Sistema de monitoramento remoto para nanocervejarias

UBERLÂNDIA

2020

Sistema de monitoramento remoto para nanocervejarias

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Orientador: Prof. Dr. Marcelo Barros de Almeida

UBERLÂNDIA

2020

Borges Filho, Mario Divino
Sistema de monitoramento e controle remoto para nanocervejarias/ **Mario Divino Borges Filho**. – **UBERLÂNDIA**, **2020**- 28 p.: il. (algumas color.); 30 cm.

Orientador: Prof. Dr. Marcelo Barros de Almeida

Trabalho de Conclusão de Curso – Universidade Federal de Uberlândia – UFU Faculdade de Engenharia Elétrica. **2020**.
Inclui bibliografia.

1. Indústria 4.0. 2. Internet das coisas. 2. Computação em nuvem. I. Orientador Prof. Dr. Marcelo Barros de Almeida. II. Universidade Federal de Uberlândia. III. Faculdade de Engenharia Elétrica. IV. Engenharia de Controle e Automação.

Dedicatória

Dedico este trabalho aos meus pais Ângela Maria da Silva e Mário Divino Borges pelo apoio e pela base sólida que me deram para que pudesse chegaram até aqui, aos meus irmãos Rodrigo Luiz da Silva, Nayara Silva Borges e Rita Stella Silva Borges, por estarem sempre ao meu lado e me fortalecer para a batalha.

Agradecimentos

Agradeço primeiramente à Deus por sempre olhar por mim e guiar meus caminhos pela luz e verdade.

À Universidade Federal de Uberlândia que juntamente com a Faculdade de Engenharia Elétrica me concederam um ensino de qualidade e toda a base para chegar neste momento.

Ao meu professor orientador Marcelo Barros de Almeida por sempre se dispor a ajudar e sempre incentivar minha evolução durante este trabalho bem como no decorrer do curso.

A todos os Professores que ao longo da minha vida nunca mediram esforços para passar seu conhecimento e colaborar com o meu crescimento e constante evolução.

Aos meus colegas de curso e amigos que tornaram esses momentos ainda mais agradáveis, à toda a minha família e todos aqueles que de alguma forma contribuíram para o meu sucesso.

“Deixem que o futuro diga a verdade e avalie cada um de acordo com o seu trabalho e realizações. O presente pertence a eles, mas o futuro pelo qual eu sempre trabalhei pertence a mim.”
(Nikola Tesla)

Resumo

A evolução de novas tecnologias implantadas na indústria caminha a passos largos, essa evolução leva à indústria 4.0. A demanda por tecnologias que integra um novo modelo já começa a crescer. É de grande importância desenvolver trabalhos e projetos que expõem novas formas e possibilidades de implementação dessas novas tecnologias. A internet das coisas já faz parte do nosso dia-a-dia e estará muito mais presente em um futuro próximo, tanto nas indústrias quanto em nossas casas com cada vez mais "coisas" conectadas à nuvem.

Dispositivos periféricos com cada vez mais inteligência agregada contribuem para um crescimento exponencial na quantidade de informações disponíveis nos processos. Com esse aumento de informações, o seu processamento se torna sempre mais complexo e de nada adianta um volume grande de dados se o processamento não é suficiente para aproveitá-los como informações concretas. Os sistemas desenvolvidos para esse processamento devem possuir escalabilidade, disponibilidade e facilidade de gerenciamento, para que não ocorram problemas de infraestrutura. Visando esses requisitos, neste trabalho são apresentadas ferramentas de computação em nuvem que proporcionam a infraestrutura necessária para projetos de IoT.

O fato de essas novas tecnologias estarem sendo implantadas primeiro nas indústrias torna o seu preço pouco acessível para serem usadas por nanocervejarias ou em automações residenciais. O objetivo deste trabalho é apresentar uma nova abordagem no monitoramento e controle remoto de uma etapa na produção de cerveja artesanal, a fermentação. A busca por tecnologias acessíveis ao bolso do pequeno empresário justifica e guia as decisões quanto ao material utilizado neste projeto, que visa implantar um sistema robusto e confiável de controle e monitoramento do processo de fermentação da cerveja artesanal.

Palavras-chave: Indústria 4.0, Internet das coisas, computação em nuvem

Abstract

The evolution of new technologies implanted in the industry is taking a long step, this evolution leads to industry 4.0. The demand for technologies that integrate a new model is already starting to grow. It is of great importance to develop works and projects that expose new forms and possibilities of these new technologies. IOT will be very present in our lives, both in the industries and in our homes with more and more “things” connected to the cloud.

Peripheral devices with more aggregate intelligence contribute to an exponential growth in the amount of information available in the processes. With this increase in information, its processing always becomes more complex and there is no use in a large volume of data if its processing is not enough to take advantage of it in the correct way. The systems developed for this processing must guarantee scalability, availability and ease of management, so that infrastructure problems do not occur. Aiming at these requirements, this work will present cloud computing tools that provide the necessary infrastructure for the project.

The fact that these new technologies are being implemented first in the industries makes their price less accessible to be used by small manufactures or in-home automation. The objective of this work is to present a new approach in the monitoring and remote control of a stage in the production of craft beer. The search for technologies accessible to the pocket of the small business owner justifies and guides the decisions regarding the material used in this project, which aims to implement a robust and reliable control and monitoring system for the fermentation process of craft beer.

Keywords: Industry 4.0, Internet of things, Cloud computation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ARM	*Advanced RISC Machine*
API	*Application Programing Interface*
ASCII	*American Standard Code for Information Interchange*
AWS	*Amazon Web Services*
CLP	Controlador lógico programável
CoAP	*Constrained Application Protocol*
CRC	*Cyclic Redundancy Check*
DB	Database
DNS	*Domain Name System*
GNS	*Graduated Neutral Density*
HTTP	*Hypertext Transfer Protocol*
IEEE	*Instituto de Engenheiros Eletricistas e Eletrônicos*
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IHM	*Interface Homem-Máquina*
IoT	Internet of Things
IPv6	*Internet Protocol version 6*
Kbits/s	Kilobits por segundo
MQTT	*Message Queue Telemetry Transport*
PWM	*Pulse Wave Modulation*
RS-485	*Recommended Standard 485*
RS-232	*Recommended Standard 232*
S3	*Simple Storage Service*
TCP/IP	*Transmission Control Protocol / Internet Protocol*

UART *Universal Asynchrounous Receiver/Transmite*

URL *Uniform Resource Locator*

6LoWPAN *IPv6 over Low power Wireless Personal Area Networks*

Sumário

1. Introdução

O mundo vive atualmente a 4° revolução industrial com o advento da internet das coisas e de novas tecnologias que tornam o processo industrial mais conectado e eficiente. A industrial 4.0 sofre grande resistência em sua disseminação no Brasil por conta de entre outras coisas, os valores agregados nos equipamentos e tecnologias, adaptações de cultura e processos e o investimento em novas tecnologias (Confederação Nacional da Indústria, 2016). Segundo (Nazaré, Rocha, Oliveira, Souza, & Ramos, 2019) o Brasil já passa por uma nova tendência de abordagem industrial onde deverá ocorrer a inserção e adaptação das tecnologias para a modernização do mercado nacional e equalização para concorrência com o mercado externo. As tecnologias empregadas neste novo modelo de indústria devem ser estudadas a fundo para que as suas possibilidades sejam exploradas ao máximo.

A internet, que já gerou uma enorme revolução no modo em que vivemos, tem potencial para mudar o mundo de uma forma ainda mais profunda. Os dispositivos eletrônicos e eletrodomésticos que antes funcionavam “sozinhos”, hoje se conectam com a rede mundial de computadores possibilitando a criação de um ambiente de sensoriamento e controle totalmente automatizado. A internet das coisas é comumente descrita como um ambiente dispositivos eletrônicos conectados à internet criando, por meio de sensores e atuadores, um ecossistema de computação onipresente voltado para a facilitação do cotidiano das pessoas, a exemplo desta integração existem máquinas de lavar que podem ser controladas por smartphone (figura 1), fechaduras que identificam o usuário e se abrem automaticamente (figura 2), entre outros.

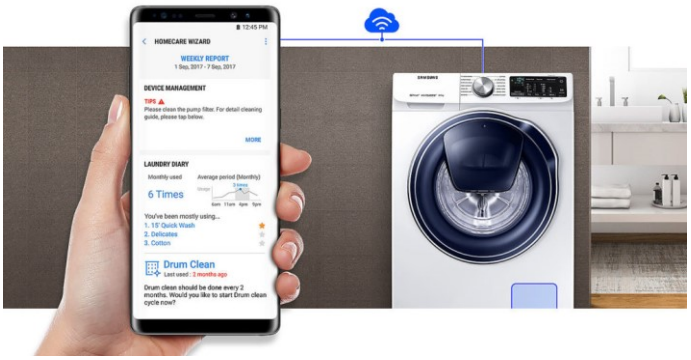

Figura 1 - Máquina de lavar roupas controlada por smartphone

Figura 2 - Fechadura eletrônica inteligente

Atualmente não há um conceito definitivo para o que é uma “coisa”, mas, a grosso modo, todo dispositivo que tem a capacidade de medir, armazenar, analisar ou compartilhar dados com outros dispositivos pela rede pode ser considerada uma “coisa”. Existem diversos protocolos de comunicação quando se trata de redes de computadores, cada um tem o seu nicho de atuação como o protocolo HTTP para aplicações web, modbus para redes industriais e o MQTT para internet das coisas.

O emprego das novas tecnologias de internet das coisas ganha mais um nicho com a difusão e popularização das técnicas de produção de cervejas artesanais. A demanda por cervejas especiais cresceu bastante no brasil. Com aumento na procura os produtores que antes utilizavam apenas suas mãos e técnicas terão de lançar mão da tecnologia para otimizar o processo. Segundo (Godoi, Santos, Campos, & Santos, 2016) somente 7% das microcervejarias tem o status de bastante automatizados, enquanto o restante tem muito pouco ou quase nenhum nível de automação em seus equipamentos. Esta carência se dá por vários motivos entre eles os altos valores agregados aos equipamentos de automação industrial.

1.1. Justificativa

O mercado de consumo de cervejas especiais vem crescendo, isso aumenta a necessidade dos produtores em otimizar o processo para suprir a demanda. A automação da malha de produção é melhor forma para se otimizar a produção, porém os altos valores das soluções em automação industrial é um dos motivos que impedem que o pequeno produtor de cerveja artesanal automatize a sua pequena fábrica. Conforme pode ser observado na figura 3 uma pequena parcela dos produtores é bastante automatizada. Os custos envolvem desde a compra dos dispositivos eletrônicos de automação industrial, até ao desenvolvimento de um sistema de gerenciamento do processo. Neste cenário é necessário encontrar formas mais acessíveis para essas soluções e assim suprir as necessidades do pequeno produtor.

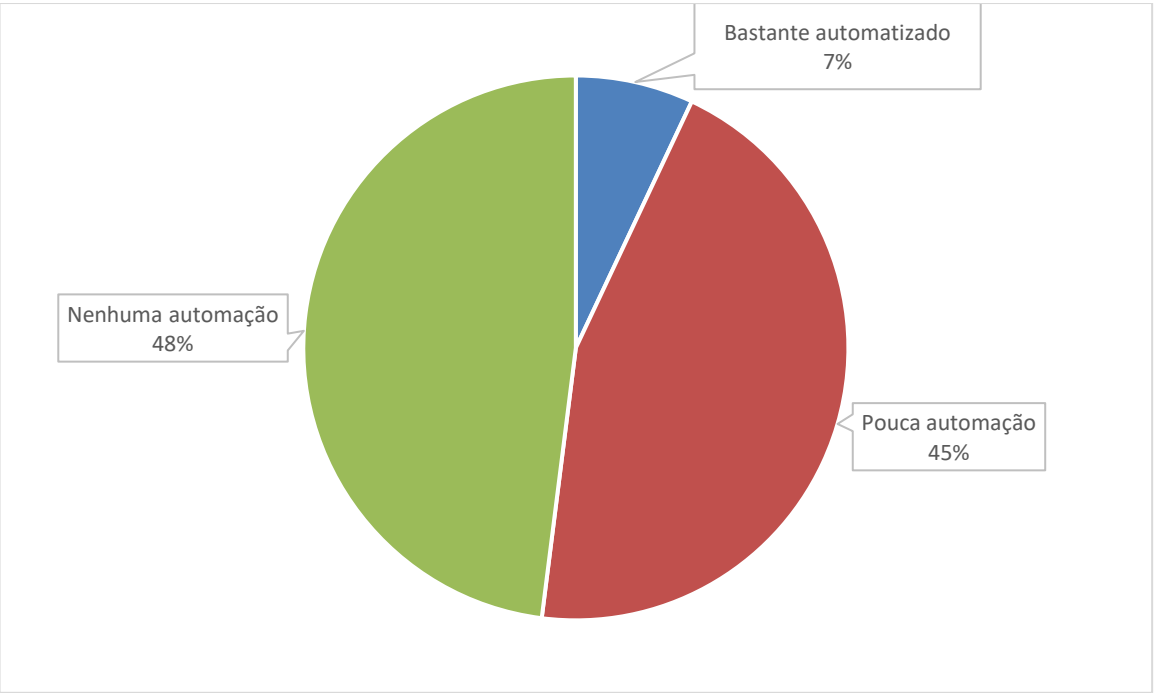

Figura 3 - Nível de automação em nanocervejarias artesanais. Adaptado de (GODOI, SANTOS. et al., 2016)

Apresentar novas técnicas e ferramentas de computação e automação é necessário para que existem alternativas às soluções tradicionais. O desenvolvimento de novas ferramentas pode viabilizar um crescimento no setor proposto e ainda de outros setores como a domótica cujas expectativas de crescimento nos próximos anos é muito grande.

1.2. Objetivo

O objetivo principal deste trabalho é projetar um sistema para monitoramento e controle remoto do processo de fermentação de cerveja artesanal que seja acessível financeiramente para os padrões de investimento das microcervejarias e que ao mesmo tempo seja robusto a ponto de suportar o processo produtivo. Este sistema deve mostrar os dados do processo em uma interface supervisória e também deve atuar sobre variáveis do processo quando for necessário.

