UNIVERSIDADE FEEVALE

LEANDRO GABRIEL RASCH

PROPOSTA DE MODELO DE SISTEMA PARA MONITORAMENTO E SEGURANÇA NA ZONA RURAL UTILIZANDO ELEMENTOS DE *IOT*

Novo Hamburgo

2020

LEANDRO GABRIEL RASCH

PROPOSTA DE MODELO DE SISTEMA PARA MONITORAMENTO E SEGURANÇA NA ZONA RURAL UTILIZANDO ELEMENTOS DE *IOT*

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Feevale

Orientador: Prof. Me. Vandersilvio da Silva

Novo Hamburgo

2020

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

Aos meus pais, Bianor e Maria, e meu irmão, Luan, por sempre priorizarem meus estudos e incentivarem minha dedicação.

Ao professor Me. Vandersilvio, pela disposição e pelas sugestões concedidas durante a construção deste trabalho.

Muito obrigado!

RESUMO

Nos últimos anos houve um grande aumento da violência no Brasil, o que tem afetado inclusive a área rural. Especialmente pequenos produtores têm sofrido desde assaltos até outros crimes mais graves, o que coloca em risco a segurança e bem-estar dos moradores de tais localidades. Desta forma surge a necessidade de criar alternativas e soluções que facilitem o dia-a-dia dos pequenos produtores, fornecendo mais tranquilidade, além de automação de processos, aumentando assim a qualidade de vida dos usuários. Este trabalho propõe um sistema de monitoramento para propriedades rurais, permitindo ao usuário acompanhar em tempo real desde a localização de animais até o status de diferentes sensores de segurança espalhados pela propriedade. Através de componentes de IoT (*Internet of Things*) é possível integrar diferentes dispositivos, como o microcontrolador ESP 32, *Web Service* e aplicativos, para que conversem entre si, permitindo tomadas de decisões automáticas e programáveis fazendo uso de redes LoRa e *Wi-Fi*. O protótipo desenvolvido utiliza de diversas tecnologias para integração dos módulos, os quais tiveram resultado efetivo para a proposta, com testes de estresse no sistema que evidenciam bom funcionamento do mesmo, embora com algumas possíveis melhorias de integração.

Palavras-chave: LoRa. Internet das Coisas. Segurança, Monitoramento.

ABSTRACT

In recent years there has been a great increase in violence in Brazil, which has even affected the rural area. Especially small producers have suffered from robberies to other more serious crimes, which puts the safety and well-being of the residents of such locations at risk. Thus, the need arises to create alternatives and solutions that facilitate the daily life of small producers, providing more peace of mind, in addition to process automation, thus increasing the quality of life for users. This work proposes a monitoring system for rural properties, allowing the user to monitor in real time from the location of animals to the status of different security sensors spread throughout the property. Through IoT (Internet of Things) components it is possible to integrate different devices, such as the ESP 32 microcontroller, Web Service and applications, so that they can talk to each other, allowing automatic and programmable decision making using LoRa and Wi-Fi networks. The developed prototype uses several technologies to integrate the modules, which had an effective result for the proposal, with stress tests in the system that show good functioning of the system, although with some possible integration improvements.

Keywords: LoRa. Internet of Things. Security, monitoring.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

1G	Primeira Geração
2G	Segunda Geração
3G	Terceira Geração
AES	*Advanced Encryption Standard*
AMPS	*Advanced Mobile Phone System*
ANATEL	Agência Nacional de Telecomunicações
ARPANET	*Advanced Research Projects Agency Network*
BW	*Bandwidth*
CDMA	*Code Division Multiple Access*
CRC	*Cyclic Redundancy Check*
CSS	*Chirp Spread Spectrum*
EDGE	*Enhanced Data Rates for GSM Evolution*
FDMA	*Frequency Division Multiple Access*
GB	*Gigabyte*
GHz	*Gigahertz*
GPIO	*General Purpose Input/Output*
GPRS	*General Packet Radio Services*
GSM	*Global System for Mobile*
HDMI	*High-Definition Multimedia Interface*
HSDPA	*High Speed Downlink Packet Access*
HSUPA	*High Speed Uplink Packet Access*
HTTP	*Hypertext Transfer Protocol*
IBSG	Internet Business Solutions Group
IEEE	*Institute of Electrical and Electronic Engineers*
IoT	*Internet of Things*
IP	*Internet Protocol*
kbps	Quilobit por segundo
kHz	*Quilohertz*
km	*Quilômetro*
LAN	*Local Area Network*
LoRa	*Long Range*
LORAWAN	*Long Range Wide Area Network*

MB	*Megabyte*
MHz	*Megahertz*
MVC	*Model View Controller*
RAM	*Random Access Memory*
ROM	*Read-Only Memory*
SD	*Secure Digital Card*
SDK	Software Development Kit
SF	*Spreading Factor*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*
TKIP	*Temporal Key Integrity Protocol*
TLS	*Transport Layer Security*
UMTS	*Universal Mobile Telecommunications Service*
USB	*Universal Serial Bus*
WEP	*Wired-Equivalent Privacy*
Wi-Fi	*Wireless Fidelity*
WPA	*Wired Protected Access*

SUMÁRIO

1. INTRODUÇÃO

Ao longo dos últimos anos a quantidade de dispositivos conectados à internet tem crescido de forma exponencial. Segunda Evans (2011), a cada pouco mais de cinco anos a internet dobra de tamanho, o que proporciona uma vasta gama de possibilidades no desenvolvimento de sistemas. Evans diz ainda que há a expectativa de que existam 50 bilhões de dispositivos conectados em 2020, enquanto Carrion e Quaresma (2019) projetam pouco mais de 75 bilhões de equipamentos conectados em 2025. A grande demanda por *hardware* e *software* deve proporcionar fortes investimentos e espaço de trabalho no setor de TI.

A Internet das Coisas (IoT) pode ser descrita de diversas formas conforme descrito por Carrion e Quaresma (2019), porém todas circundam o ideal de dispositivos conectados entre si, enviando e recebendo informações, onde podem auxiliar de forma significativa na automação de processos, além de diversos outros campos. Evans (2011) entende que o “nascimento” da IoT ocorreu entre 2008 e 2009, quando o número de dispositivos conectados em rede superou o número de pessoas no planeta.

Além da grande quantidade de dispositivos, a IoT tem crescido no número de setores de aplicação, atingindo a área médica com monitoramento de sinais vitais, o varejo com o aumento da interação e usabilidade do usuário no estabelecimento, nas residências e indústrias com a automação, no trânsito com semáforos inteligentes, entre outros, possibilitando uma forte redução de custos e maior rapidez das operações (CARRION; QUARESMA, 2019).

A difusão dos meios de comunicação está cada vez mais evidente ao longo dos últimos anos, atingindo inclusive áreas onde antes não era possível, como pontos da Zona Rural, por exemplo. A energia elétrica, tão necessária para grande parte das tecnologias atuais, tem se tornado cada vez mais acessível aos produtores rurais. Da mesma forma as redes de comunicação, sobretudo a internet, têm se alastrado por grande parte do globo, especialmente com novas tecnologias que vem sido aplicadas na área, motivadas pelo grande interesse e investimento relacionado à rede global.

Atualmente já existem sistemas que possibilitam a utilização da grande rede e da IoT para monitoramento e automação agrícola, porém na maior parte das vezes com custos elevados suportados por fazendeiros com propriedades de

médio e grande porte. Por se tratar de projetos com altos custos se torna inviável para pequenos produtores, que não possuem grande volume de produtividade para justificar tal investimento.

O presente trabalho tem por objetivo trazer uma solução de monitoramento e automação rural para pequenas propriedades, fazendo com que produtores possam controlar processos de maneira mais simples, permitindo maior qualidade de vida no campo e eliminando algumas atividades necessárias do dia-a-dia, podendo utilizar o sistema para controle de equipamentos da propriedade remotamente, assim como ativar alarmes que ao serem acionados realize determinadas ações como a gravação de vídeo ou acionamento de uma lâmpada, por exemplo.

De encontro a isso, uma questão cada vez mais crescente no Brasil é a segurança pública no âmbito rural, onde houve aumento da criminalidade nos últimos anos. Os crimes passam desde invasão de terras, roubo de animais, plantações e maquinários, chegando a homicídios e massacres (CARVALHO, 2018). Na Zona Rural, a IoT pode oferecer uma melhora de vida significativa, além de um aumento da segurança da propriedade. Através de sensores pode-se monitorar desde animais até o controle de acessos a determinados setores da propriedade, podendo para isso usar de diversos elementos de *hardware* e *software.*

1.1. OBJETIVOS

1.1.1 Objetivo Geral

O objetivo desta pesquisa é desenvolver um modelo de sistema para monitoramento de propriedades rurais, permitindo tomada de decisões com apoio de dispositivos IoT.

