria própria.

A comparação de redução pode ser vista na Figura 28 onde há um gráfico unindo os gráficos de tempo de leitura do NODEMCU e o Python. É possível perceber

que o gráfico do Python termina muito antes em 37 segundos e o NODEMCU em 7 minutos e 52 segundos.

Figura 28: Gráfico unindo tempos de leitura do NODEMCU e Python



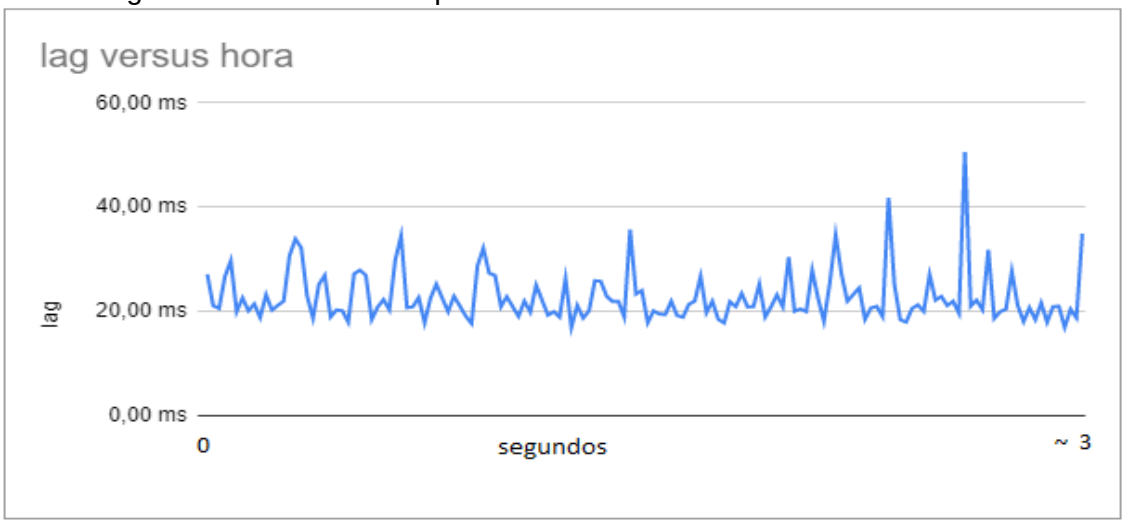
Fonte: Autoria própria.

O tempo médio que o Python, fazendo chamadas da API REST do Firestore, levou para fazer as leituras mostrou como a velocidade do sistema pode ser melhorada com novos microcontroladores ou a utilização de CLPs industriais.

Aplicando a estrutura de conexão com a nuvem através da biblioteca do Firestore para Python foi possível ter uma atualização instantânea dos valores alterados no banco. A partir daí o lag existente foi devido ao tempo de conexão com dispositivo IOT em modo servidor WEB.

É possível ver na Figura 29 e na Tabela 3 o tempo das chamadas HTTP para o servidor do dispositivo IOT. Sua média foi de 21 milissegundos e sua redução, comparada à chamada da API pelo Python, foi 91,40%.

Figura 29: Gráfico do tempo de conexão do ESP8266 como servidor WEB



Fonte: Autoria própria.

Tabela 03: Comparação entre a chamada da API REST com Python e o uso do ESP8266 em modo web server com a biblioteca do Firestore para Python

Tempos de Leitura - Comparação				
Dia	Quantidade de leituras	Tempo de teste	Média	Microcontrolador
26/11/2021	150	37 s	246 ms	Python-Desktop
29/11/2021	150	3 s	21 ms	ESP-WEB-SERVER
Redução		91,89%	91,40%	

Fonte: Autoria própria.

Outro ganho dessa infraestrutura é a redução de leituras na nuvem pela retirada das chamadas contínuas para checar se houveram mudanças nos dados do dispositivo. Em 24 horas de uso foram aproximadamente 100 leituras, totalizando uma redução de 99,8%.