Além do objetivo principal este trabalho também possui alguns objetivos específicos descritos abaixo:

- Implementar uma rede modbus RTU sobre meio físico RS-485 para aquisição confiável dos dados do processo;
- Desenvolver um sistema para o processamento local e troca dos dados com o sistema web via protocolo MQTT;
- Apresentar ferramentas disponíveis e desenvolver um sistema web sem servidor que troque dados com o sistema embarcado instalado na fábrica;

2. Metodologia

2.1. Fermentação de cerveja artesanal

O processo de fermentação do mosto cervejeiro segundo (Rosa & Afonso, 2015), transmite propriedades organolépticas à cerveja e por conta disso a fermentação é a etapa mais relevante para definir o sabor do produto final. O controle desta etapa do processo, portanto é extremamente importante para garantir as características esperadas naquela batelada, bem como para garantir um padrão nos produtos, o que é relevante para qualquer marca que queira criar uma identidade.

A fermentação começa logo após o mosto cervejeiro ser resfriado à temperatura específica para cada receita. As leveduras são adicionadas ao mosto dentro do tanque de fermentação. Pequenos produtores usam reatores como duas camadas e uma serpentina entre essas camadas para controlar a temperatura do processo, como pode ser observado na figura 4.

Figura 4 - Tanques de fermentação de cerveja artesanal

O atuador no controle de temperatura é comumente chamado de glicol. O glicol mais utilizado é composto por uma mistura de água com etanol em uma proporção suficiente para que a mistura não se congele a baixas temperaturas e troque bem o calor com os tanques de fermentação. O banco glicol é composto por um trocador de calor, um tanque onde o líquido é armazenado e uma bomba que manda o líquido até os tanques quando necessário. A estrutura do banco glicol pode ser observada na figura 5.

Figura 5 - Banco de líquido de resfriamento glicol

A etapa de fermentação é a que leva mais tempo na produção de cerveja artesanal. Leva-se desde várias horas a algumas semanas para uma receita fermentar corretamente. A fermentação cervejeira, de uma forma simples, consiste na conversão dos açúcares fermentescíveis do mosto em álcool, gás carbônico e subprodutos pela ação da levedura. O principal foco dessa etapa é, usando a tecnologia adequada, conduzir as interações de todos os parâmetros envolvidos no processo (composição do mosto, linhagem da levedura, temperatura de processo, geometria do tanque, etc.), para obter a cerveja com as características sensoriais, químicas e físico-químicas previamente determinadas (Briggs, Boulton, Brookes, & Stevens, 2004). A criticidade desta etapa faz com que um monitoramento constante dos parâmetros do processo seja de extrema importância. Por se tratar de um processo químico exotérmico a variável do processo é a temperatura.

2.2. Recursos web

A interface web neste trabalho busca ser o mais simples possível no sentido de arquitetura e que possua robustez, escalabilidade e confiabilidade. Para a criação do sistema web seguindo as características já mencionadas o caminho encontrado foi utilizar as ferramentas de computação em nuvem da Amazon Web Services (AWS). A AWS oferece diversos serviços como de bancos de dados com o DynamoDB, armazenamento de dados com o S3 desenvolvimento, desenvolvimento e gerenciamento de APIs com o API gateway e computação em nuvem com o Lambda. Todas essas ferramentas podem ser usadas em conjunto para o desenvolvimento de aplicativos e sistemas em geral.

A aplicação web deste trabalho usa o S3 para a hospedagem do site estático com todos os seus elementos. A estrutura do back-end é composta por um banco de dados que armazena e organiza as informações do processo, este é o dynamoDB e suas

características serão explanadas posteriormente, a plataforma de computação em nuvem lambda que processa os dados e as requisições da aplicação web e o Api Gateway que faz a integração do back-end com o front-end através de APIs.

Os recursos e serviços oferecidos pela AWS são hospedados em diferentes locais físicos por todo o mundo. Cada local é composto por uma região, uma zona de disponibilidade e uma zona local. As regiões são áreas geográficas separadas e cada região contém vários locais isolados que são chamados de zonas de disponibilidade. Quando os recursos são instanciados o desenvolvedor escolhe uma região. As regiões são projetadas para serem isoladas umas das outras para maximizar a tolerância a falhas. Alguns recursos não estão disponíveis para serem instanciados em todas as regiões existentes, por conta disso há uma variação nas regiões dos recursos usados neste projeto. Cada bucket possui políticas que restringem o acesso aos seus objetos e recursos. A política de acessos pode ser definida através da lista de controle de acessos ou da política de buckets, esta política será usada no sistema de segurança dos dados e recursos do sistema.

Simple Storage Service

O AWS S3, segundo o (Amazon Web Service, 2020) é um serviço de armazenamento para a internet e foi desenvolvido para tornar a computação para web para os clientes desenvolvedores. Através do S3 é possível armazenar e recuperar arquivos, gerenciar permissões em seus recursos e criar buckets, repositórios onde os dados são armazenados. Ao criar um bucket o seu nome é exclusivo globalmente e não pode ser utilizado por outra conta até que o mesmo seja excluído. A plataforma de serviços da AWS usa um esquema de regiões para o uso dos seus recursos, ainda segundo (Amazon Web Service, 2020) é mais vantajoso escolher uma região geograficamente próxima para instanciar os recursos para que a latência da rede seja otimizada e requisitos regulatórios sejam cumpridos.

Serviço de processamento lambda

O serviço de computação em nuvem utilizado neste projeto é o AWS lambda que segundo (Amazon Web Service, 2020) é uma ferramenta de computação que permite a execução de códigos sem que o desenvolvedor precise provisionar ou gerenciar servidores. Tradicionalmente uma aplicação web consiste em três partes, o front-end que abriga a interface da aplicação e todas as funcionalidades client-side que são executadas pelo navegador, o back-end que recebe, processa e responde às solicitações do cliente e a integração entre ambas as partes que é feita por APIs que executam as trocas de dados em cada solicitação. A AWS oferece a possibilidade de se desenvolver aplicações sem servidor, que tem como principal diferença do método tradicional o fato de que o cliente é cobrado com base no seu uso dos recursos e não pela quantidade de

servidores alocados ou por uma quantidade fixa de largura de banda.

A arquitetura sem servidor permite que os clientes escrevam e implantem seus códigos sem precisar se preocupar com a infraestrutura subjacente. Na arquitetura tradicional, com servidor, o mesmo hospeda as funções das aplicações, e o cliente só conta com as características de hardware do próprio servidor como capacidade de armazenamento, processamento e latência de rede, esse fato se torna um empecilho quando a quantidade de solicitações cresce de forma repentina pois a arquitetura não é escalável e se o sistema não consegue responder às solicitações no tempo programado ele perde também a sua disponibilidade.

A confiabilidade de um sistema é medida com relação à sua disponibilidade e escalabilidade, portanto para suprir esta necessidade é necessário um acompanhamento ininterrupto de analistas para tomar as ações cabíveis para que o sistema esteja sempre disponível. O trabalho necessário para manter os sistemas é grande e isso faz com que os custos envolvidos sejam elevados tanto em infraestrutura quanto em mão de obra. Uma alternativa seria a contratação de uma empresa terceira para hospedar e manter o sistema, o que resolve em parte os problemas de escalabilidade e disponibilidade, porém os custos ainda são elevados em comparação com a computação em nuvem.

A computação em nuvem oferece através da internet um modo de acesso conveniente e quando houver demanda a um pacote de recursos computacionais que podem ser instanciados e liberados rapidamente com contato mínimo com o provedor de serviços (Mell & Grace, 2011). Neste cenário os custos envolvidos são por uso dos recursos oferecidos, o que otimiza o gerenciamento de custos e diminui o valor gasto em comparação às soluções anteriores. O AWS lambda garante alta escalabilidade e disponibilidade contínua para o sistema cobrando pelo número de solicitações e pelo tempo de duração de execução do código. Segundo (Amazon Web Services, 2020) com o lambda é possível desenvolver aplicações sem servidor engatilhadas por eventos.

Base de dados

A organização e o arquivamento das informações são indispensáveis em qualquer sistema, o banco de dados tem esse papel de organizar os dados e permitir buscas otimizadas quando necessário. Segundo (Date, 2004) um banco de dados é um conjunto de dados com persistência utilizados por aplicações em geral. Neste projeto de IoT a persistência e disponibilidade dos dados é fundamental uma vez que são críticos ao projeto. A AWS oferece serviço de banco de dados com o dynamoDB. Segundo o (Amazon Web Services, 2020) o dynamoDB é um recurso de banco de dados NoSQL com suporte a modelos de dados de documento e valor-chave que fornece alta

velocidade de desempenho, previsibilidade e escalabilidade contínuas, o que supre todas as necessidades previstas deste projeto. No DynamoBD cada linha qualquer número de colunas no momento em que for mais conveniente permitindo adaptações rápidas e sob demanda.

2.3. Central de processamento local

As unidades de processamento de plantas industriais, de qualquer tipo de processo, necessitam de robustez, confiabilidade, disponibilidade e segurança uma vez que lidam com a produção da planta. As paradas das linhas nestas plantas, mesmo que por alguns minutos, podem resultar em perdas financeiras significativas. Nas pequenas fábricas não é diferente, a perda de capital em relação ao investimento no caso de parada também é significativa.

Comunicação com a nuvem

A comunicação com a nuvem, onde o back-end da aplicação web foi implementado, é um ponto crítico do projeto. A computação em nuvem é o caminho de evolução natural para as novas tecnologias, com o advento da internet das coisas alguns protocolos de comunicação com a nuvem surgiram. O protocolo *Zigbee* que é um pacote de especificações desenvolvido pela Zigbee Alliance com emprego em IoT e domótica que define camadas subsequentes estabelecidas pelo IEEE 802.15.4, provendo segurança, robustez e conexão com novos dispositivos (Rotta, Charão, & Dantas, 2017) e o 6LoWPAN que permite a transmissão de pacotes IPv6 em redes de baixo consumo de energia. Seu conceito inicial era permitir que os protocolos de internet se estendessem para todos os tipos de dispositivos.

Os protocolos de baixo consumo foram mais utilizados nos últimos tempos para aplicações de IoT. Por possuírem pacotes de mensagens pequenos e gerenciador de mensagens, esses protocolos são ideais para utilização em sistemas embarcados (Stansberry, 2015). O protocolo CoAP definido como um protocolo de transferência web com especialização para funcionamento e emprego com nós restritos em redes IoT (Amâncio, 2018). Projetado para modelo M2M é semelhante ao modelo cliente/servidor do protocolo HTTP. Já o protocolo MQTT definido como um protocolo de modelo publish/subscribe, no qual um dispositivo publica dados de uma das pontas e outro subscreve, lê esses dados em na outra ponta (Correa, Cunha, Almeida, & Moraes, 2016). Com este protocolo é possível implementar um modelo many-to-many onde vários dispositivos publicam em um broker e vários dispositivos subscrevem neste mesmo broker, desta forma não há necessidade de os dispositivos que compõem a rede estarem acoplados.

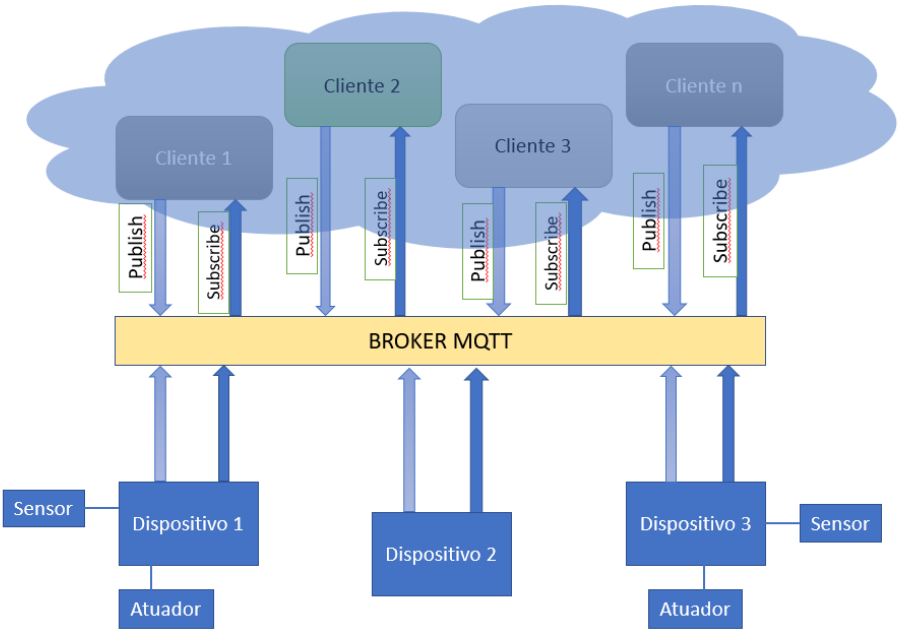

Figura 6 - Estrutura geral do protocolo MQTT

Rede de comunicação local

A comunicação entre os dispositivos no chão de fábrica deve cumprir requisitos básicos como disponibilidade, confiabilidade e robustez. Os padrões de comunicação industrial normalmente cumprem esses requisitos, porém, os dispositivos com suporte nativo a esses protocolos têm valor de investimento elevado para os padrões deste projeto. A plataforma raspberry pi possui suporte para comunicação serial o que possibilita o implemento de protocolos industriais baseados no padrão serial, como modbus RTU sobre o padrão RS-485.

O protocolo modbus é um dos protocolos mais utilizados em automação industrial, graças à sua simplicidade e facilidade de implementação podendo ser utilizado em diversos padrões de meios físicos como por exemplo RS-232, RS-485 e TCP/IP. O padrão de transmissão RS-485 utiliza um sinal diferencial como estratégia de transmissão, o que torna o sinal menos susceptível a ruídos, interferência eletromagnética e variação de terra. A figura 7 mostra como é feita a transmissão diferencial de sinal. Este padrão é multiponto, permite a conexão de até 32 dispositivos a um único enlace. As topologias possíveis são barramento, daisy chain, estrela e anel, figura 8.