1.1.2 Objetivos Específicos

- Realizar pesquisa bibliográfica quanto a ferramentas de *hardware* e *software* que possam ser utilizados no projeto;

- Levantar requisitos de sistema necessários para atender as necessidades de monitoramento e segurança dos usuários;
- Propor solução de IoT e sistema de monitoramento com base nos requisitos levantados, utilizando soluções de baixo custo;
- Realizar piloto da aplicação juntamente com os dispositivos de IoT;
- Analisar os resultados a fim de avaliar o grau de contribuição do sistema para com os usuários.

1.2. ESTRUTURA DO TRABALHO

Este trabalho está estruturado da seguinte forma: primeiramente, no capítulo 2 é apresentada a metodologia utilizada no desenvolvimento, seguida pela apresentação da revisão da bibliografia quanto à internet das coisas no capítulo 3, sinalizando a origem da mesma e em quais áreas é aplicada nos dias de hoje. Em seguida, no capítulo 4, são abordadas redes de computadores sem fio, abrangendo redes *Wireless*, LoRa e redes celulares. No capítulo 5 são apresentadas plataformas de IoT, com maiores detalhes a respeito do Raspberry Pi e o ESP 32. No capítulo 6 é apresentado o MQTT, protocolo de comunicação amplamente utilizado em aplicações de Internet das Coisas. No capítulo 7 é realizada a discussão a respeito de sensores e atuadores a fim de entender as classificações destes e sua importância em sistemas de IoT. A fim de finalizar a revisão bibliográfica, no capítulo 8 são apresentados conceitos cruciais de análise de sistemas, ponto essencial no projeto e que trará contribuições em uma melhor usabilidade da plataforma ao usuário. O capítulo 9 apresenta trabalhos relacionados a este projeto, enquanto o capítulo 10 é constituído pela descrição completa do projeto e como este está estruturado. O capítulo 11 traz as percepções obtidas e resultados, enquanto o capítulo 12 conclui o texto, vinculando alguns pontos do projeto com a bibliografia.

2. METODOLOGIA

O presente trabalho apresenta uma metodologia experimental quanto aos procedimentos, conforme Prodanov e Freitas (2013), uma vez que utilizará dos conhecimentos obtidos através das pesquisas para criar uma solução ao problema proposto, sendo que estes conhecimentos são obtidos a partir de uma pesquisa bibliográfica, utilizando uma ampla base de materiais já publicados.

Por ser utilizada uma metodologia dedutiva a pesquisa é realizada com início no geral e término no particular, partindo assim de leis e princípios tidos como verdadeiros e aplicação dos mesmos no problema específico de uso da rede LoRa em pequenas propriedades rurais. As premissas em relação à IoT e as especificações gerais das redes a serem utilizadas são utilizadas para o desenvolvimento específico dos dispositivos para pequenas propriedades rurais.

Tendo em mente o desenvolvimento prático do protótipo a fim de solucionar um problema de interesse social trata-se de uma metodologia aplicada no ponto de vista de sua natureza.

Para o desenvolvimento do protótipo do projeto houve a criação primeiramente de um *firmware* para comunicação LoRa entre o *gateway*. Além desta foram desenvolvidas duas aplicações *Java*, sendo uma o *Web Service* e um aplicativo *Android*. Para conclusão desta etapa foi também necessária a pesquisa por bibliotecas e componentes que possam complementar os dispositivos e permitam que a comunicação seja feita de forma eficaz. Esta etapa, feita em laboratório, não possui como requisitos testes de funcionamento a longas distâncias ou validação das informações trafegadas pela rede.

Em um segundo momento houve um aprimoramento do *firmware* com tratamentos de erros na comunicação, com controle dos pacotes enviados e garantia do recebimento da informação pelo receptor. Neste ponto também são validadas as informações enviadas por ambas as partes a fim de garantir a autenticidade dos dados expostos.

Com a rede funcionando adequadamente, os esforços foram voltados à utilização de sensores no cliente e envio das informações coletadas ao equipamento central. A título de testes foram utilizados os sensores de umidade e temperatura para sensoriamento e monitoramento, enquanto um relé foi incluso para permitir a utilização de outros dispositivos.

Posteriormente foi também realizada a conexão com servidores externos, onde é conectada uma interface de monitoramento e tomada de decisões.

O desenvolvimento da interface foi feito levando em consideração a realidade do autor, morador da zona rural do município de Sapiranga / RS.

O piloto foi realizado considerando situações reais de funcionamento, intempéries e distâncias comuns em pequenas propriedades. Além do monitoramento feito a partir dos sensores, houve a realização de testes com atuadores conectados ao cliente, simbolizando ações que podem ser feitas remotamente.

Ao longo do uso da funcionalidade algumas melhorias se mostraram essenciais para o bom funcionamento do sistema, as quais foram realizadas e descritas ao longo do texto.

Os resultados do protótipo também foram avaliados pelo autor em diversos aspectos, como usabilidade, confiabilidade dos dados, taxa de resposta de comandos, amplitude do sinal e outros critérios.

3. INTERNET DAS COISAS

O cientista e matemático Alan Turing, em 1950, defendeu a possibilidade de existência de máquinas com a capacidade de pensar, as quais poderiam competir com os seres humanos em atividades puramente intelectuais. Segundo Carrion e Quaresma (2019) a melhor forma de tornar isso possível seria fornecer aos equipamentos os melhores órgãos sensoriais, ensinando posteriormente a utilizá-los para pensar e agir, podendo inclusive entender e falar uma língua. A inteligência artificial é observada hoje atuando em espaços inteligentes, utilizando de interfaces para se conectarem e interagirem com usuários e outros equipamentos.

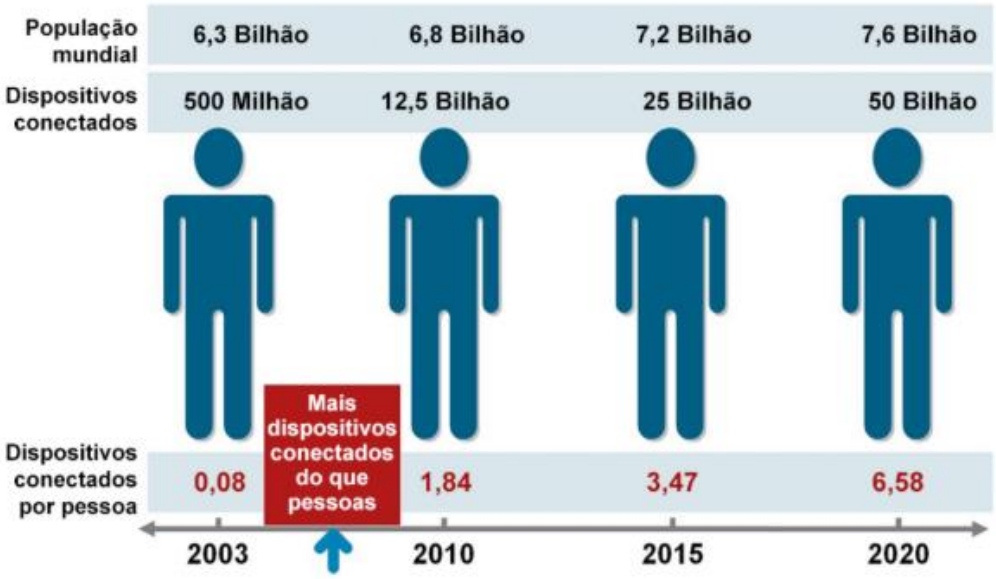
3.1 IOT: MUNDO DE COISAS CONECTADAS

Para entender o que significa Internet das Coisas é necessária a definição destes termos. Seguindo a possibilidade levantada por Turing, as “Coisas” seriam objetos interconectados, podendo tomar ações com base em um protocolo de comunicação, o que pode ser definida como “Internet” (CARRION; QUARESMA, 2019). De forma mais abstrata a IoT pode ser definida como uma rede onde os objetos se comunicam, pensam e tomam ações de forma autônoma, sem necessidade de intervenções do usuário para que elas aconteçam.

De acordo com o Cisco IBSG, a IoT também pode ser definida como o momento exato em que o número de “coisas” conectadas à internet superou o número de pessoas. Após a explosão na utilização de dispositivos celulares e smartphones o número de dispositivos conectados aumentou de forma exponencial. Enquanto o número de pessoas aumentou de 6,3 bilhões para 6,8 bilhões entre 2003 e 2010, a quantidade de equipamentos saltou de 500 milhões para 12,5 bilhões (EVANS, 2011). De acordo com estes dados o Cisco IBSG estima que a IoT tenha “nascido” entre 2008 e 2009. Conforme a figura 1, criada pelo Cisco IBSG há a expectativa de que em 2020 existam cerca de 7,6 bilhões de pessoas, enquanto o número de equipamentos deve chegar a incríveis 50 bilhões, formando uma densidade de 6,58 dispositivos conectados por habitante. Vale salientar que o número de dispositivos apresentado é uma média geral, uma vez

que enquanto algumas pessoas possuem vários dispositivos outras possuem poucos ou até nenhum dispositivo conectado.