5 CONCLUSÕES

Este trabalho descreveu e discutiu uma proposta de um sistema IoT *Serverless* para a Indústria 4.0. Esse modelo de aplicação da IoT no meio industrial foi fundamentado em melhorar o gerenciamento de produção de uma empresa, com o objetivo de torná-la mais eficiente e eficaz, além de render diversos outros benefícios para a empresa que aderir a esse modelo, como a redução de gastos desnecessários, entre eles o consumo de energia e o de matéria prima, que atualmente é um dos maiores problemas presentes dentro de qualquer empresa.

A partir do que foi apresentado, conclui-se que a IoT pode proporcionar uma grande evolução para a área de automação e controle, uma vez que existem algumas técnicas ultrapassadas quanto a maneira que é realizada a coleta de dados da linha de produção de uma parte das indústrias que atuam no mercado atualmente, o que pode ser resolvida facilmente com a implementação de um sistema supervisório de baixo custo baseado em um serviço Web, fazendo assim com que o monitoramento da linha de produção possa ser feito de forma remota.

Além de tornar o controle mais eficiente, o uso da tecnologia IoT também carrega o potencial de oferecer uma maior mobilidade, já que pode ser acessada de qualquer lugar, tendo assim conhecimento do que ocorre dentro do ambiente fabril e proporciona também uma melhoria de aproveitamento de espaço do local, contendo somente o necessário.

Como melhorias e análises futuras têm-se a implementação em um controlador industrial para avaliar a melhoria do tempo de processamento em relação ao microcontrolador escolhido, implementar o banco com o Firebase realtime e controlar uma planta industrial real.

Por fim, é possível concluir que este trabalho atendeu de forma satisfatória todos os seus objetivos, e pode contribuir para pesquisas futuras a respeito da implementação de tecnologias *Serverless* e IoT na Indústria 4.0.

REFERÊNCIAS

DATE, Christopher J. Introdução a sistemas de bancos de dados. Elsevier Brasil, 2004.

DE OLIVEIRA, Sérgio. Internet das coisas com ESP8266, Arduino e Raspberry PI. Novatec Editora, 2017.

ELMASRI, Ramez; NAVATHE, Sham. Fundamentals of database systems. Pearson, 2017.

FACCIONI FILHO, Mauro. Internet das coisas. Unisul Virtual, 2016.

HENDRICKSON, Scott et al. Serverless computation with openlambda. In: 8th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 16). 2016.

KERSCHBAUMER, Ricardo. Microcontroladores. Santa Catarina: Instituto Federal de Educação, Ciência e Tecnologia, 2018.

MACDOUGALL, William. Industrie 4.0: Smart manufacturing for the future. Germany Trade & Invest, 2014.

MAGRANI, Eduardo. A internet das coisas. Editora FGV, 2018.

MCKINSEY GLOBAL INSTITUTE, The Internet of Things: Mapping the Value Beyond the Hype, 2015.

NEUMANN, Andy; LARANJEIRO, Nuno; BERNARDINO, Jorge. An analysis of public REST web service APIs. IEEE Transactions on Services Computing, 2018.

RICHARDSON, Leonard et al. RESTful Web APIs: Services for a Changing World. "O'Reilly Media, Inc.", 2013.

SIMATIC IOT2050. Siemens, 2020. Disponível em: <https://new.siemens.com/br/pt/produtos/automacao/pc-based/iot-gateways/simatic-iot2050.html>. Acesso em: 22 de nov. 2021.

TANENBAUM, Andrew S. Redes de computadores. Campus, 2003.

TAURION, Cezar. Cloud computing-computação em nuvem. Brasport, 2009.

THOMAZINI, Daniel; DE ALBUQUERQUE, Pedro Urbano Braga. Sensores industriais: fundamentos e aplicações. Saraiva Educação SA, 2020.

TORRES, Gabriel. Redes de Computadores: curso completo. Axcel Books do Brasil Editora Ltda, 2004.

VELTE, Anthony; VELTE, Toby; ROBERT, Elsenpeter. Computação em Nuvem: Uma Abordagem Prática. Alta Books, 2011.

APÊNDICE

Figura 1 – Qr-code com vídeo e documentos do projeto

Fonte: Autoria própria.

Este documento possui 6900 palavras.