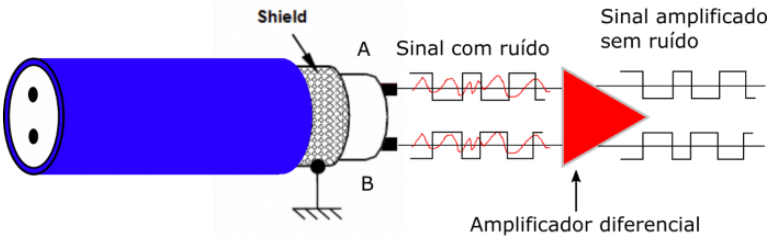

Figura 7 - Transmissão diferencial de sinal

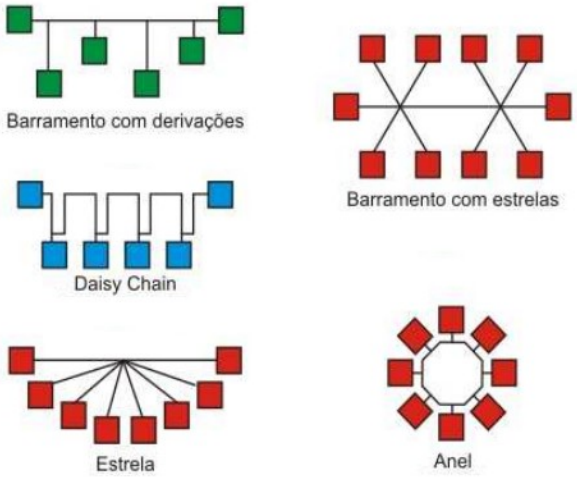

Figura 8 - Topologias de redes industriais

Em redes com poucos pontos ou distâncias pequenas a topologia não apresenta grande influência sobre o desempenho, porém em redes maiores isso deve ser levado em consideração, (Novus, s.d.) e neste caso a topologia mais recomendada é a daisy chain. Segundo (Cunha, 2000) o modelo de transmissão diferencial de dados oferece uma taxa de transmissão mais elevada, o padrão RS-485 permite trabalhar com taxas de 12 Mbps ou até 50 Mbps, dependendo de algumas condições, quanto maior a distância da rede menor e a latência, a distância máxima da rede é de 1200 m. Para evitar reflexão de sinal devem ser ligados terminadores nas extremidades do barramento, normalmente os dispositivos já vêm com esses terminadores.

O protocolo modbus possui dois modos de transmissão de bytes, ASCII e RTU, não é possível utilizar os dois modos de transmissão na mesma rede. O modo RTU é mais disseminado sendo amplamente utilizado em CLPs, inversores de frequência e em diversos sensores e atuadores. No modo RTU, cada mensagem de 8 bits contém dois caracteres hexadecimais de 4 bits. A principal vantagem desse modo é que sua maior densidade de caracteres permite um melhor processamento de dados do que o modo ASCII para o mesmo baudrate (velocidade de comunicação) (Freitas, 2014). A tabela 1 mostra a estrutura do modo RTU. Neste modo não existe um caractere específico que indique o início ou fim da mensagem, esta indicação é feita pela ausência de sinal por no mínimo 3,5 tempos de transmissão. O campo de checagem de erros é baseado no método CRC.

Address	Function	Data	CRC Check
8 bits	8 bits	N x 8 bits	16 bits

Tabela 1 - Payload modbus RTU. Fonte (Freitas, 2014)

Produção de cerveja artesanal

A produção de cerveja artesanal possui várias etapas que são executadas em sequência. A primeira etapa é a moagem que vai desde a escolha e preparação dos grãos à moagem de fato dos mesmos. A segunda etapa é chamada brasagem onde é preparada a água que entrará na mistura e os grãos moídos são combinados e a mistura é aquecida formando o mosto cervejeiro. A terceira etapa é a clarificação onde o mosto é filtrado, lavado e clarificado de forma que não restem partículas no mosto. A quarta etapa é a fervura e lupulagem onde lúpulos são adicionados ao mosto cervejeiro e este é aquecido a uma temperatura pré-definida para cada receita e assim permanecem entre 60 e 90 minutos. A quinta etapa é o resfriamento onde a mistura recém fervida é resfriada até a uma temperatura pré-definida. A sexta etapa é a fermentação onde a mistura de fato se tornará cerveja com a ação das leveduras em um processo que dura de algumas horas a várias semanas e tendo como produto final a cerveja que posteriormente passa por um período de maturação específico para cada receita e depois a cerveja é envasada.

O processo de fermentação, que é o foco principal deste trabalho, consiste em leveduras que consomem os açucares fermentáveis do mosto cervejeiro transformando-o em álcool e gás carbônico, este processo eleva a temperatura e pressão no interior do tanque. A pressão, até certo ponto não interfere no resultado final da receita, portanto uma válvula de segurança resolve o problema de sobrepressão. Cada receita segue uma curva pré-definida de temperatura, sendo extremamente importante que haja controle desta variável no processo. A receita pode ter as suas características alteradas ou até mesmo a produção pode ser completamente perdida caso o controle seja mal executado ou esteja indisponível por um intervalo significativo de tempo.

3. DESENVOLVIMENTO

3.1 Aplicação Web

A aplicação web tem por finalidade apresentar de forma simples, intuitiva e acessível os dados de processo. Desenvolvida com auxílio do framework bootstrap com as linguagens HTML, CSS e javascript possui um design simples, porém compatível com o ambiente semi-industrial ao qual é aplicável. O framework bootstrap tem várias ferramentas que permitem o desenvolvimento de interfaces responsivas que dão à aplicação acessibilidade de qualquer dispositivo com conexão à internet e suporta a navegadores, é possível desta forma ter uma boa experiência de usuário independentemente da plataforma de acesso.

O desenvolvimento da aplicação web foi dívida em duas etapas, a primeira foi a construção do front-end, a interface da aplicação no navegador web. As tecnologias utilizadas foram as linguagens HTML, CSS, JavaScript o framework bootstrap que permite criar projetos com sistemas de grid responsivos e o plugin jQuery. A segunda etapa foi o desenvolvimento de todas as funcionalidades da aplicação, o back-end em nuvem, a integração com o front-end e a integração com a aplicação local. O back-end foi desenvolvido utilizando serviços lambda, S3, DynamoDB e API Gateway, todos oferecidos pela AWS.

Hospedagem da aplicação Web

O principal recurso do AWS S3 que foi utilizado neste projeto é a hospedagem de site estático. O front-end da aplicação web com todas as funcionalidades client-side e arquivos ficam guardados no bucket que, quando configurado para a hospedagem de site estático, providencia uma URL de acesso ao site. Se a requisição respeita as políticas de acesso é possível ver o site estático independente do navegador utilizado. Para hospedar um site estático é necessário criar um bucket com o nome do projeto como na figura 9 abaixo.

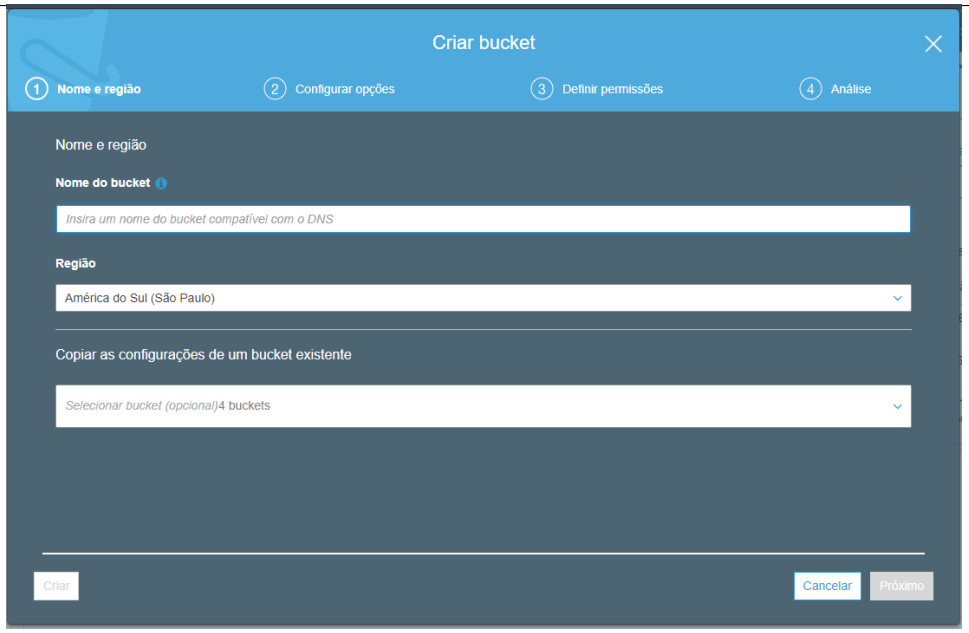

Figura 9 - Criando um bucket no AWS S3

O nome do repositório deve ser exclusivo globalmente, isso resolve todos os problemas de DNS para o site que será hospedado. Após a criação do repositório os arquivos do site estático devem ser carregados. A configuração para hospedagem do site é feita na aba propriedades no recurso hospedagem de site estático como visto na figura 10. Feito isso o endpoint é disponibilizado e o site estático já pode ser acessado por qualquer navegador.

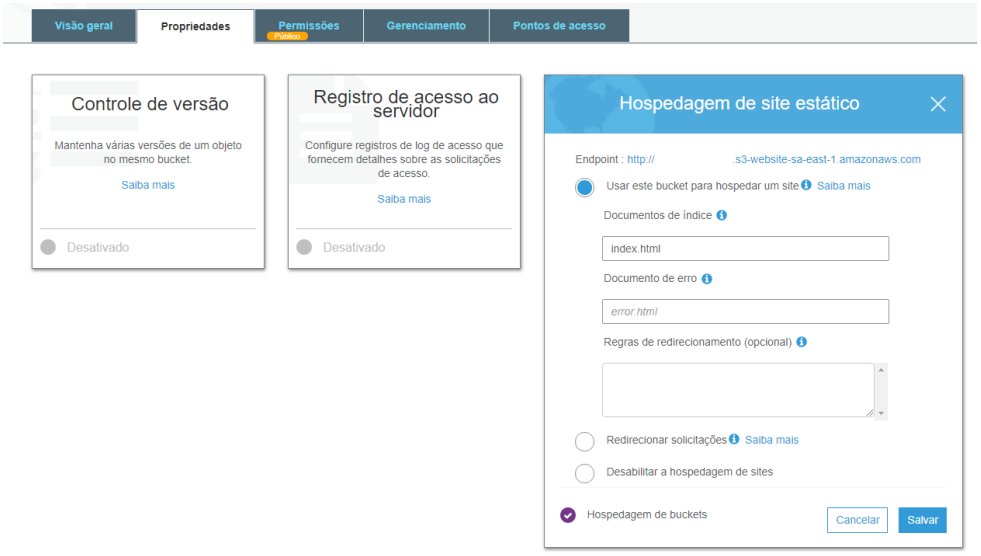

Figura 10 - Ativação de hospedagem de site estático

Seção web

A interface tem duas seções, a supervisão e o controle e um menu interativo na parte superior da interface. A seção de supervisão é a que primeiro aparece quando se acessa a aplicação web e ela foi baseada, para telas acima de 7 polegadas, no layout da fábrica usada como exemplo neste trabalho onde os 4 tanques de fermentação e o banco

glicol ficam perfilados na mesma disposição apresentada na interface, foi feito desta forma para tornar a tela de supervisão mais intuitiva e mais próxima da planta real. A disposição dos tanques muda de acordo com a resolução das telas que acessam a aplicação, efeito foi desenvolvido com ferramentas do framework bootstrap. Na figura 11 são mostrados apenas dados referentes à supervisão do sistema. Não é possível interagir com os elementos nesta seção da interface apenas acompanhá-los.

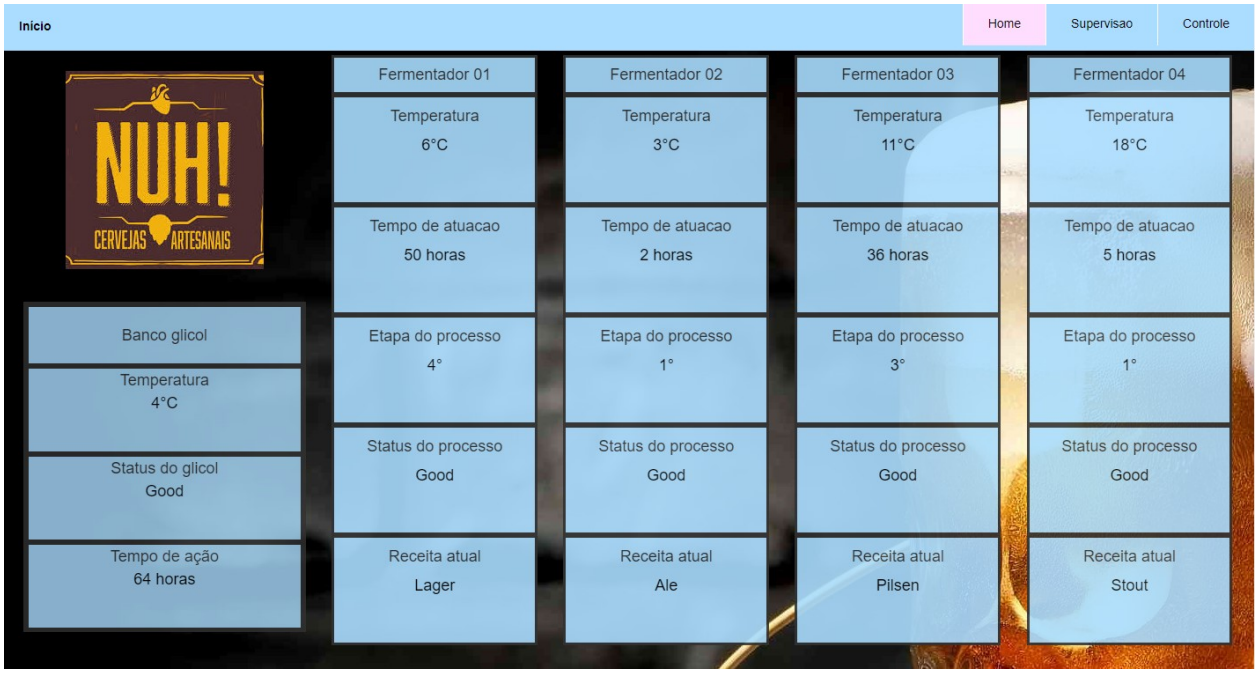

Figura 11 - Interface de supervisão da aplicação web

A segunda seção é a interface de controle que foi baseada em um painel de controle físico, com alguns designs mais modernos, para facilitar o uso dos controles do mesmo e a visualização dos dados e parâmetros da planta. Nesta seção, que pode ser vista na figura 12, estão os elementos de controle, para atuação direta no processo. Para interagir com os controles basta tocá-los, para dispositivos touch screen ou clicar sobre eles no restante.