Figura 1 - Relação entre dispositivos conectados e população ao longo dos anos



Fonte: Evans (2011)

Para Evans (2011) esta nova rede mudará tudo, inclusive já vem impactando de forma cada vez mais evidente em áreas como educação, comunicação, negócios, ciência e tantas outras. Ainda segundo o autor a IoT proporcionará a evolução da internet, dando um salto enorme na possibilidade de coletar, analisar e distribuir dados que são posteriormente transformados em informações, conhecimento e sabedoria. A internet quando foi projetada, na era da ARPANET, possuía diversos protocolos de comunicação como AppleTalk, Token Ring e IP. Existente desde a época o protocolo padrão hoje é o IP.

A principal e mais importante contribuição da internet das coisas hoje é a possibilidade de sentir, coletar, transmitir, analisar e distribuir dados em grande escala, permitindo que estes sejam processados pelas pessoas a fim de gerar informações e conhecimentos, chegando à sabedoria. Ao unir esta possibilidade com a inteligência artificial tem-se um aumento significativo do ganho, proporcionando uma forma de prosperar mais no futuro. No próximo tópico serão abordados exemplos de usos da IoT nos dias atuais.

3.2 APLICAÇÕES DA INTERNET DAS COISAS

Com o grande número de dispositivos conectados e com capacidade de coletar, processar e transmitir dados são abertos novos horizontes nas áreas de automação e comunicação entre máquinas. A IoT hoje pode ser aplicada a praticamente qualquer área da sociedade, trazendo possibilidades quase infinitas para mudar a vida das pessoas (EVANS, 2011).

Evans (2011) traz alguns exemplos de aplicações nos dias de hoje. Um destes, intimamente ligado a este trabalho, é a implantação de sensores nas orelhas do gado, a fim de permitir que fazendeiros monitorem a saúde e localização das vacas, garantindo assim um suprimento maior e mais saudável de carne para o consumo. Este cenário implantado pela empresa holandesa Sparked gera cerca de 200 megabytes de informações por animal ao ano.

O autor ainda traz um exemplo com o objetivo de diminuir diferenças entre ricos e pobres com o monitoramento da distribuição de serviços essenciais. Neste cenário têm-se duas regiões da cidade de Warden Road (Índia), onde na parte mais rica a água custa $0,03, enquanto na parte mais pobre o metro cúbico custa $1,12. Esta diferença ocorre tanto em função das ineficiências de infraestrutura na parte com menos recursos, tanto pelos furtos realizados por parte da população carente. Com o auxílio da IoT é possível disponibilizar às autoridades informações para identificar e corrigir estes problemas, permitindo preços mais baixos aos bairros mais pobres, o que também iria encorajar os que utilizam os serviços de graça a se tornarem clientes contribuintes.

Já Silva (2019) propõe uma solução de baixo custo para monitoramento de dados climáticos. Estes, quando obtidos da internet geralmente tem como base uma estação meteorológica existente da cidade, porém as grandes dimensões e diferenças de relevo e vegetação de alguns municípios podem gerar grandes distorções entre pontos da região. Com esta solução menos custosa é possível que cada propriedade possua sua própria estação, gerando dados mais precisos, especialmente sobre temperatura e pluviometria, possibilitando assim mais efetividade em decisões na agricultura.

A indústria vem sendo muito beneficiada pelo uso de IoT. Como exemplo tem-se a fabricação de veículos, que vem sofrendo fortes e positivos impactos pela “Indústria 4.0”, denominada como uma nova revolução industrial. A conexão entre os equipamentos, aliada a sistemas autônomos para agilizar as decisões e alterar o gerenciamento do processo de modo automático, gera muito mais

rapidez na construção de novos carros, minimizando a possibilidade de erros e trazendo mais lucratividade para a montadora (HABEKOST, 2019).

No próximo capítulo serão abordadas as redes sem fio mais utilizadas na Internet das Coisas, sendo este um ponto crucial para o bom funcionamento.

4. REDES DE COMPUTADORES SEM FIO

As redes de computadores vêm passando por um longo processo de evolução desde 1969, quando surgiu a primeira conexão entre máquinas (ARPANET), permitindo a transferência de 50 kbps (kilobits per second). Neste momento a rede possuía apenas poucos nodos, interligando quatro universidades dos Estados Unidos, porém com um rápido crescimento em 1973 já havia mais de 30 instituições conectadas (MORIMOTO, 2011).

Em 1974 passou a ser utilizado o TCP/IP como protocolo padrão para comunicação na ARPANET, sendo também utilizado na Internet até os dias atuais. Desde então houve diversas mudanças tanto do ponto de vista físico quanto lógico. Como melhorias físicas pode ser citado o aumento da capacidade de processamento dos emissores e receptores, a utilização de tecnologias como cabos coaxiais, cabos de par trançado e cabos de fibra óptica, além de roteadores e repetidores cada vez mais potentes. Já no ponto de vista lógico novos padrões e ferramentas vêm proporcionando cada vez mais segurança e consistência nos dados transmitidos.

4.1 REDES WIRELESS

Segundo Morimoto (2011), as redes sem fio permitem uma maior mobilidade, trazendo flexibilidade para usuários de dispositivos móveis e computadores portáteis. Como exemplo pode se utilizar um telefone celular, com o qual é possível andar entre os cômodos de uma casa sem a necessidade de desconectar o equipamento da rede Wireless. Além da facilidade de locomoção, as redes sem fios têm tomado grande espaço no mercado pois muitas vezes são construídas com um custo muito reduzido quando considerada a uma rede cabeada. A compra de centenas de metros de cabos, além de *switches* e roteadores, além da possível alteração necessária na infraestrutura do local e o custo de instalação são alguns fatores que tornam redes wireless menos custosas.

A ALOHAnet foi a primeira rede *wireless* funcional, entrando em operação a partir de 1970, antes mesmo da ARPANET. Seu surgimento se deu pela necessidade de comunicação entre as diversas ilhas do Havaí, pois até então o

único método de comunicação utilizado era através de mensagens escritas e enviadas em barcos entre as localidades.

Os primeiros transmissores *Wireless* suportavam poucos usuários simultâneos, porém os aparelhos modernos podem lidar tranquilamente com várias conexões. Ao considerar uma empresa este ponto é sensível, uma vez que tanto funcionários quanto clientes podem estar conectados na rede. Existem alguns padrões para estas redes, grande parte criada pela IEEE (instituto que trabalha em prol do avanço das tecnologias para bens da humanidade), tendo como exemplos o 802.11b, 802.11n e 802.11g (MORIMOTO, 2011).

O alcance das redes Wi-Fi pode variar drasticamente dependendo dos transmissores envolvidos, obstáculos e fontes de interferência presentes no ambiente. Segundo Morimoto (2011), as redes 802.11b e 802.11g tem alcance de 30 metros para ambientes fechados e 150 metros para ambientes abertos, enquanto a rede 802.11n chega a 70 e 250 metros, respectivamente. Exemplos de obstáculos são superfícies metálicas, tintas com pigmentos metálicos, espelhos, paredes de concreto e madeira, além de corpos de água, como caixas d'água, piscinas e o próprio corpo humano.

Os roteadores permitem configurações para que seja utilizado um canal específico para a transmissão de dados. Caso existam muitas redes em uma mesma frequência pode haver interferências entre uma e outra, porém com essas divisões é possível existir redes distintas no mesmo local, sem problemas de comunicação. O quadro 1 demonstra os principais canais utilizados. Existem 13 canais diferentes, porém grande parte dos dispositivos trabalha com apenas 11 em função dos Estados Unidos, principal mercado consumidor, utilizar apenas estes. O Brasil e boa parte dos países europeus permitem o uso dos canais 12 e 13, com frequência nominal de 2.467 GHz e 2.472 GHz, respectivamente.

Quadro 1 - Abrangência dos canais de rede Wireless

(continua)

Canal	Frequência nominal	Frequência prática
1	**2.412 GHz**	**2.401 a 2.423 GHz**
2	2.417 GHz	2.405 a 2.428 GHz
3	2.422 GHz	2.411 a 2.433 GHz

Quadro 1 - Abrangência dos canais de rede Wireless

(conclusão)

Canal	Frequência nominal	Frequência prática
4	2.427 GHz	2.416 a 2.438 GHz
5	2.432 GHz	2.421 a 2.433 GHz
6	**2.437 GHz**	**2.426 a 2.448 GHz**
7	2.442 GHz	2.431 a 2.453 GHz
8	2.447 GHz	2.436 a 2.458 GHz
9	2.452 GHz	2.441 a 2.463 GHz
10	2.457 GHz	2.446 a 2.468 GHz
11	**2.462 GHz**	**2.451 a 2.473 GHz**

Fonte: Morimoto (2011)

É possível usar até 3 canais para redes no mesmo local sem que haja interferências significativas entre as mesmas. Considerando o quadro 1, é possível a utilização conjunta dos canais 1, 6 e 11 em conjunto, sem que estes se sobreponham. No caso de utilização de outros canais em um mesmo espaço pode ocasionar quedas no desempenho das redes, porém este geralmente não é percebido pelo usuário.

Nos últimos anos ocorreram diversas melhorias na velocidade das redes *Wireless*. Exemplificando, a rede 802.11g suporta uma velocidade de 54 *megabits*, enquanto a 802.11n utiliza diversos artifícios para aumento de desempenho, como a ampliação no número de pontos de acesso do roteador, diminuição do intervalo entre troca de informações, utilização de mais de um canal de forma simultânea para envio de informações, entre outros (MORIMOTO, 2011).

Devido a estes avanços nas redes é possível um alcance cada vez maior para conexão entre o ponto de acesso e o dispositivo móvel, o que traz uma mobilidade muito grande para os usuários. Por outro lado pessoas indesejadas podem se conectar a rede, tendo acesso assim, além da web, a arquivos que estejam compartilhados na mesma. Desta forma o quesito segurança tem se tornado ponto chave na estruturação de uma rede *wireless* confiável, o que abre espaço para algumas soluções. A criptografia de dados é a forma mais utilizada

para garantir a integridade das informações, geralmente feita por dois protocolos existentes:

WEP é a abreviação de *"Wired-Equivalent Privacy"* e surgiu com o intuito de permitir a mesma segurança existente nas redes cabeadas, o que logo ficou em cheque. Os padrões WEP podem ter 64 bits (utilizado nos primeiros pontos de acesso 802.11b) e 128 bits (padrão atual), porém alguns fabricantes chegaram a criar chaves de 256 bits, estas podendo ser utilizadas apenas entre equipamentos do mesmo fabricante. A grande limitação é que este protocolo é baseado no uso de vetores de iniciação combinados, o que juntamente com outras vulnerabilidades, torna relativamente fácil a quebra da chave (MORIMOTO, 2011).

Em 2003 foi criado o WPA (*Wired Protected Access*), que veio junto com um pacote de melhorias chamado 802.11i, sendo resposta às múltiplas vulnerabilidades do WEP. Um dos grandes diferenciais entre as redes foi a extinção do uso de vetores de inicialização para configuração da chave de acesso, adotando o sistema TKIP (*Temporal Key Integrity Protocol*), no qual a chave é trocada periodicamente e a chave definida na configuração da rede é usada apenas para a conexão inicial.

Morimoto (2011) explana que no ano de 2004 o padrão 802.11i foi finalizado, trazendo uma nova melhoria ao WPA, o WPA2, o qual utiliza o AES como sistema de encriptação (mais seguro e pesado). Devido ao maior processamento necessário para utilização do AES alguns pontos de acesso não o suportam. Desta forma este é um ponto que deve ser considerado na construção do sistema proposto, a fim de garantir que todos os receptores tenham capacidade necessária para atender o padrão adotado.

Apesar da alta velocidade e consistência das redes, uma limitação para as redes *Wireless* é a distância, pois em áreas rurais a extensão necessária pode chegar a quilômetros. A seguir será abordada a rede LoRa como alternativa para este impedimento.

4.2 REDES LORAWAN

LoRa (*long rage*) é um termo criado pela LoRa Alliance, uma organização com mais de 500 companhias e suporte a desenvolvedores em diversos países do planeta. Seu objetivo é facilitar a implementação de soluções IoT em larga escala

através da utilização de um ecossistema robusto e confiável chamado de LoRaWAN. Ao redor do mundo há algumas padronizações para a utilização desta rede de acordo com o definido pelos órgãos reguladores. No Brasil, por exemplo, a regulamentação de comunicações sem fio é feita pela ANATEL (Agência Nacional de Telecomunicações) (RIBEIRO, 2019).

O alcance possível com a rede LoRa é realmente surpreendente. Pereira e Cruvinel (2019) afirmam que em campo aberto os sinais podem chegar a 15 km de alcance, o que se torna suficiente para pequenas propriedades com alguns hectares de área. Por outro lado a taxa de transmissão de dados é relativamente baixa quando comparada a outras redes, chegando até 50 kbps (BONOTTO, 2018).

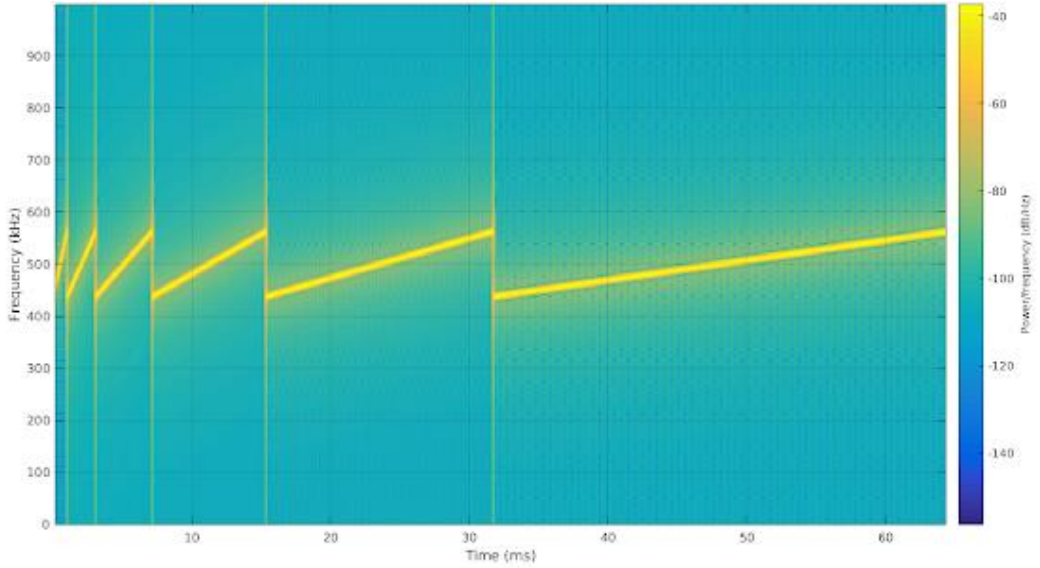
O sistema LoRa é baseado em um protocolo de comunicação MAC que pode ser colocado em plataformas de IoT para transmissão de dados, e também em *gateways*, onde os dados podem ser convertidos em conexão IP, atingindo assim inclusive servidores externos (PEREIRA, CRUVINEL, 2019).

LoRa utiliza a modulação CSS (*Chirp Spread Spectrum*), desenvolvida para aplicações de radar e utilizada em sistemas militares. Os sinais *Chirp* (*Compressed High Intensity Radar Pulse*) varrem toda a largura de banda e possuem amplitude constante, variando a frequência da transmissão de uma forma linear em um determinado espaço de tempo. Quando esta variação é feita da menor para a maior frequência trata-se de um sinal *up-chirp*, já quando a mudança é da maior para a menor denomina-se *down-chirp*. O deslocamento dos *Chirps* forma os *Chirps* de transporte de dados, o que por sua vez transporta a informação (MARQUES, BOCHIE, 2018).

Nestas redes existem diversos parâmetros (conforme listagem abaixo) que podem ser configurados, os quais impactam na taxa de transferência de bits, na resistência a ruídos e a facilidade de codificação.

- *Bandwidth* (BW): Define a largura de banda ocupada por um *Chirp*, que pode ser de 125 kHz, 250 kHz e 500 kHz.
- *Spreading Factor* (SF): Fator de espalhamento, o que influencia na duração de um *Chirp* no tempo. Quanto maior o valor deste parâmetro (que podem variar entre 7 e 12), maior será o tempo de transmissão no ar e menor a quantidade de dados transmitidos. Na figura 2 segue a comparação entre os fatores de espalhamento possíveis.

Figura 2 - Comparação de Fatores de espalhamento do Lora

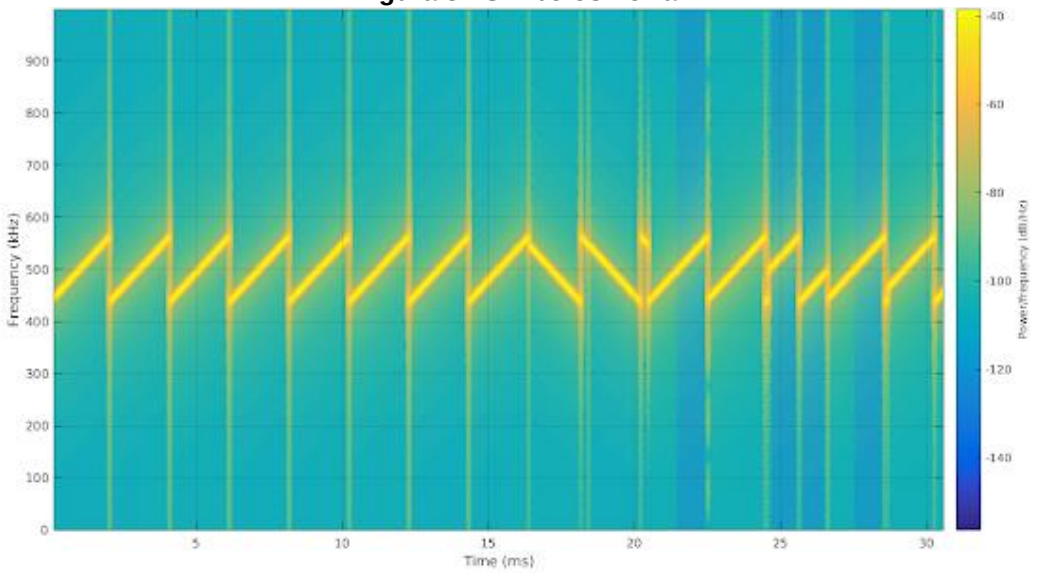


Fonte: Ghoslya (2017)

- *Code Rate* (CR): Taxa de código.