Figura 12 - Interface de controle da aplicação web

Nesta seção é possível interagir com os controles e foram desenvolvidos mecanismos de intertravamento para impedir ou prevenir comandos indesejados, uma vez que podemos encontrar esses problemas em telas touch-screen. Os mecanismos de intertravamento foram desenvolvidos em linguagem javascript, o primeiro deles é que para realizar qualquer alteração no sistema é preciso que o modo de execução esteja em manual caso contrário o navegador informa através de um pop-up que não é possível fazer alterações como é mostrado na figura 13. Outro mecanismo de segurança fica nas chaves das válvulas onde o navegador pede uma confirmação no ato da alteração, figura 14.

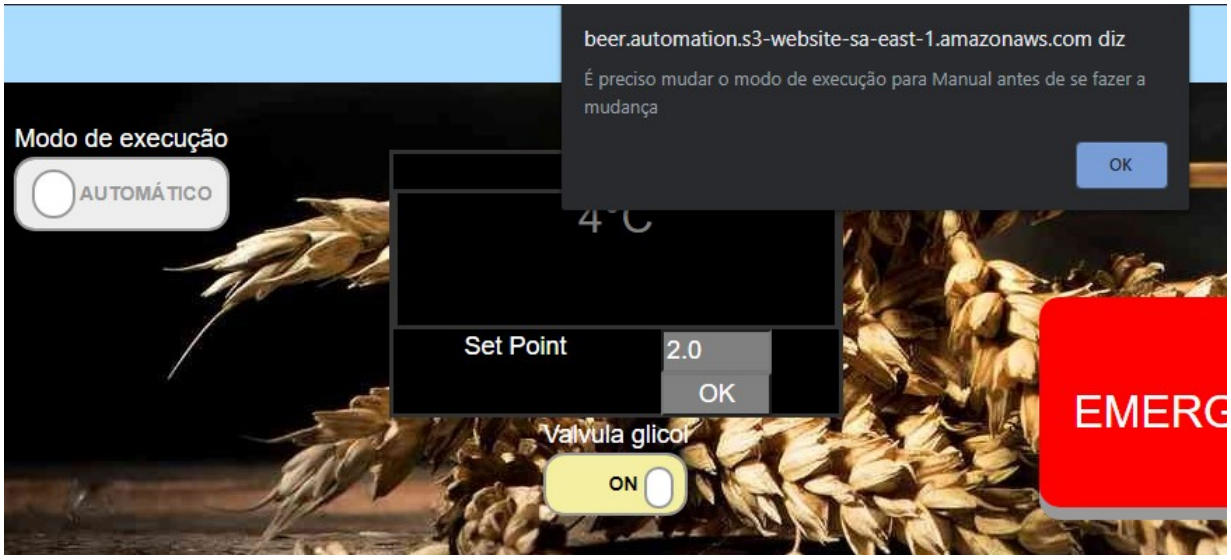

Figura 13 - Mensagem de alerta para alteração do modo de execução

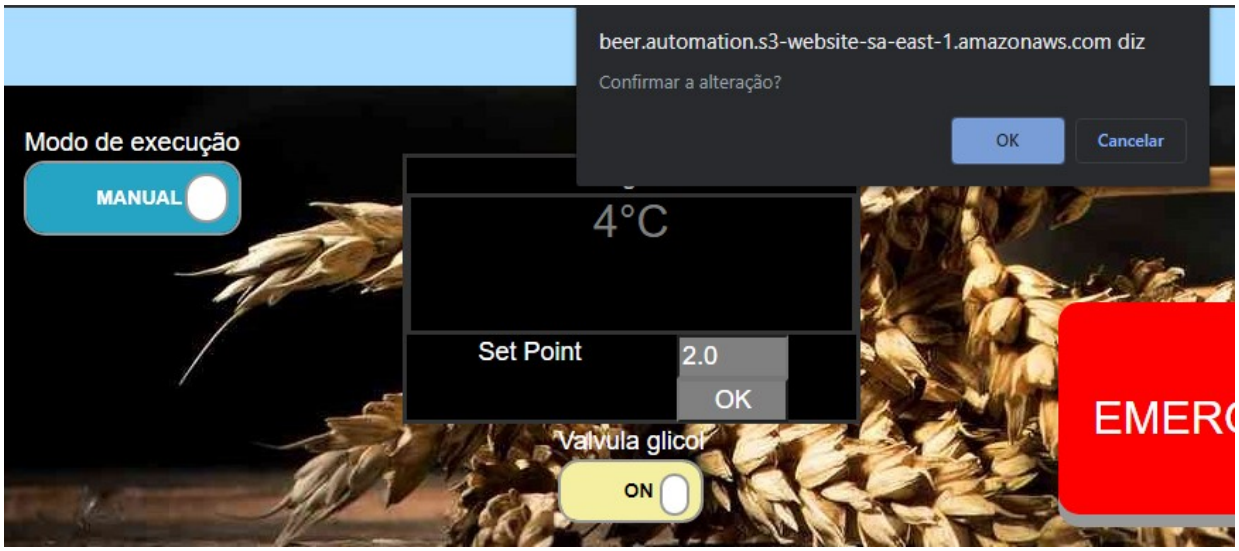

Figura 14 - Mensagem de confirmação de alteração nas chaves

Interface de programação de aplicação

As requisições de API seguem o conceito ajax que, segundo (Marques, 2016) é um pacote de técnicas em javascript utilizado para carregar informações de forma assíncrona sem interromper o fluxo de código o que permite que os valores se atualizem na tela sem a necessidade de carregar a página novamente pois o seu uso não interrompe o fluxo de execução de código, recurso que traz leveza à aplicação web. Para utilizar o conceito ajax bastou importar a biblioteca JQuery que permite o uso de funções com estrutura simplificada para trabalhar com as requisições assíncronas. As respostas das requisições são tratadas de forma independentes pro funções de callback. Foi feito desta forma para deixar a aplicação mais leve e imersiva uma vez que as informações de supervisão se atualizam a casa 30 segundos. A ferramenta API Gateway da AWS foi utilizada para criar e implementar as APIs utilizadas neste projeto, a figura 15 mostra a estrutura geral do API Gateway.

Figura 15 - Estrutura geral do API Gateway

As APIs utilizadas neste projeto foram desenvolvidas com a arquitetura API REST, foi feito desta forma pois a arquitetura REST é composta de um conjunto de diretrizes que podem ser implementadas sob demanda tornando as APIs REST mais leves e rápidas, requisitos importantes para aplicações de supervisão e IoT. A fim de testar o protótipo foram desenvolvidas 3 APIs REST, cada uma com a função de integrar diferentes recursos do back-end com o front-end. Abaixo é possível visualizar as API utilizadas e suas funções:

- pcontrole: Envia as informações da seção e controle da aplicação web para o servidor, ligada à função Beer_controle;

- Monitora: Envia os dados de supervisão da planta para a interface web, ligada à função Supervisão;

- bcontrole: Busca os estados atuais de controle na planta e os envia para a interface web, ligado à função Get_Controle.

Base de dados

A criação da base de dados na AWS é simples o que torna desenvolvimento mais fácil para quem não domina as linguagens de bancos de dados. Para criar uma tabela é necessário navegar até a plataforma dynamoDB, escolher o botão criar tabela, inserir a chave primária e se necessário a chave de partição e clicar em criar. A figura 16 mostra a interface de criação de tabelas.

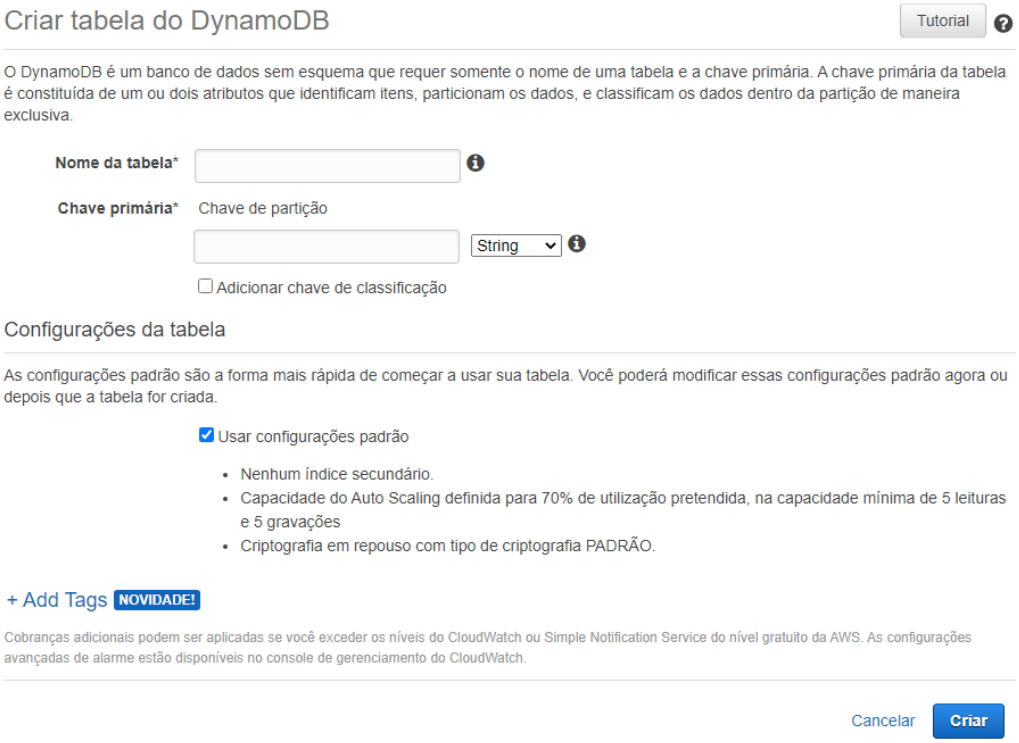

Figura 16 - Interface de criação de tabelas no DynamoDB

Neste projeto foram criadas duas tabelas, uma para os dados de supervisão e outra para os dados de controle nomeadas supervisão e controle respectivamente. A tabela de supervisão possui como chave primária o ativo, ou seja, a identificação de cada tanque na fábrica, as demais colunas da tabela são etapa, receita, status, temperatura e tempo de execução. A tabela de controle possui como chave primária a posição, ou seja, válvulas, chave de modo ou setpoint de controladores. As colunas dessa tabela são os estados de cada válvula, os setpoints de cada controlador e o estado da chave modo de execução. O acesso à base de dados é feito por funções lambda executam ações de leitura e escrita conforme programadas.

Funcionamento da seção web

A seção web executa todas as suas ações em nuvem. Quando algum controle tem o seu estado alterado pela interface web essa alteração faz a chamada para uma função javascript que monta uma estrutura JSON e a envia pelo método HTTP ajax POST, pela API que das informações de controle. Quando a API é acionada ela envia a estrutura JSON para uma função do AWS lambda, no caso a função Beer_controle que recebe uma mensagem com a estrutura JSON e a decodifica. Após decodificar a mensagem a função insere os estados na tabela do DynamoDB. No momento em que ocorre uma alteração na base de dados a função Envia_Controles é acionada e publica via protocolo MQTT no tópico de controle os dados alterados que posteriormente serão subscritos pelo sistema local.

O protocolo utilizado neste projeto foi o MQTT. Por ser um protocolo no modelo *Publisher/*subscribe e necessitar de um *midleware* chamado broker seu desenvolvimento do zero é bem trabalhoso, mas por ser um protocolo confiável e com baixo gasto de energia várias ferramentas foram desenvolvidas a fim de tornar o seu uso viável do ponto de vista computacional. A AWS oferece o serviço AWS IoT core que oferece suporte para a criação e gerenciamento de "coisas" se comunicando sobre o protocolo MQTT, desta forma o desenvolvedor só tem o trabalho de "criar" o que for necessário usando as ferramentas na plataforma da AWS. Segundo (Amazon Web Services, 2020) o IoT Core é um recurso de nuvem gerenciada onde é possível se conectar de forma simples e com segurança os dispositivos da rede à própria nuvem.

O protocolo MQTT vem sendo mais difundido nos últimos tempos por conta das ferramentas que facilitam o seu uso. Neste trabalho como o dispositivo utilizado não é preparado para esta comunicação foi usado um SDK de dispositivo da AWS para a linguagem python. Este SDK possui todas as ferramentas e funções necessárias para a comunicação, desde que o dispositivo tenha acesso à internet e os certificados estejam no dispositivo no local correto. Por ser desenvolvido e disponibilizado pela própria Amazon é possível ter segurança de que este conjunto de ferramentas mantém as especificações e funcionalidades oferecidas pela AWS.

A segurança dos dados é um requisito básico em qualquer sistema de automação, principalmente aqueles que lidam com produção de bens. O serviço da AWS possui um sistema de configuração e autenticação automatizadas, na primeira conexão de um dispositivo, e criptografia em todos os pontos de conexão, a figura 17 dá uma visão geral do sistema de segurança. Este esquema barra qualquer troca de dados entre dispositivos sem uma identidade comprovada através das políticas e certificados gerados no servidor.

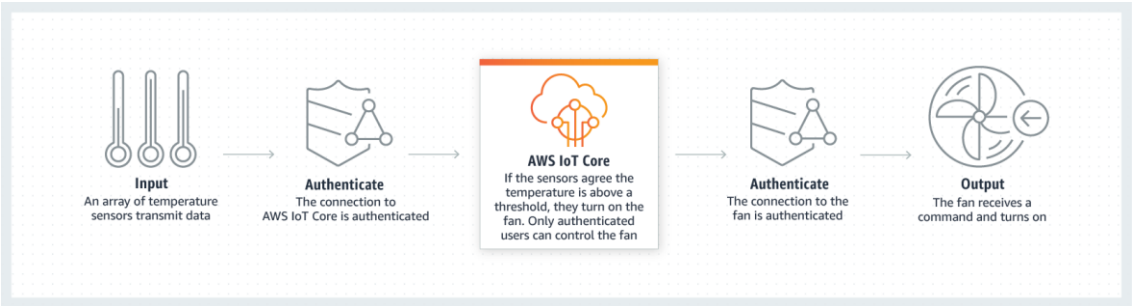

Figura 17 - Visão geral do sistema de segurança das conexões IoT

O serviço IoT core tem integração com os outros serviços oferecidos como o Lambda, isso permite que o processamento das informações enviadas ao broker seja feito nuvem sem nenhuma outra camada visível de comunicação. Para criar um dispositivo ou coisa basta, na página inicial escolher a seção 'gerenciar', depois 'coisa',

‘criar uma única coisa’ e nomear o dispositivo. Este processo é mostrado nas imagens abaixo.