Na figura 3 tem-se um exemplo de transmissão de dados a partir de *Chirps*. Os primeiros 8 *up-Chirp* são utilizados para detecção do sinal LoRa; os dois *down-Chirp* seguintes representam a sincronização de tempo; os 5 *up-Chirps* seguintes refletem a transmissão de pacotes modulados, possuindo dados úteis.

Figura 3 - Símbolos LoRa

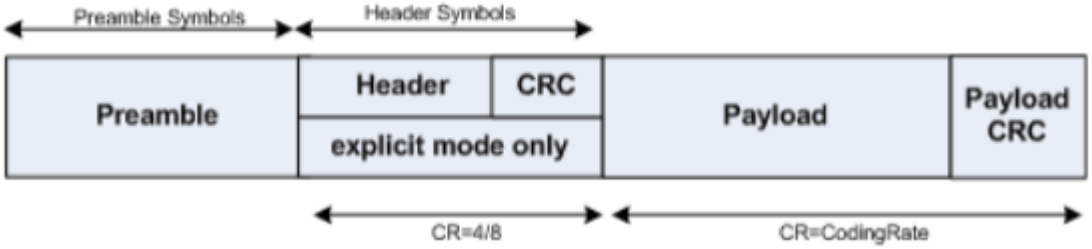


Fonte: Ghoslya (2017)

A estrutura de dados de um pacote Lora é observada na figura 4. O preâmbulo pode ter seu tamanho configurado e atua como identificador de início do pacote. O *Header* (cabeçalho) possui informações do pacote, como por

exemplo, o tamanho do *payload*. O *payload* por sua vez contém os dados úteis da comunicação, enquanto o CRC possibilita a conferência dos dados recebidos.

Figura 4 - Pacote Lora



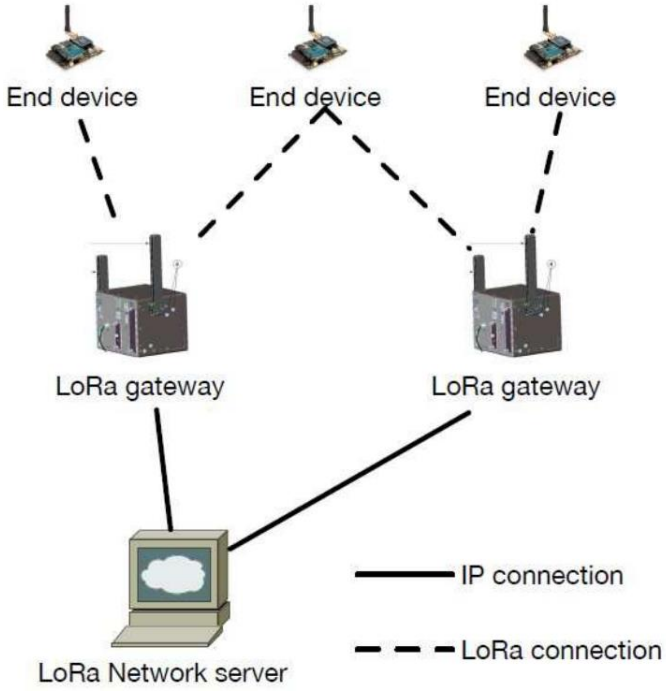
Fonte: Lavric; Popa *apud* Bonotto (2018)

Segundo Ribeiro (2019), a arquitetura de uma rede LoRa é definida com a utilização de 3 tipos de dispositivos, sendo:

- *End devices*: dispositivos finais que possuem o papel de gerar dados e enviar ao centralizador, assim como receber informações para executar ações. É importante salientar que estes equipamentos não podem conversar entre si nesta rede. Podem por sua vez ser classificados em A, B e C, sendo A apresentando duas janelas para recepção de *downlinks* após o envio de uma mensagem, B possibilitando agendamento de janelas para recepção de dados, e C apresentando maior tempo de recepção de dados (consumindo assim mais energia);
- *Gateways*: centralizadores que recebem dados LoRa e transmitem ao servidor central via conexão IP;
- *Network Servers*: servidores centrais, onde ficam armazenadas as aplicações que utilizarão os dados.

  Na figura 5 é possível verificar de forma visual esta estrutura.

Figura 5 - Estrutura da rede LoRa



Fonte: AUGUSTIN; et al (2016) *apud* Ribeiro (2019)

Ainda segundo Ribeiro (2019), a autenticação de dispositivos nessas redes pode ser feitas através de 4 métodos:

- *End Device Address*: Autenticação é feita tendo o endereço do dispositivo final cadastrado na rede;
- *Application identifier*: Identificação exclusiva da aplicação global, identificando o dono do *end device*;
- *Network Session key*: Chave exclusiva utilizada entre o dispositivo final e o servidor central, utilizada na verificação de integridade das mensagens;
- *Application Session Key*: Semelhante ao método anterior, porém utilizando encriptação do *payload* transmitido.

Conforme visto anteriormente, a rede Wireless é uma ótima opção a ser utilizada na conexão entre os *gateways* e o servidor central, porém há a possibilidade de queda nesta transmissão de dados. Visando uma forma de minimizar os possíveis impactos, a seguir serão apresentadas as redes celulares, que podem ser utilizadas como contingência para a troca de informações.

4.3 REDES CELULARES

Hoje em dia as redes celulares estão muito presentes no cotidiano da população. Através destas redes os usuários podem acessar a internet em seus celulares, por exemplo. Neste tópico se verá a evolução destas redes e como podem ser utilizadas neste projeto.

4.3.1 PRIMEIRA GERAÇÃO (1G)

As primeiras redes de celulares surgiram a partir de 1979 nos Estados Unidos, com o AMPS (*Advanced Mobile Phone System*). Este padrão utiliza a tecnologia FDMA (*Frequency Division Multiple Access*) que trabalha com transmissões analógicas (transmite apenas voz) e com canal dedicado. Desta forma a banda é dividida em radiofrequências, exigindo um canal de emissão e outro de recebimento de dados, sempre na faixa de 800 MHz (MORIMOTO, 2011).

4.3.2 SEGUNDA GERAÇÃO (2G)

Segundo Morimoto (2011), em função do aumento das redes analógicas e a chegada do limite de sua capacidade, houve a necessidade da criação de sistemas digitais, alavancada também por outros benefícios como a codificação digital de voz mais poderosa, melhora na qualidade de voz, criptografia, maior eficiência espectral e maior facilidade na comunicação de dados. No Brasil esta geração foi implantada através das tecnologias TDMA, CDMA e GSM. Quando implantada, o custo para transferência de dados era realmente alto, chegando a R$ 15,00 por *megabyte* em planos pré-pagos de telefonia.

O TDMA (*Time Division Multiple Access*) na frequência de 800 MHz, e a divisão dos canais é feita em até 6 intervalos de tempo, chegando a uma velocidade de 30 KHz. Devido à mudança das operadoras para outras tecnologias, esta tecnologia deve ser extinta em breve.

Já o CDMA (*Code Division Multiple Access*) defende a técnica de espalhamento espectral, utilizando toda a largura de banda disponível para um determinado canal (1,23 MHz). Sua vantagem é a possibilidade de manter diversos assinantes conectados à rede de forma simultânea, identificando cada transação com um código único. Esta rede trabalha na mesma faixa da tecnologia GSM (vista à frente), entre 800 e 1900 MHz (DUSSAUX *et al*., 2010).

O GSM (*Groupe Spécial Mobile*, mais tarde renomeado para *Global System for Mobile*) foi desenvolvido na Europa, entrando em operação na Finlândia em 1991 e se propagando rapidamente para diversas regiões no mundo, se expandindo até hoje por diversos países. Seu diferencial é o uso de cartões de memória SIM (*Subscriber Identity Module*) nos aparelhos, o que permite a identificação do usuário mesmo em outro aparelho ou outra rede GSM. Sua operação é feita com uma combinação das técnicas FDMA e TDMA e opera nas faixas de 850, 900, 1800 e 1900 MHz. Com o GSM surgiram diversos serviços, como o envio de mensagens de texto, transmissão de pacotes de dados, configuração remota do aparelho, localização do equipamento celular, teleconferência, entre outros (MORIMOTO, 2011).

4.3.3 ENTRE GERAÇÕES (2,5G)

A principal diferença para esta e as gerações anteriores é a modulação, permitindo a comutação por pacotes (assim como as redes IP e arquitetura TCP/IP) ao invés de circuitos (DUSSAUX *et al*., 2010).

O GPRS é fornecido pelas operações, provendo transmissão de pacotes, operando a uma velocidade média de 30 a 40 kbps, podendo chegar a 115 kbps. A vantagem dos dados serem enviados por pacotes é a possibilidade de permanecer sempre conectado à rede, porém com a cobrança sendo feita sempre em relação aos dados transmitidos. Esta tecnologia hoje possui ampla cobertura móvel, permitindo vários usuários em um mesmo canal devido à comutação de pacotes, porém com uma latência de conexão entre 500 e 1000 ms (MORIMOTO, 2011).

Como melhoria do GPRS surgiu o EDGE (*Enhanced Data Rates for GSM Evolution*), utilizando esquemas de correção de erros e modulação diferentes, chegando a um limite teórico de 473,6 kbps, além de ser uma conexão mais robusta e confiável.

4.3.4 TERCEIRA GERAÇÃO (3G)

Os sistemas 3G permitem serviços de comunicação de dados e telefonia maiores que seus antecessores, podendo chegar a 2 Mbps de velocidade.

Uma das tecnologias é a UMTS (*Universal Mobile Telecommunications Service*), tendo base em pacotes IP e velocidade média de 220 a 320 kbps em movimento. Seu desenvolvimento teve como objetivo prover infraestrutura necessária para maiores consumos de banda, como videoconferência e *streaming*, tanto em celulares como *laptops*. Por ser compatível com a EDGE e GPRS é possível sair de uma rede UMTS e conectar nestas outras redes de forma automática (MORIMOTO, 2011).

Já o CDMA 1xEV-DO (*Evolution, Data-Optimized*) é uma evolução do CDMA, provendo alta desempenho e picos de dados de até 2,4 Mbps, porém são necessárias portadoras distintas para dados e voz.

O HSDPA (*High Speed Downlink Packet Access*) / HSUPA (*High Speed Uplink Packet Access*), respectivamente velocidade de *downlink* e *uplink*, que permite o envio e recebimento de grandes arquivos, acesso remoto e transferência de vídeos e imagens em alta resolução.

Como citado anteriormente, a utilização destas tecnologias no projeto visa manter a conexão de dados entre *gateway* e servidor central no caso de instabilidade da rede *Wireless*, apesar da menor velocidade de transmissão de dados. Na próxima seção serão abordadas plataformas e *chips* para desenvolvimento de aplicações em IoT (MORIMOTO; 2011), as quais farão uso das redes descritas.

5. PLATAFORMAS DE IOT

Para o desenvolvimento do projeto foram analisadas algumas plataformas de *hardware* para aplicações em IoT. Nesta seção serão exploradas as duas que melhor se adaptaram com a necessidade.

5.1 RASPBERRY PI

O Raspberry Pi é um microcomputador de baixo custo. Pesando cerca de 45 g, possui o tamanho de um cartão de crédito e um *hardware* com 1 GB de RAM, processador de 700 MHz e conectores HDMI, USB, áudio, Ethernet, entre outros, podendo haver configurações superiores dependendo da versão escolhida. Desta forma é possível utilizar o equipamento para diversas funções, seja como

controlador de robôs, atividades de sensoriamento, e até mesmo como um *Desktop*. Além da parte física, é possível programar as aplicações em Python, C++, Java ou até mesmo Assembly (ANDRADE, SOMA, NAKAMURA, 2016).

As especificações do Raspberry Pi 4 seguem no quadro 2:

Quadro 2 - Especificações Raspberry Pi 4

Especificações do Raspberry Pi 4
Processador
1GB, 2GB ou 4GB RAM (depende o modelo)
Porta Ethernet Gigabit
BCM54538 wireless LAN e Bluetooth Low Energy (BLE) a bordo
GPIO de 40 pinos
4 portas USB (2 x 2.0, 2 x 3.0)
2 portas Micro HDMI com suporte a resolução 4K
Conexão de Câmera
Saída para áudio 3,5 mm
GPIO com 40 pinos
Entrada para cartão SD com suporte a 50 MB/s
Temperatura de operação entre 0ºC e 50°C

Fonte: Raspberry (2019)

Na figura 6 é possível visualizar a placa do Raspberry Pi 4, com diversas conexões.

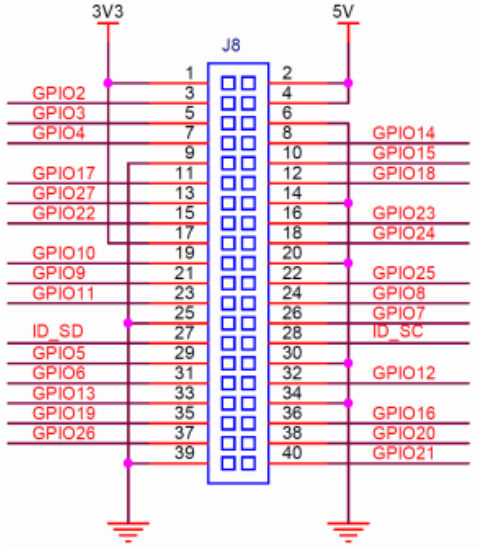
Figura 6 - Raspberry Pi 4

Fonte: Robun.in (2019)

Como benefícios pode se citar a acessibilidade proporcionada pelo baixo custo, o baixo consumo de energia quando comparado a outros computadores, a facilidade de utilização (inclusive com gráficos para exibição de imagens e vídeos), junção de todos os componentes em uma única placa e a conectividade, podendo ser feita via Wi-Fi, *bluetooth*, entre outros (ANDRADE, SOMA, NAKAMURA, 2016).

Dos 40 pinos disponíveis, 12 possuem funções específicas, ficando 28 disponíveis para funcionalidades de entrada e saída (conforme figura 7).

Figura 7 - Pinagem GPIO Raspberry Pi 4

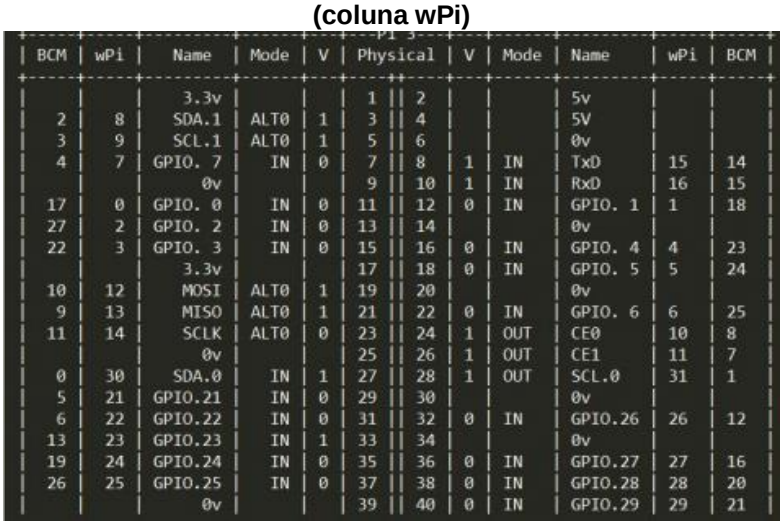


Fonte: Raspberry (2019)

Para acesso aos pinos de GPIO é utilizada a biblioteca Wiring Pi, esta desenvolvida para as linguagens C/C++ e Python, disponibilizando as seguintes funções :

- *wiringPiSetup(void)*: inicializa o wiringPi. Vale ressaltar que a enumeração da biblioteca é diferente da oficial do Raspberry, o que pode ser observado na figura 8.

Figura 8 - Mapeamento entre pinos do Raspberry Pi (coluna BCM) e da biblioteca WiringPi (coluna wPi)



BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM
		3.3v			1 || 2			5v		
2	8	SDA.1	ALT0	1	3 || 4			5V		
3	9	SCL.1	ALT0	1	5 || 6			0v		
4	7	GPIO. 7	IN	0	7 || 8	1	IN	TxD	15	14
		0v			9 || 10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11 || 12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13 || 14			0v		
22	3	GPIO. 3	IN	0	15 || 16	0	IN	GPIO. 4	4	23
		3.3v			17 || 18	0	IN	GPIO. 5	5	24
10	12	MOSI	ALT0	1	19 || 20			0v		
9	13	MISO	ALT0	1	21 || 22	0	IN	GPIO. 6	6	25
11	14	SCLK	ALT0	0	23 || 24	1	OUT	CE0	10	8
		0v			25 || 26	1	OUT	CE1	11	7
0	30	SDA.0	IN	1	27 || 28	1	OUT	SCL.0	31	1
5	21	GPIO.21	IN	0	29 || 30			0v		
6	22	GPIO.22	IN	0	31 || 32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	1	33 || 34			0v		
19	24	GPIO.24	IN	0	35 || 36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37 || 38	0	IN	GPIO.28	28	20
		0v			39 || 40	0	IN	GPIO.29	29	21

Fonte: Pereira; Yamada (2018)

- *pinMode (int pin, int mode)*: define o pino *pin* para um dos modos possíveis: INPUT, OUTPUT, PWM_OUTPUT ou GPIO_CLOCK.
- *digitalWrite (int pin, int value)* e *digitalRead (int pin)*: permite respectivamente escrita e leitura dos pinos GPIO.