Figura 18 - Criar coisas no IoT Core parte 1

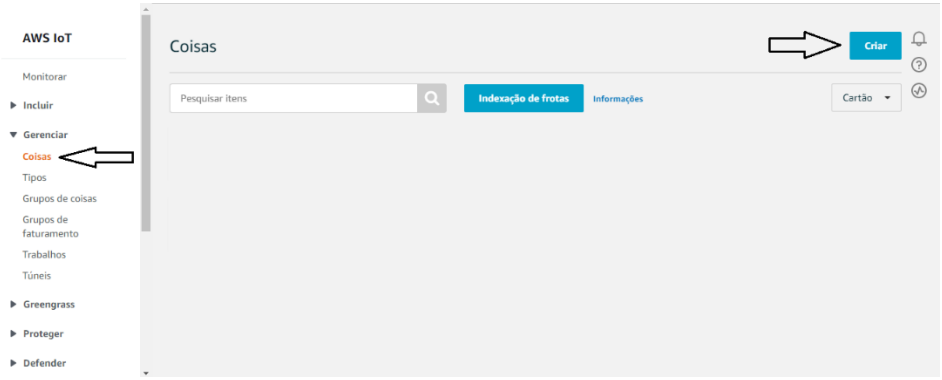

Figura 19 - Criar coisas no IoT Core parte 2

Figura 20 - Criar coisas no IoT Core parte 3

A comunicação desta coisa só é realizada com dispositivos previamente

registrados, o registro a e autenticação é realizada através de certificados de dispositivos criados para cada dispositivo que deve se comunicar com a coisa criada. Para se registrar um dispositivo é necessário seguir os seguintes passos.

- Criar e ativar um certificado

Para isso é necessário ir à aba 'Proteger', 'Certificados', e escolher a opção 'criar certificado'. Este processo pode ser visto na imagem abaixo. Após a criação dos certificados X.509 é necessário fazer o download de todos os arquivos gerados. Criar uma politica

Figura 21 - Criar um certificado de autenticação para dispositivo

É necessário criar uma política de uso para os dispositivos pois, segundo (Amazon web Services, 2020) "os certificados X.509 são usados para autenticar o dispositivo com o AWS IoT Core. As políticas do AWS IoT Core são usadas para autorizar o dispositivo a realizar as operações do AWS IoT Core, como assinar tópicos MQTT ou publicar neles. Para criar uma política é necessário, ainda na aba proteger, clicar em 'Políticas', 'Criar' e preencher os campos apresentados. O campo ação determina o que o dispositivo com aquela política pode fazer como se conectar, publicar ou subscrever. É possível restringir quais dispositivos podem se conectar pelo campo 'Recurso ARN' inserindo o ARN dos dispositivos desejados. Ao fim é necessário marcar o campo 'Permitir' e clicar em criar para finalizar a criação da política.

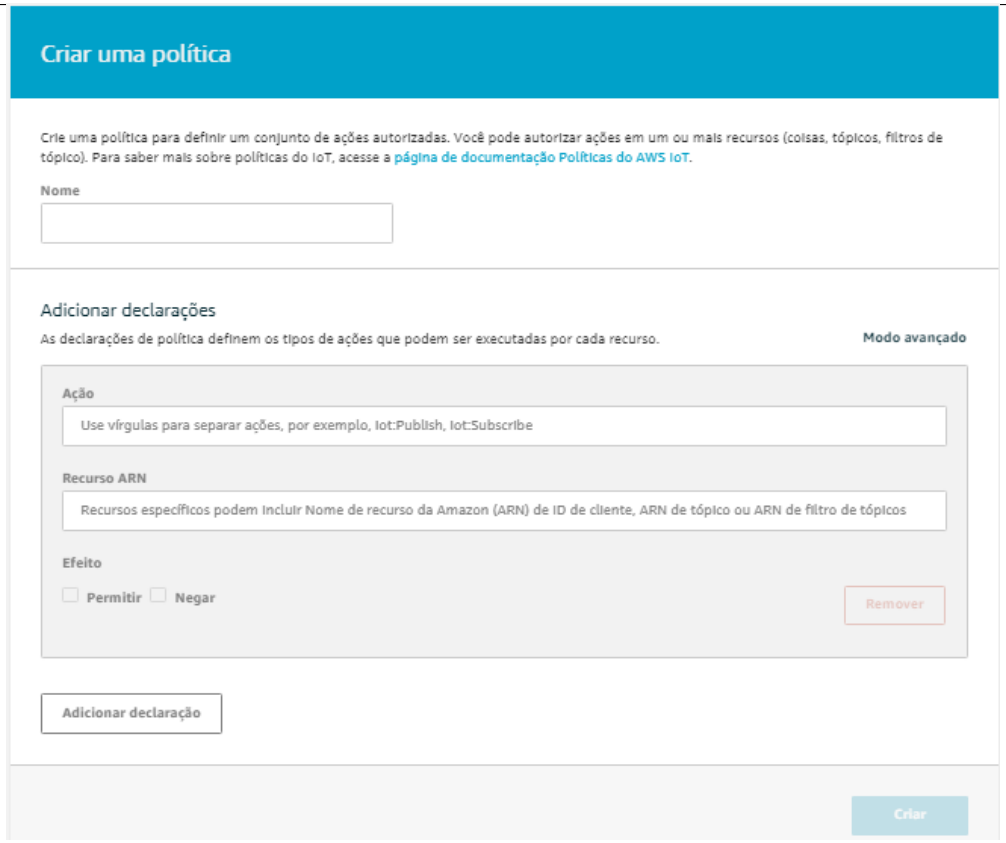

Figura 22 - Criar uma política de ações para um dispositivo

Para que o dispositivo tenha as permissões especificadas na política é necessário anexar a política ao certificado do dispositivo. Na aba 'Certificados' selecione a política, clique em 'Ações', 'Anexar política' escolha as políticas necessárias e clique em 'Anexar'. A imagem abaixo mostra este processo.

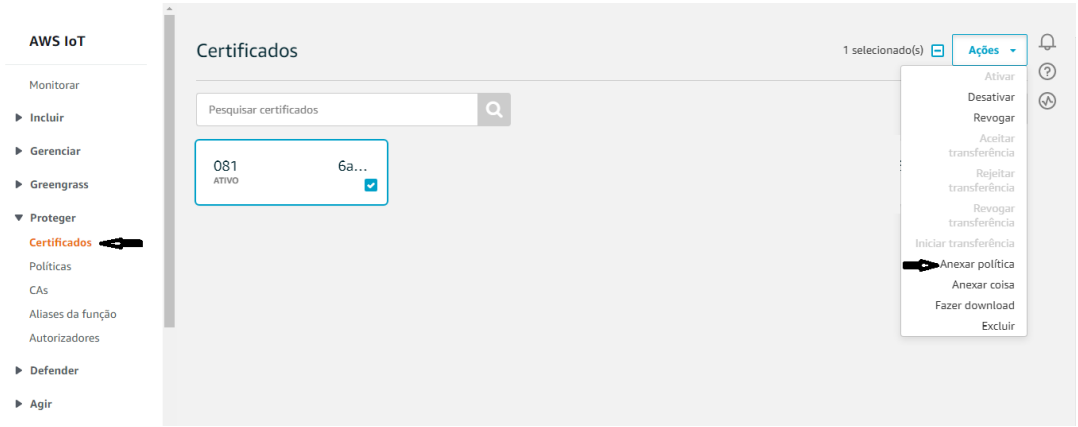

Figura 23 - Anexar uma política ao certificado de dispositivo

- Anexar o certificado ao dispositivo desejado

De forma análoga ao processo de anexar uma política ao certificado é o processo de anexar uma coisa ao certificado, isso é feito pois permite criar políticas com permissões baseadas nos certificados de cada coisa.

A criação da coisa, suas políticas e certificados são a maior parte do trabalho para estabelecer uma comunicação segura entre o dispositivo e a nuvem. Os certificados que foram gerados e baixados agora devem ser ineridos no dispositivo que irá se comunicar com a nuvem, é necessário que o dispositivo possua suporte para comunicação sob protocolo MQTT para a comunicação acontecer, caso o dispositivo não possua essa função nativamente, como é o caso do dispositivo utilizado neste trabalho, o uso de ferramentas que possibilitem este processo deve ser feito.

A AWS possui outros serviços de internet das coisas como o IoT Greengrass que possui arquitetura de computação de borda. O IoT Greengrass permite a execução de recursos locais de computação isso permite que as coisas operem mesmo com conexão intermitente com a nuvem, quando a conexão é feita os dispositivos são sincronizados. A desvantagem do IoT Greengrass em relação ao IoT core está na segurança da informação. O IoT Core conta com os mecanismos de segurança em nuvem que protegem os dados à medida que eles se movem e oferece canais de transporte seguro usando o certificado TLS. O TLS é um protocolo de segurança que permite que as atividades realizadas no ambiente aplicados estejam protegidos contra ataques de hackers e vazamento de informações, requisito muito importante quando se trata de segurança das informações de segurança.

3.2 Sistema local

O sistema local foi divido em duas partes no seu desenvolvimento. Uma parte é o sistema de controle e supervisão local desenvolvido na linguagem de programação python-3.7 e embarcado na placa raspberry pi. A outra parte foi a rede local implementada sob o protocolo modbus, para servir à rede como dispositivos remotos foram utilizadas placas Arduino UNO R3 e alguns periféricos, que são abordados mais a frente neste trabalho, em conjunto com bibliotecas de comunicação modbus. A integração entre o sistema local e o sistema web é realizada via protocolo MQTT, para isso foi criada uma estrutura de tópicos para a troca de dados pelo broker MQTT.

Interface do sistema local

A interface do sistema local foi desenvolvida em linguagem python3 com o framework PyQt5. Foi feito desta forma pela facilidade de uso do QT Creator, programa que permite a criação de interfaces gráficas no modo arrasta e solta. Este programa gera códigos de interface gráfica .ui que podem ser convertidos para códigos .py usando ferramentas do framework já mencionado. Os elementos da interface gráfica foram dispostos para que a aplicação local ficasse bem parecida com a aplicação web, isso para facilitar a operação em ambas as plataformas.

O sistema deste projeto foi desenvolvido uma placa Raspberry Pi model B Revision 2.0, qualquer versão superior e esta é suficiente para implementar o sistema. A placa possui suporte a sistema operacional Linux, o que simplifica o desenvolvimento do sistema por conta do número de ferramentas já prontas que podem ser usadas pelo desenvolvedor. A placa conta com processador ARM, interfaces de comunicação Ethernet, UART, SPI e I2C além de acesso Wi-Fi, pinos com função PWM, pinos digitais e pinos GND, 3V3 e5V, onde a corrente drenada não pode ser superior a 16mA por pino, ou 50 mA por toda a placa. A placa não possui suporte nativo para comunicação em protocolos industriais, mas é possível usar a interface serial UART junto com um módulo conversor para o meio físico RS-485 e assim implementar uma comunicação modbus RTU.

A central de processamento também cumpre o papel de sistema supervisório local, uma aplicação foi desenvolvida em linguagem python com o framework PyQt5. A aplicação conta com uma tela de supervisão, semelhante à tela da aplicação web figura 24. Nesta tela é possível visualizar todos os dados monitorados do processo. A tela de controle da aplicação local pode ser vista na figura 25 por ela é possível configurar a referência de cada tanque, atuar diretamente sobre as válvulas caso ocorra algum imprevisto e acionar a emergência caso se faça necessário. Ao se iniciar essa aplicação providencia a conexão com o servidor via protocolo MQTT, porém não é dependente dela para funcionar, desta forma caso ocorra algum problema com a internet o processo pode continuar normalmente.

Figura 24- Interface de supervisão da aplicação local

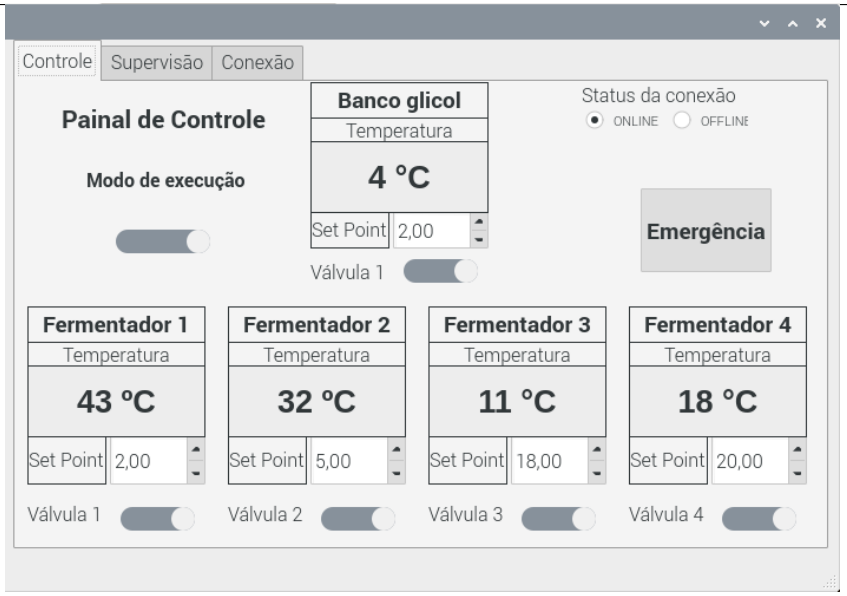

Figura 25 - Interface de controle da aplicação local

O aplicativo trabalha com persistência de dados, ou seja, sempre que reinicia, os estados anteriores à parada são buscados na nuvem e logo após essa atualização o aplicativo volta a ler os dados atuais da planta. A persistência faz com o sistema sempre leve em consideração os dados anteriores, como tempo de execução, referência e receita quando reiniciado. A conexão com a rede local também é providenciada durante a inicialização da aplicação, caso ocorra algum problema de conexão é possível acessar o arquivo de *log* gerado em cada execução para verificar o possível motivo da falha. Na aplicação há um indicador de status da conexão, figura 26, desta forma é possível eliminar possíveis causas de mal funcionamento.

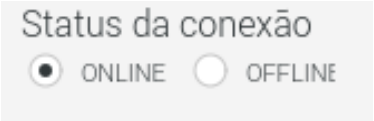

Figura 26 - Barra de status de conexão modbus

Comunicação entre o sistema local e o sistema web

O protocolo utilizado para a comunicação entre o sistema local e o sistema web foi o protocolo MQTT que funciona no modo Publisher/subscribe e conta com um middleware chamado broker onde as publicações e subscrições podem ser feitas. O protocolo foi implementado no sistema local com o auxílio de um SDK python3 disponibilizado pela própria AWS. Este SDK possui ferramentas que facilitam o processo de conexão, autenticação e transferência de dados com o broker, desta forma o único trabalho que o desenvolvedor tem é integrar as funcionalidades deste SDK com a aplicação desenvolvida. A aplicação local publica no tópico de controle sempre que alguma válvula tem seu estado alterado e publica no tópico de supervisão a cada 30 segundos. Os tópicos utilizados são os seguintes:

- CervejariaNuh/fermentadores/controle -> Onde são publicados os dados de controle do processo;

- CervejariaNuh/fermentadores/supervisão -> Onde são publicados os dados de supervisão do processo;

A autenticação das coisas que podem publicar ou se subscreverem nestes tópicos é realizada através de métodos do SDK. Ao se registrar uma coisa no IoT Core são geradas chaves públicas e privadas e também um código de autenticação, esses arquivos devem ser baixados nos dispositivos que são cadastrados como coisas autorizadas a publicar e se subscrever nos tópicos necessários. Para realizar a autenticação, basta informar ao método do SDK o diretório onde estão esses arquivos e como eles estão nomeados, o método faz o restante. Este mecanismo de autenticação de coisas trás proteção tanto aos dados trocados que não podem ser lidos por coisas não registradas quanto ao processo pois garante que o sistema local não irá receber comandos de uma coisa não registrada.

As alterações que ocorrem na planta física seguem um caminho parecido com os dados de alterações da interface web, porém no sentido contrário. Quando um estado de controle é alterado no sistema local, um método do python é chamado e monta uma estrutura JSON que é publicada no tópico de publicação de controle. Quando ocorre esta publicação no tópico a função Get_Controle é acionada e então faz a subscrição no broker. A função decodifica a mensagem e insere os novos estados na base de dados do DynamoDB. A cada 30 segundos o sistema web faz uma requisição com o método ajax GET, neste momento a função le_topicos_mqtt é acionada e busca na base de dados os estados atuais de controle, uma função de callback javascript decodifica a mensagem e altera os estados dos elementos no front-end.

Comunicação entre o sistema local e seus periféricos

A comunicação entre o sistema local e seus periféricos é feita sob o protocolo Modbus no padrão RTU sobre o meio físico RS-485. A linguagem python facilita muito o trabalho do desenvolvedor por já existir uma enorme gama de módulos com os mais variados fins, neste trabalho para tornar o sistema local um mestre modbus foi utilizado o módulo pymodbus. Este módulo conta com métodos que permitem uma comunicação modbus de alto nível entre o mestre e seus dispositivos escravos. Desta forma, na linguagem python3 foi possível implementar funções de conexão, troca de dados e tornar o sistema embarcado na placa raspberry pi um mestre modbus.

Os dispositivos remotos da rede formam implementados em placas Arduino UNO R3 com o auxílio da biblioteca ArduinoModbus.h disponibilizada no site da Arduino. Um dos modos de implementação desta biblioteca é o modo de transmissão RTU no padrão serial sobre meio físico RS-485, configuração utilizada neste projeto. Com poucas linhas de código na linguagem utilizada no IDE do Arduino foi possível desenvolver duas funções, uma que lê um registro enviado pelo mestre da rede e atua sobre uma válvula de acordo com o valor digital recebido e outra que quando solicitada realiza a leitura analógica do sensor de temperatura e envia este dado para o mestre da rede. A fim de testar o protótipo estas duas funções já substituem o trabalho que antes era manual na planta, ler a temperatura dos tanques e atuar sobra as válvulas que liberam ou bloqueiam o líquido refrigerante.

As placas Arduino UNO R3 e raspberry pi não possuem suporte nativo para a comunicação sobre o meio físico RS-485, que utiliza um sinal diferencial para a transferência de dados, elas têm suporte para comunicação I2C, SPI e Serial, esta última que pode ser convertida mais facilmente para sinal diferencial. Para realizar esta conversão foi utilizado o módulo conversor RS-485 que faz a conversão bidirecional do sinal, ou seja, ele converte o sinal serial em RS-485 para ser transmitido na rede e o sinal RS-485 vindo da rede para serial podendo assim ser processado pelo sistema embarcado na placa Arduino. O esquema de ligação é mostrado na figura 27 A fim de testar o protótipo foram utilizados cabos individuais de cobre para a conexão entre os elementos da rede.

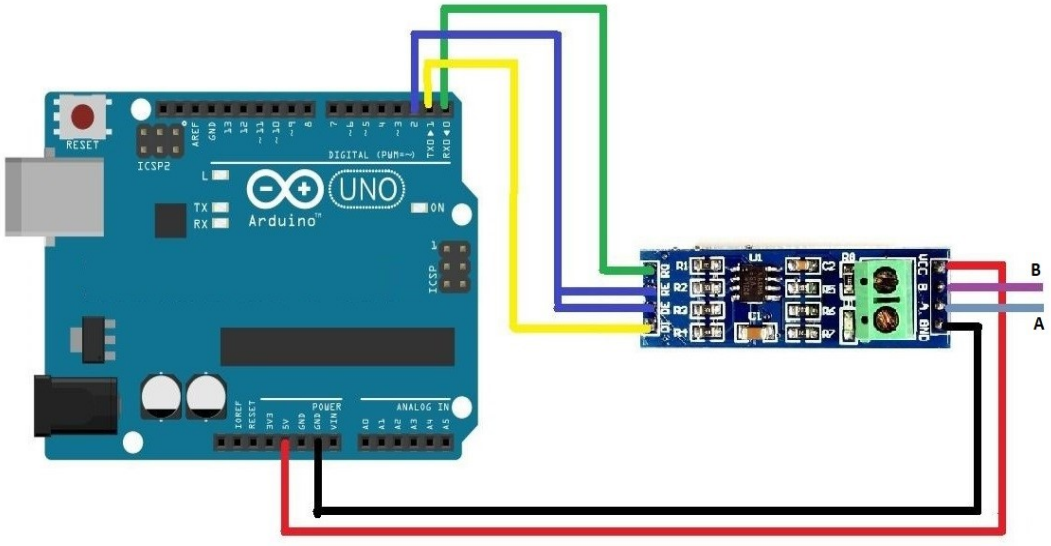

Figura 27 - Esquema de ligação entre Arduino e módulo conversor RS-485

O protocolo modbus trabalha com o modo de comunicação mestre/escravo onde os dispositivos escravos só enviam informações para a rede quando essa ação é solicitada pelo dispositivo mestre, esta estratégia evita problemas de colisão e congestionamento no barramento. O dispositivo mestre é quem faz o processamento

dos dados e, portanto, deve ter uma certa capacidade de processamento, a depender da finalidade. Neste projeto o dispositivo mestre da rede é a placa raspberry pi, que assim como o Arduino não possui suporte nativo para comunicação no padrão RS-485, problema resolvido de forma análoga na raspberry pi. O módulo pymodbus foi utilizado para estabelecer conexão e enviar comandos e ler as respostas no barramento. A topologia escolhida foi daisy chain por conta de sua maior tolerância a erros de reflexão de sinal.

Estrutura de controle da planta

A planta que utilizada neste trabalho possui 4 tanques fermentadores perfilados lada-a-lado e um banco glicol, a figura 28 representa o layout utilizado na planta. Os 4 tanques compartilham o mesmo sistema trocador de calor. Na entrada da camisa de cada tanque há uma válvula solenoide ON/OFF que libera/bloqueia a passagem do líquido refrigerante. A medida da temperatura é feita por sensores resistivos PT-100 instalados na parte inferior dos tanques. O banco glicol passa o líquido refrigerante por um trocador de calor a ar forçado e uma bomba envia o liquido através da tubulação até os tanques de fermentação.

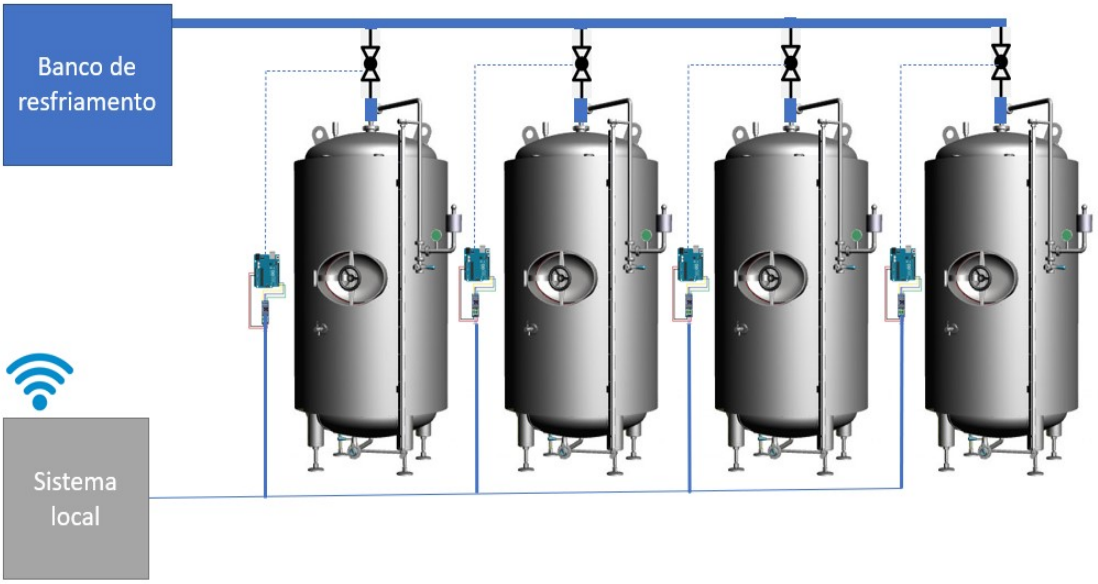

Figura 28 - Representação da disposição de equipamentos na fábrica

No processo de fermentação alcoólica o controle preciso da temperatura é de extrema importância pois sob essa condição as leveduras utilizadas vão produzir o sabor esperado em cada receita (Rosa & Afonso, 2015) . A fermentação alcoólica é um processo exotérmico, porém no Brasil onde a temperatura ambiente que normalmente é muito alta, este é o fator que mais interfere na qualidade de produto final. Ambos os processos, fermentação e temperatura, são lentos e neste caso não é necessário um

sistema de controle muito sofisticado para estabilizá-los. A maioria das receitas seguem curvas de temperatura abaixo do que encontramos naturalmente por aqui, as figuras 29 e 30 mostram receitas mais populares, portanto com o controle atuando apenas no resfriamento é necessário e suficiente para um bom produto final.

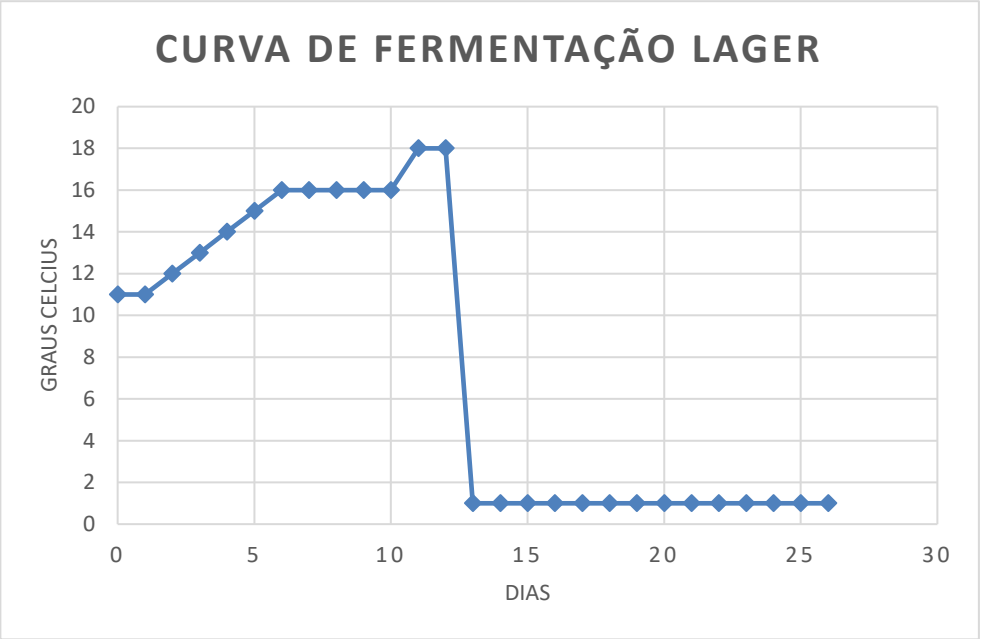

Figura 29 - Curva de temperatura para fermentação de cerveja estilo lager

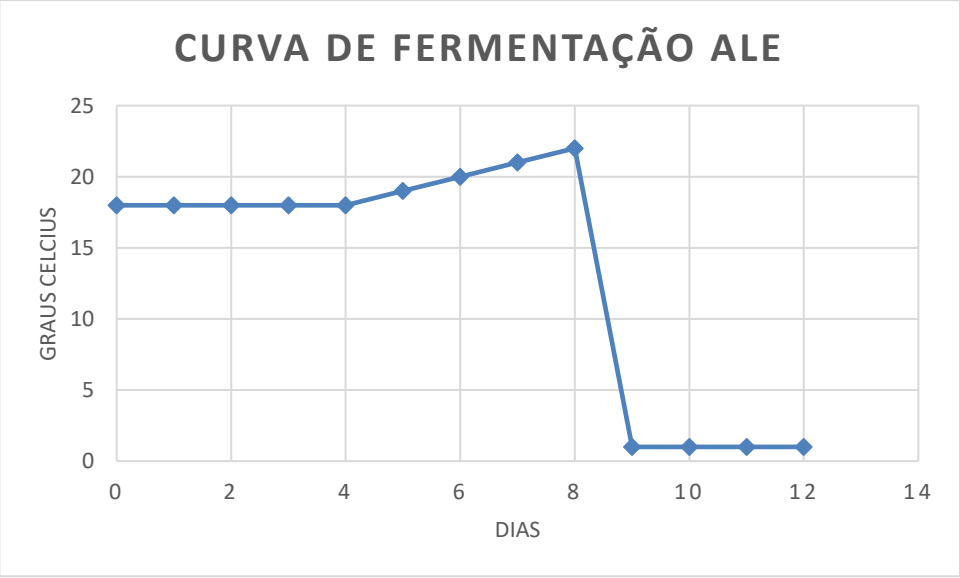

Figura 30 - Curva de temperatura para fermentação para cervejas estilo ALE

Os sistemas de temperatura, com já foi dito, têm constantes de tempo mais altas que outros tipos de sistemas. Os atuadores do sistema usado de base para este trabalho são válvulas ON/OFF, levando isso em consideração a estratégia de controle ON/OFF com histerese foi adotada. Em conversas com produtores que já têm experiência nesta forma certa experiência com essa estratégia de controle foi concluído que o erro de 1 ºC é aceitável para a maioria das receitas. O controle atua da seguinte forma, a válvula é aberta quando o erro assume o patamar de 1 ºC e assim segue até que a temperatura

obtenha um erro de 0,5 ºC abaixo da referência. Como o controle do sistema não é o objetivo deste trabalho este tipo de controle é suficiente para testar este projeto.