5.2 ESP32

O ESP32 é um microcontrolador projetado pela *Espressif Systems*, lançado e apresentado no mercado em 2016, sendo considerado desde então um dos controladores mais robustos devido à sua velocidade de processamento, acessibilidade e conectividade. Seu processador pode possuir um ou dois núcleos de 32 bits que chegam a trabalhar com um *clock* de até 240 MHz e uma memória RAM de 520 kB.

O quadro 3 apresenta algumas especificações técnicas do ESP32.

Quadro 3 - Especificação ESP32

Especificações ESP32
Processador *single* ou *dual-core* de 32 bit
520 KB RAM
34 GPIOs programáveis
Temperatura de operação entre -40ºC e 125ºC

Fonte: Espressif Systems (2019)

Na figura 9 é exibido um módulo ESP32 com um pequeno visor incorporado. Há diversos modelos no mercado, porém este modelo será o utilizado visando facilitar o monitoramento e visualização das informações sensoriadas.

Figura 9 - ESP 32

Fonte: Amazon (2019)

Com um baixo consumo de energia permite a conexão com diversas redes, entre elas a *bluetooth*, *Wi-Fi* e a LoRaWAN, sendo as duas últimas já apresentadas neste trabalho. Devido à robustez do ESP32 e ao alto número de componentes que podem ser incorporados, vem sendo muito utilizado em projetos de IoT em diversas áreas, especialmente na automação residencial e industrial.

Os pontos abordados são essenciais para que este modelo seja utilizado no projeto, em especial a possibilidade de conexão via LoRaWAN, aumentando muito o alcance do sinal para uso na área agrícola. Na próxima seção será abordado o MQTT como protocolo de comunicação para estes dispositivos.

6. MQTT

O MQTT (*Message Queue Telemetry Transport*) é um protocolo de comunicação desenvolvido inicialmente pela IBM no final dos anos 90 para vincular sensores de plataformas de petróleo a satélites e tornou-se padrão para comunicações IoT, possuindo suporte à comunicação assíncrona entre as partes. Sendo um protocolo assíncrono é escalável em ambientes com rede não confiável, uma vez que desacopla o emissor e o receptor no espaço tempo.

Por ser leve, permite a implementação em *hardware* com pouco poder de processamento e com redes de baixo desempenho, com banda limitada e alta latência. Outro ponto benéfico é a possibilidade de integração com outros serviços e aplicações em IoT, inclusive opções comerciais utilizadas hoje em dia. A

transmissão de dados é feita através de JSON, XML, entre outros formatos (YUAN, 2017).

Um dos protocolos mais conhecidos e utilizados na Internet hoje é o HTTP, porém há alguns pontos em que o MQTT se destaca, como:

- por ser assíncrona, o MQTT permite maior escalabilidade uma vez que os sensores podem enviar leituras e permitir que a rede controle a sincronização ideal entre dispositivos e serviços. O grande número de equipamentos envolvidos e a alta latência das redes trazem perdas de desempenho em outros protocolos;
- sensores e dispositivos de IoT são clientes, porém no protocolo HTTP é necessário que este inicie uma conexão com o servidor central, sem possibilidade de receber dados passivamente;
- MQTT permite o envio de comandos para todos os clientes de uma só vez, caso comum em aplicativos de IoT;
- MQTT é um protocolo mais leve que outros conhecidos na *Web*.

O protocolo MQTT é constituído por um *message broker* e diversos clientes. O *broker* é um servidor que recebe as mensagens dos clientes e roteia para os clientes necessários, estes podendo ser outros dispositivos ou uma aplicação que processará os dados (SANTOS; JUNIOR, 2019).

Por se tratar de um modelo de publicação e assinatura, os clientes podem assinar “tópicos” de mensagens do *broker*, sendo que esta conexão pode ser feita tanto por TCP/IP quanto por conexão TLS criptografada. Quando um cliente publica uma mensagem em determinado tópico, todos os “assinantes” recebem os dados, sendo que esta mensagem deve possuir até dois *bytes*. Yuan (2017) exemplifica um cenário onde os sensores publicam leituras no tópico “sensor_data” e assinam o “config_change”, enquanto a aplicação assina “sensor_data” e publica no “config_change”. Quando o servidor central publicar uma mensagem todos os clientes receberão, enquanto quando um cliente publicar uma leitura apenas o servidor central receberá os dados (YUAN, 2017).

6.1 MOSQUITTO

Para desenvolvimento com MQTT, o módulo Python Mosquitto é bastante utilizado. Este faz parte do projeto Eclipse Paho (projeto da Eclipse que fornece

implementações de software livres para IoT) e fornece SDKs e bibliotecas que podem ser utilizadas em diversas linguagens de programação. No quadro 4 são exibidos alguns comandos comuns do Mosquitto.

Quadro 4 - Exemplos de Comando MQTT

Comando	Função
$ mosquittto -d	Inicialização do *broker* do MQTT
$ mosquitto_sub -t “dw/demo”	Conecta o cliente ao *broker*
$ mosquitto_pub -t “dw/demo” -m “hello world!”	Conecta o cliente ao *broker* e envia uma mensagem

Fonte: Yuan (2017)

O Mosquitto possui alguns comandos para troca de mensagens. Abaixo seguem alguns exemplos de acordo com Yuan (2017).

- Comando CONNECT solicita conexão entre cliente e *broker*, possuindo os seguintes parâmetros:
    - *cleanSession*: define se a conexão é persistente ou não, armazenando todas as mensagens perdidas;
    - *username:* usuário para conexão ao *broker*;
    - *password*: senha para conexão ao *broker*;
    - *lastWillTopic*: tópico para publicação de mensagem no caso de uma queda inesperada na conexão;
    - *lastWillQos*: O QoS para o tópico anterior;
    - *lastWillMessage*: mensagem para o caso de uma queda inesperada;
    - *keepAlive*: intervalo mínimo de tempo entre troca de mensagens entre servidor para a conexão ser mantida como ativa.
- O comando CONNACK do *broker* para o cliente é enviado como confirmação da conexão;
    - *sessionPresent*: Indica se a conexão é persistente;
    - *returnCode*: “0” indica sucesso. Caso contrário é retornado o valor da falha.
- Comando SUBSCRIBE realiza a inscrição em um tópico:
    - *qos*: indica qual a consistência de dados para entrega das mensagens aos clientes (0 para não confiável, caso o cliente esteja indisponível acaba por não receber a mensagem; 1 para que a

mensagem seja entregue pelo menos uma vez; 2 para que a mensagem seja entregue exatamente uma vez);
    - *topic*: tópico para o qual deseja realizar a inscrição.
- Ao realizar a inscrição do cliente, *broker* retorna comando SUBACK:
    - *returnCode*: Valores 0, 1 e 2 identificam o *QoS* da inscrição; Valor 128 indica falha no processo;
- Comando UNSUBSCRIBE realiza a exclusão da assinatura:
    - *topic*: identifica o tópico ao qual quer deixar de assinar;
- Comando PUBLISH publica uma mensagem que será enviada para todos os inscritos:
    - *topicName*: identifica o tópico da publicação;
    - *QoS*: identifica o nível de qualidade do serviço de entrega;
    - *retainFlag*: indica se o *broker* deve manter a mensagem enviada como última do tópico;
    - *usefulLoad*: conteúdo da mensagem a ser enviada.

Devido à facilidade de uso e à leveza proporcionada por este protocolo, o MQTT mostrou-se um excelente padrão para troca de dados entre os dispositivos clientes do projeto e o servidor central. Outro ponto positivo é o fato de se tratar de uma tecnologia de software livre, sem necessidade de pagamentos para utilização da plataforma.

A fim de concluir o fluxo de dados desde as pontas até o servidor central, no próximo capítulo serão abordados sensores e atuadores, elementos fundamentais para a Internet das Coisas.

7. SENSORES E ATUADORES

O corpo humano capta sinais externos do ambiente através dos sentidos (tato, visão, olfato, paladar e audição), proporcionados por uma rede de inúmeros sensores que captam informações, como luminosidade e pressão, e encaminham através de uma rede de neurônios para o cérebro. Esta analogia é muito semelhante ao funcionamento de uma rede de IoT, onde o cérebro é representado pelo servidor central, a rede de neurônios são as redes de transmissão de dados (*Wireless*, LoRaWAN, entre outros) e os sensores de IoT são como os sensores do corpo. Conforme Wendling (2010), “sensores servem para informar um circuito eletrônico a respeito de um evento que ocorra externamente, sobre o qual ele deve atuar, ou a partir do qual ele deve comandar uma determinada ação”.