4. Resultados e discussões

A aplicação web apresenta uma interface responsiva, que se adapta aos diferentes tamanhos de tela, no apêndice as figuras 31, 32, 33 e 34 mostram a interface em telas de diferentes tamanhos. Desta forma é possível o acesso de qualquer dispositivo eletrônico que possua suporte a um navegador web e com acesso à internet, desde smartphones até microcomputadores. A experiencia do usuário com aplicações web passa pelo desempenho quando é feita a chamada no navegador. Indicadores como velocidade de carregamento, tempo até se tornar interativa, tempo de mudança de layout influenciam diretamente. Existem diversas ferramentas online que mede o desempenho com base nesses e outros indicadores, uma dela é a Google PageSpeed Insights.

Esta ferramenta dá uma pontuação de desempenho de 0 a 100 para acessos de dispositivos móveis e desktop. A tabela 3 mostra os resultados obtidos simulando o acesso por dispositivos móveis, já a tabela 4 mostra os resultados simulando o acesso de desktops. A pontuação de desempenho total para acesso de dispositivos móveis variou em torno de 70, pontuação considerada moderada segundo a ferramenta, já para acessos de desktop variou em torno de 95 pontuação considerada boa. O desempenho pode variar de acordo com a largura de banda da rede utilizada, e do desempenho do dispositivo que faz o acesso.

INDICADOR	TEMPO
Primeiro aparecimento com conteúdo	4,1
Indice de velocidade	5,2
Maior exibição de conteúdo	4,3
Tempo até ficar interativa	4,1
Tempo total de bloqueio	0
Mudança de layout cumulativa	0

Tabela 2 - Indicadores de desempenho para acesso ao app web por dispositivos móveis

INDICADOR	TEMPO
Primeiro aparecimento com conteúdo	0,8
Índice de velocidade	1,3
Maior exibição de conteúdo	1,1
Tempo até ficar interativa	0,8
Tempo total de bloqueio	0
Mudança de layout cumulativa	0

Tabela 3 - Indicadores de desempenho para acesso ao app web por desktops

A alta disponibilidade oferecida pela plataforma AWS foi testada através de

requisições ping feitas de várias partes do mundo, a tabela 5 mostra os resultados obtidos neste teste. A coluna de perda de pacotes dá um panorama sobre a confiabilidade da aplicação, neste teste nenhum pacote foi extraviado ou perdido em nenhuma das requisições, isso significa que a aplicação está disponível em qualquer lugar do mundo situação, como oferece a AWS. A coluna de tempo de resposta que em média os tempos de resposta de todos os locais são abaixo de 300 ms, da mesma forma o RTT, *round trip time* médio, fatos que atestam a alta disponibilidade e confiabilidade da aplicação web.

Local	IP	Perda de pacotes [%]	RTT médio [ms]	Tempo de resposta (ms)
Fremont-CA - US	52.95.164.77	0	182	182
Las Vegas - US	52.95.163.49	0	178	177
Paris - FR	52.95.165.29	0	188	187
Moscow - RU	52.95.165.73	0	230	230
Sao Paulo - BR	52.95.163.69	0	3	3
Santiago - CL	52.95.164.21	0	803	803
Sydney - AUS	52.95.165.69	0	312	311
Johannesburg - ZA	52.95.164.37	0	354	354
Singapore - SG	52.95.164.9	0	349	348
Mumbai - IN	52.95.163.17	0	314	314
Beijing - CHN	52.95.165.29	0	370	370

Tabela 4 - Teste de disponibilidade da aplicação web

A AWS através do IoT Core oferece suporte aos níveis 0 e 1 de QoS. O controle via aplicação web precisa ser confiável, isso significa que os pacotes publicados nos tópicos que são assinados pelo sistema local precisam ser entregues, neste caso o nível 1 de QoS garante a entrega dos pacotes, uma mensagem é enviada pelo menos uma vez e, em seguida, repetidamente até que uma resposta PUBACK seja recebida quando se assina um tópico com nível 1 de QoS. Como essa é uma característica oferecida pelo serviço da AWS não são necessários testes de entrega nesta camada, mais uma vantagem do serviço de computação em nuvem poupando tempo e agregando confiabilidade e segurança ao projeto.

O protocolo modbus no padrão RTU sobre o meio de transmissão RS-485 permite trabalhar com taxas de comunicação que podem chegar a 12 Mbps dependendo da qualidade dos cabos e do comprimento da rede. Afim de testar o funcionamento do protótipo da rede modbus foram realizadas algumas baterias de testes *polling*, ou seja, escritas e leituras cíclicas, em taxas diferentes, ao fim foi verificado quantos pacotes foram entregues normalmente ou perdidos durante a comunicação. As taxas usadas foram 1 solicitação a cada segundo, a cada 5 segundos e a cada segundo. Os tempos escolhidos para este teste condizem com as taxas de requisição necessárias para o bom funcionamento do produto. Os testes mais rápidos foram realizados para verificar a robustez da rede. Com estes testes foi possível verificar que a rede modbus RTU sobre

o meio físico RS-485 funciona, nenhum pacote foi perdido durante o *polling* em nenhuma das 3 taxas utilizadas. O módulo conversor serial-RS-485 cumpre a sua proposta sem nenhuma perda assim como o Arduino cumpre bem o seu papel de dispositivo remoto para este projeto.

Os materiais utilizados para o desenvolvimento deste protótipo têm seu custo baixo quando comparados a equipamentos industriais que cumpririam o mesmo papel no sistema. É possível se encontrar uma Raspberry Pi por cerca de R$ 265,00, em versões mais antigas, porém ainda assim com poder suficiente para integrar este protótipo. Os dispositivos remotos da rede local foram desenvolvidos na placa Arduino uno R3, que pode ser encontrada por cerca de R$ 60,00 e com uso do módulo conversor serial para RS-485 encontrado por cerca de R$ 5,90 cabos e jumpers que são encontrados em pacotes com 100 unidades por R$ 9,60. O interface homem-máquina local usa uma tela touch própria para a raspberry que pode ser encontrada por R$ 250,00. Neste caso o investimento nos materiais que compõem o sistema local seria num total de R$ 784,50.

Os custos operacionais da computação em nuvem oferecida pela Amazon são calculados de forma individual e por tempo de uso em cada serviço. Não existem taxas referentes à infraestrutura utilizada e nem sobre tempo inutilizado, só se paga realmente pelo tempo de uso de cada elemento. AWS tem uma métrica especifica de definição de preço para cada função. Para o funcionamento normal deste projeto os recursos usados são os seguintes:

- Serviço de processamento lambda;
- Serviço de armazenamento e hospedagem S3;
- Serviço de APIs API gateway
- Serviço de banco de dados DynamoDB;
- Serviços de internet das coisas IoT Core;

No AWS lambda a cobrança incide pelo uso sobre cada solicitação das funções e pelo tempo de execução do código. A cada vez que o código inicia sua execução é contada 1 solicitação e o tempo de duração é calculado entre o início e o fim da execução sempre sendo arredondado para os 100 ms mais próximos, o preço pela duração depende da quantidade de memória que é alocada para a execução da função. O back-

end da aplicação web deste projeto faz o uso de 5 funções no lambda, todas alocam 128 MB, que é a quantidade mínima de memória que pode ser alocada. Os valores são mostrados nas tabelas 9 e 10 retiradas de (Amazon Web Services, 2020).

Tipo	Preço
Solicitações	0,20 USD por 1 milhão de solicitações
Duração	0,0000166667 USD por cada GB/segundo

Tabela 5 - Custos do AWS Lambda por solicitações e por duração de execução. ADAPTADO DE: (Amazon Web Services, 2020)

Memória (MB)	Preço por 100 ms (USD)
128	0,0000002083 USD
512	0,0000008333 USD
1.024	0,0000016667 USD
1.536	0,0000025000 USD
2.048	0,0000033333 USD
3.008	0,0000048958 USD

Tabela 6 - Custos do AWS Lambda por tamanho de memória alocada por função. ADAPTADO DE: (Amazon Web Services, 2020)

A funcionalidade de supervisão tem atualização automática, há uma atualização a cada 30 segundos dos dados na interface, neste caso são usadas 2 funções, 2 solicitações com duração de 3 segundos. Se mantida apenas em modo de supervisão, por hora é cobrado o valor de 0,0000365 USD, por mês seria cobrado 0,0263 USD pelo processamento em nuvem. As solicitações de controle utilizam de 2 a 4 funções, a quantidade de solicitações, em caso de funcionamento normal do sistema, seria bem menor que as 120 solicitações por hora da supervisão portanto seu valor nem chegaria ao valor cobrado por mês se o sistema cumprisse apenas o papel de supervisão.

O sistema de armazenamento e hospedagem S3 cobra pelo tamanho dos arquivos armazenado e pelo tempo de armazenamento durante o mês. Os arquivos armazenados no S3 são apenas arquivos do site estático, do *front-end* da aplicação web, neste caso os dados permanecem armazenados por todo o mês. O valor cobrado por até 50 TB/mês é de 0,0405 USD por GB, os arquivos do site estático totalizam 21 MB o que dá um valor de 0,0083 USD por mês pelo armazenamento e hospedagem do site estático.

O serviço de banco de dados DynamoDB cobra pela leitura, gravação e armazenamento de dados em suas tabelas. O DynamoDB tem dois modos de capacidade, cada um com uma forma específica de faturamento, capacidade sob demanda e capacidade provisionada. O foco deste projeto é atender apenas um cliente ou poucos usuários, o número leituras e gravações é baixo portanto foi utilizada o banco de dados com capacidade sob demanda onde é cobrado apenas pelas operações realizadas, leitura e gravação. O armazenamento dos primeiro 25 GB é gratuito caso em que este projeto se encaixa, a tabela 11 mostra os valores cobrados por solicitações.

Tipo de cobrança	Preço
Unidades de solicitação de gravação	1,875 USD por milhão de unidades de solicitação de gravação
Unidades de solicitação de leitura	0,375 USD por milhão de unidades de solicitação de gravação

Tabela 7 - Custos de utilização do DynamoDB. ADAPTADO DE: (Amazon Web Services, 2020)

O sistema de supervisão faz uma operação de gravação e uma operação de leitura a cada 30 segundos neste caso o valor cobrado por mês é 0,162 USD pelas gravações e 0,0324 USD pelas leituras, no total 0,1944 USD por mês com a aplicação web aberta por todo o mês. As operações de controle realizam também realizam 2 solicitações por operação. Sempre que a aplicação local é inicializada são realizadas duas operações para buscar os estados anteriores gravados no banco, estas operações bem como as intervenções no controle são poucas enquanto o sistema estiver em funcionamento normal tornando os valores cobrados pouco relevantes.

O serviço de integração de aplicações API Gateway cobra pela quantidade de chamadas de Api recebidas. Os valores cobrados pelo implemento da API REST utilizada neste projeto podem ser vistos na tabela 12, a aplicação web faz 1 requisição sempre que é inicializada e após isso faz requisições a cada 30 segundos para atualização dos dados. O valor cobrado para manter a interface de supervisão aberta todos os dias durante um mês seria de 0,303 USD, nesta situação ideal não seria necessária nenhuma ou quase nenhuma intervenção de controle pelo sistema web.

Número de solicitações (por mês)	Preço (por milhão)
Primeiros 333 milhões	3,50 USD
Próximos 667 milhões	2,80 USD
Próximos 19 bilhões	2,38 USD
Mais de 20 bilhões	1,51 USD

Tabela 8 - Custos de utilização do serviço API Gateway. ADAPTADO DE: (Amazon Web Services, 2020)

O serviço de internet das coisas IoT Core cobra pela quantidade de dispositivos registrados, pela garantia de disponibilidade chamada de conectividade, pelo sistema de mensagem utilizando o protocolo MQTT e pelo mecanismo de regas que permitem a interação com os outros serviços. A conectividade verifica e garante que os dispositivos estejam conectados em disponíveis 24 por dia e 7 dias por semana, o valor cobrado é de 0,08 USD por milhão de minutos o que resulta em um valor de 0207 USD por mês. O serviço de mensagens cobra 1 USD por bilhão de mensagens, o sistema de supervisão funcionando por todo o mês custaria neste caso 0,0026 USD. O mecanismo de regras cobra o valor de 0,15 USD por milhão de regras acionadas o que totaliza 0,389 USD por mês.

Os custos operacionais com a plataforma AWS podem ser previstos apenas para a funcionalidade de supervisão, pois as intervenções de controle são imprevisíveis, de toda forma é possível se ter uma base dos custos. Pode-se concluir que os custos mensais de monitoramento, com a aplicação web aberta 24 horas por dia e 7 dias por semana seria de 0,921 USD valor que, convertido na cotação do dia 18/09/2020, resulta em aproximadamente R$ 4,93.

5. Conclusão e trabalhos futuros

O principal objetivos deste trabalho foi desenvolver uma solução de baixo custo, porém robusta, para a supervisão e controle de nanocervejarias, de forma mais específica do processo de fermentação de cervejas artesanais. Como objetivos secundários estavam apresentar as ferramentas de desenvolvimento em nuvem e desenvolver uma aplicação web de supervisão e controle da planta, desenvolver uma aplicação local para processamento de dados, implementar uma rede modbus RTU sobre o meio físico RS-485 e um sistema de controle distribuído.

A apresentação das ferramentas de desenvolvimento foi feita ao longo de todo este trabalho mostrando o passo-a-passo em cada ferramenta utilizada e os seus respectivos custos. Os resultados apresentados na análise da aplicação web mostram que ela é disponível em qualquer lugar do mundo, de forma responsiva para qualquer dispositivo que possua um navegador com acesso à internet. A interface não é 'pesada', para uma aplicação com esta proposta, como mostram os tempos de carregamento e resposta. A plataforma AWS garante a segurança dos dados da aplicação e o gerenciamento dos recursos computacionais necessários por um preço bem acessível como o mostrado nos resultados.

O valor de um CLP s7 1200 que tem suporte ao protocolo MQTT, essencial para este sistema, figura em torno de R$ 1850,00, valor esse bem acima do valor de R$ 784,50 do sistema completo proposto neste trabalho. O valor seria maior ainda incluindo os dispositivos remotos, a IHM, o cabeamento e os módulos adicionais que seriam necessários. Os resultados apresentados com relação à rede local e ao funcionamento do sistema embarcado se mostraram suficientes para a proposta, comunicação em taxa relativamente elevadas, confiabilidade nas respostas das requisições e a robustez que o meio físico RS-485 promover por ser um meio de transmissão de sinal diferencial. Portanto é possível concluir que o objetivo principal deste trabalho, propor uma alternativa robusta e de baixo custo para o monitoramento e controle de plantas em pequenas fábricas, foi cumprido com êxitos.

Os resultados obtidos tem como base as características de produção da planta citada no decorrer do estudo, é fato que um sistema industrial convencional com CLPs, IHMs, cabeamento estruturado e tudo mais possui níveis de robustez e confiabilidade bem maiores. O investimento nessas soluções se paga no longo prazo ao contrário da solução apresentada que possui um custo mensal, por menor que seja esse custo ele ainda é algo indispensável. Para plantas com características diferentes como o nível de produção por batelada, quantidade de ativos, tamanho e localização física da planta, necessidades de mais mecanismos de segurança, necessidade de atuação constante no processo o valor da solução certamente será maior podendo até se tornar menos viável que as soluções tradicionais no longo prazo. Essas são situações que devem ser analisadas no planejamento do projeto levando em consideração as particularidades de cada planta e processo.

Como proposta de trabalhos futuros já existe a deixa para atualizar o sistema para monitorar e controlar outras etapas do processo como a chamada parte quente, etapa onde acontece a preparação do mosto cervejeiro que depois irá para a fermentação. Outra proposta para trabalho futuro é agregar inteligência aos sensores de temperatura do processo com o intuito de prever falhas e comportamentos fora do comum no processo. Como o processo é muito sensível a variações de temperatura essa tecnologia seria bem quista para trazer mais segurança aos produtores.

O desenvolvimento deste protótipo deixa espaço para várias propostas de trabalhos futuros, entre elas estão:

- Desenvolvimento de um sistema de alarmes e notificações integrado ao sistema web que notifica o usuário periodicamente ou caso ocorra algum imprevisto no processo. Isto é possível utilizando a ferramenta SNS da AWS;

- Implementação da possibilidade de inserção de novas receitas e alteração de receitas já dispostas na base de dados;

- Projeto de um sistema de controle utilizando métodos mais sofisticados como PID ou métodos de controle moderno para um controle mais eficiente do processo;

- Implementação de mais uma camada de segurança com uso da ferramenta de autenticação e registro de usuários Cognito da AWS

Todas essas propostas de trabalhos futuros são formas de implementar ainda mais a solução sem ter um impacto tão grande no preço da mesma.

6. Referencias

AMÂNCIO, Anderson. "O Protocolo CoAP para IoT". 2018. Disponível em: http://wireengenharia.com.br/br/o-protocolo-coap-para-iot/. Acesso em: 03/09/2020.

AMAZON Web Service. "Guia do desenvolvedor Amazon Simple Storage Service". 2020. Disponível em: https://docs.aws.amazon.com/pt_br/AmazonS3/latest/dev/s3-dg.pdf. Acesso: 13/08/2020.

AMAZON Web Services. "AWS Pricing policies". 2020. Obtido em 10 de setembro de 2020, Disponível em: https://aws.amazon.com/pt/pricing/. Acesso: 10/09/2020.

AMAZON Web Services. "Guia do desenvolvedor Lambda". 2020. Disponível em: https://docs.aws.amazon.com/pt_br/lambda/latest/dg/lambda-dg.pdf. Acesso: 20/08/2020.

AMAZON Web Services. "Guia do desenvolvedor DynamoDB'. 2020. Disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/dynamodb-dg.pdf. Acesso em: 27/08/2020.

AMAZON web Services. (2020). "Guia do Desenvolvedor IoT Core". 2020. Disponível em: https://docs.aws.amazon.com/pt_br/iot/latest/developerguide/iot-dg.pdf. Acesso em: 03/09/2020.

BRIGGS, Dennis, BOULTON, Chris, BROOKES, P. A., & STEVENS, R. Brewing Science and pratice. Cambridge: Woodhead Publishing, 1º ed., Vol. I, 2004.

CONFEDERAÇÃO Nacional da Indústria. "Desafios para a indústria 4.0 no brasil". 2016. Disponível em: https://www.portaldaindustria.com.br/publicacoes/2016/8/desafios-para-industria-40-no-brasil/. Acesso em: 13/08/2020

CONFEDERAÇÃO Nacional da Indústria. "Portal da Indústria".2016. Disponível em: https://www.portaldaindustria.com.br/publicacoes/2016/8/desafios-para-industria-40-no-brasil/. Acesso em: 13/08/2020

CORREA, Remington, CUNHA, Márcio, ALMEIDA, Marcelo, & MORAES, Josué. Simulação de aplicações utilizando o protocolo de comunicação MQTT com aplicações em ambientes industriais. CONFERÊNCIA DE ENGENHARIA ELÉTRICA. Disponível em: https://www.peteletricaufu.com/static/ceel/doc/artigos/artigos2016/ceel2016_artigo108_r01.pdf. out, 2016.

CUNHA, Judson. Protótipo de rede industrial utilizando o padrão serial RS-485 e protocolo modbus. **Universidade federal de Blumenau**. dez,2020

DATE, Christopher. Introdução a sistemas de bancos de dados (8 ed., Vol. I). (D. Vieira, Ed.) Rio de Janeiro: Elsevier. 2004

FREITAS, Carlos. "Protocolo Modbus: Fundamentos e aplicações". 2014 Disponível em: https://www.embarcados.com.br/protocolo-modbus/. Acesso em: 05/09/2020.

GODOI, André, SANTOS, Rafael, CAMPOS, Thiago, & SANTOS, Alexandre, Pesquisa do perfil cervejeiro artesanal caseiro e suas oportunidades. **Anais simpósio de pesquisa e seminário de iniciação científica**, p. 47-60, out, 2016

MARQUES, Rodrigo. "Ajax com JQuery: Trabalhando com requisições assíncronas". 2016. Disponível em: https://www.devmedia.com.br/ajax-com-jquery-trabalhando-com-requisicoes-assincronas/37141. Acesso em: 29/08/2020

MELL, Peter, & GRACE, Timothy. "A definição da NIST de computação em nuvem". 2011 Disponível em: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf. Acesso em: 25/08/2020

NAZARÉ, Tiago, ROCHA, Jéssica, OLIVEIRA, Luiz, SOUZA, Felipe, & RAMOS, Ritler. Os desafios da indústria 4.0 no brasil. **Revista Mythos**, pp. 129-137. doi:10.36674, set, 2019

NOVUS. "Conceitos Básicos de RS-485 e RS-422". 2020. Disponível em: https://www.novus.com.br/downloads/Arquivos/conceitos%20b%C3%A1sicos%20de%20rs485%20e%20rs422.pdf. Acesso em: 04/09/2020

ROSA, Natasha, & AFONSO, Júlio. A química da cerveja. **Química e Sociedade**, 37(2), 98-105. doi:10.5935/0104-8899.20150030, 2015

ROTTA, Giovanni, CHARÃO, Andrea, & DANTAS, Mario. Um Estudo sobre Protocolos de Comunicação para Ambientes de Internet das Coisas. ANAIS DA ESCOLA REGIONAL DE ALTO DESEMPENHO DA REGIÃO SUL (ERAD-RS), p. 387-390, mar, 2017

STANSBERRY, James. "MQTT and CoAP: Underlying Protocols for the IoT". 2015. Disponível em: https://www.electronicdesign.com/technologies/iot/article/21800998/mqtt-and-coap-underlying-protocols-for-the-iot. Acesso em: 02/09/2020

7. Apêndices

A aplicação web foi desenvolvida para apresentar uma interface responsiva acessível de qualquer dispositivo com um navegador web e acesso à internet. As imagens abaixo mostram como fica a apresentação do app web em telas com diferentes tamanhos.

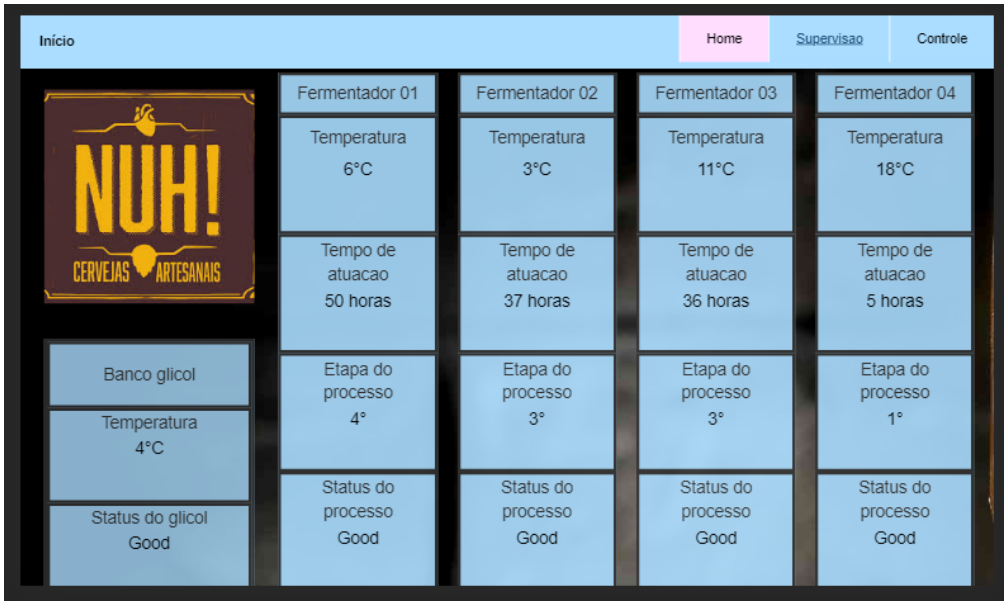

Figura 31 - Interface de supervisão do app web em telas entre 7 e 10 polegadas

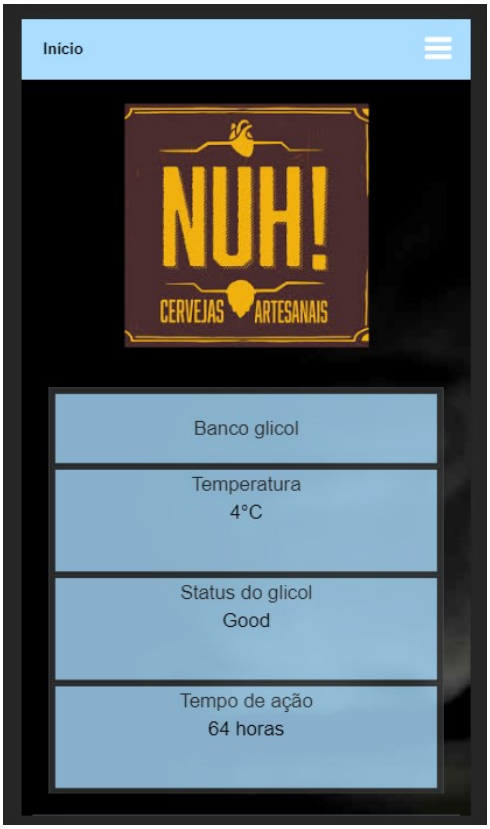

Figura 32 - Interface de supervisão do app web para telas abaixo de 7 polegadas

Figura 33 - Interface de controle do app web para telas entra 7 e 10 polegadas

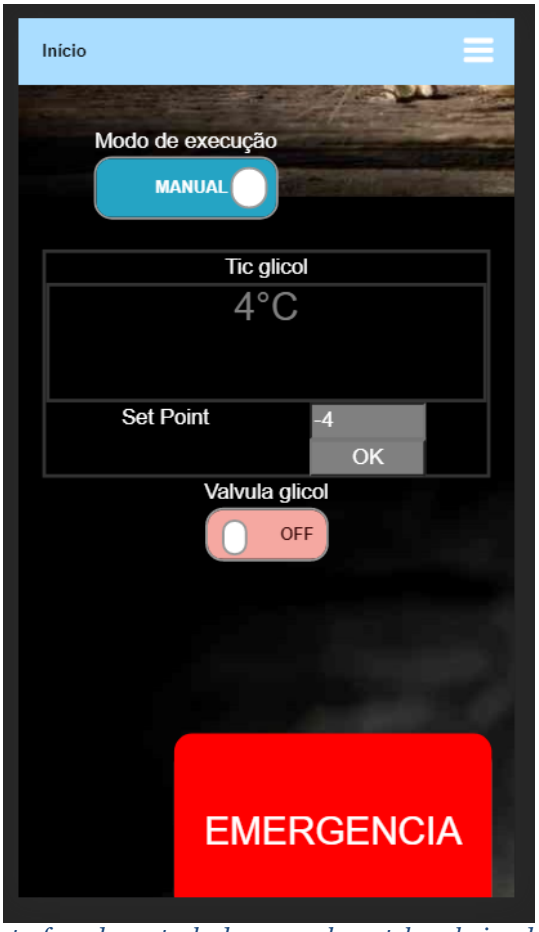

Figura 34 - Interface de controle do app web em telas abaixo de 7 polegadas