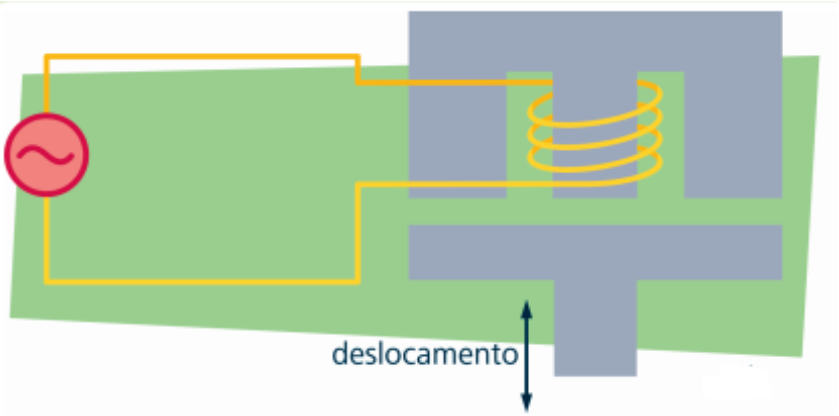
A partir da revolução industrial do século XIX passaram a surgir diversos instrumentos e técnicas de medidas para alavancar a produção industrial. Com o avanço da eletrônica e dos computadores no final do século XX o uso de sensores dos mais variados tipos passou a ser essencial para a maior automação e robotização dos sistemas (SEIDEL, 2011).

Na maioria dos casos os impulsos elétricos gerados pelos sensores não podem ser utilizados sem algum tratamento. Um exemplo de tratamento que pode ser necessário é em sensores onde a saída é gerada com uma tensão muito baixa, necessitando assim de um amplificador, permitindo assim a interpretação do sinal pela interface sistêmica (WENDLING, 2010).

Segundo Seidel (2011), os sensores podem ser classificados em:

- *Capacitivo*: constituído de duas placas metálicas paralelas com dielétrico, podendo ser empregado para detectar proximidade e captar níveis de líquidos, pressão, deslocamento e umidade;
- *Resistivo*: utilizado na medição de temperatura e medida de proximidade, utiliza resistências termométricas para realizar a captação de dados;
- *Magnético Indutivo*: transforma o movimento numa variação de indutância entre os elementos do sensor, esquema visível na figura 10.

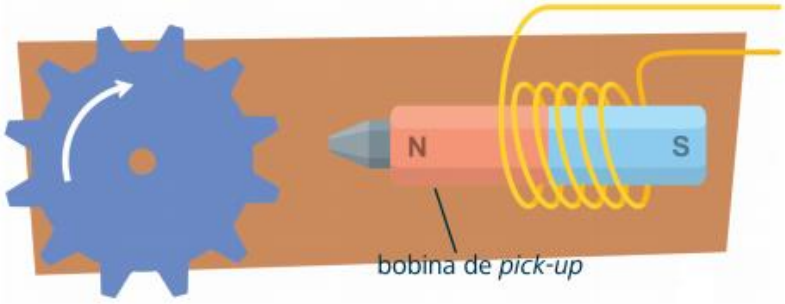
Figura 10 - Esquema de sensor magnético indutivo



Fonte: Seidel (2011)

- *Magnético de Relutância*: utiliza uma bobina enrolada em um núcleo ferromagnético. No exemplo da figura 11, conforme a placa magnética gira acaba por gerar impulsos elétricos que são utilizados para leitura da velocidade.

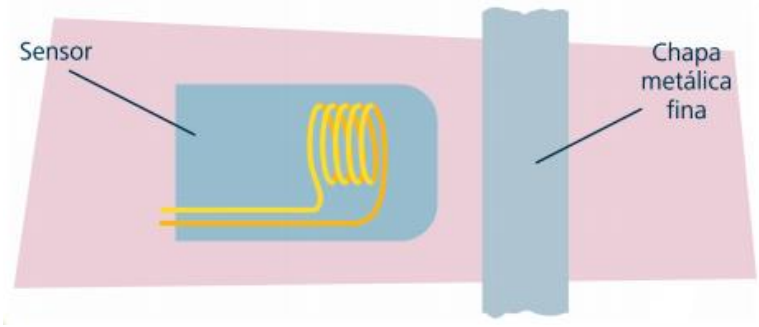
Figura 11 - Esquema de sensor magnético de relutância



Fonte: Seidel (2011)

- *Magnético de corrente parasita*: constituído de uma bobina que é excitada à alta frequência. A corrente gerada pelo movimento é reduzida quando próxima a alvos metálicos, sendo assim é possível medir distâncias pequenas entre estes. Esquema visível na figura 12.

Figura 12 - Esquema de sensor magnético de corrente parasita

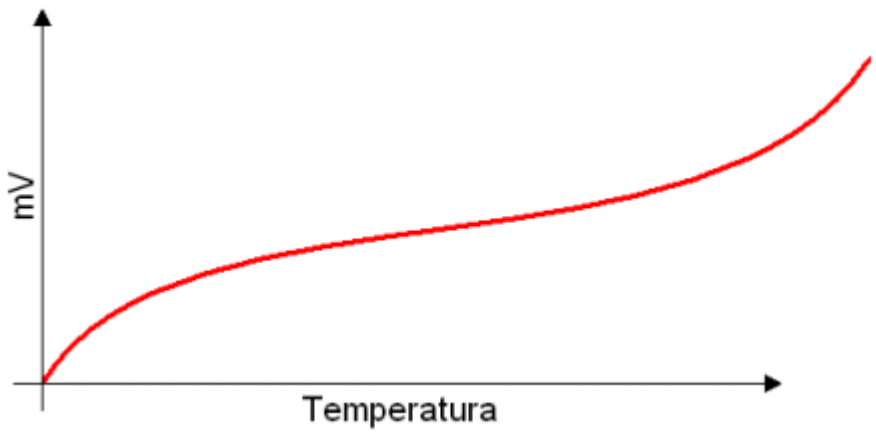


Fonte: Seidel (2011)

- *Efeito Hall*: utilizado para medir campos magnéticos, é constituído por um condutor que é percorrido por uma corrente perpendicular ao campo magnético.
- *Piezoelétrico*: frequentemente é utilizado em sensores ultrassônicos, de aceleração, força e pressão, produz uma tensão em seus terminais quando alguma força é aplicada.
- *Strain gauges*: constituído por um fio metálico em forma de *zig-zag* é montado em folha flexível. Quando aplicada uma tensão mecânica sobre o sensor, a forma do fio muda, permitindo detectar pequenos deslocamentos.
- *Óptico*: baseado na modulação de fontes de luz e em um detector de luz, permitindo medir nível de luminosidade do ambiente e proximidade.
- *Ultrassônico*: utilizado para medições de distâncias, níveis de líquidos e deslocamentos, o dispositivo emite ultrassons recebidos por outro dispositivo, onde o tempo de viagem ou mudança de fase entre transmissor e receptor é utilizado como fator de medida.

Um sensor analógico pode assumir qualquer valor como seu sinal de saída ao longo do tempo dentro de uma faixa de operação. A figura 13 evidencia em forma gráfica a medição de temperatura em um sensor analógico.

Figura 13 - Exemplo de medição por sensor analógico



Fonte: Wendling (2010)

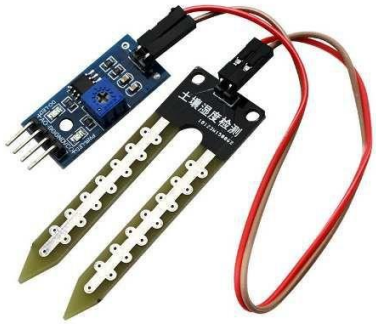
Sensores digitais assumem apenas dois valores como saída, podendo estes ser interpretados como zero e um, e são utilizados para detecção de passagem de objetos, medição de distância e velocidade, por exemplo.

Os atuadores são dispositivos utilizados nas pontas da hierarquia e funcionam de forma similar aos sensores, porém estes são utilizados para realizar ações, como abrir um portão ou acender uma lâmpada, por exemplo.

7.1 EXEMPLOS

Sensor de umidade de solo: podendo ser utilizado em diversos tipos de solo, é capaz de identificar se o mesmo está seco ou úmido. O dispositivo é visível na figura 14.

Figura 14 – Sensor de Umidade do Solo



Fonte: Medidor de umidade do solo (2019)

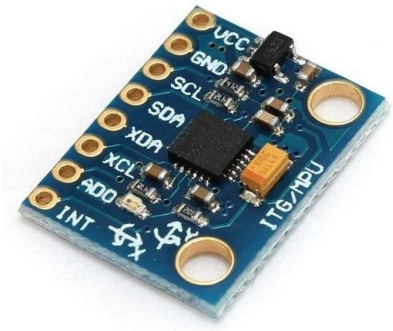
Sensor de presença: representado pela figura 15 gera impulsos elétricos sempre que há alguma alteração na distância entre o sensor e os objetos ao seu alcance.

Figura 15 – Sensor de Presença

Fonte: Sensor de presença (2019)

Acelerômetro: sensor capaz de identificar deslocamentos, sendo representado na figura 16.

Figura 16 - Acelerômetro



Fonte: Acelerômetro (2019)

Para a maior facilidade de uso do sistema de IoT é fundamental ter uma interface com boa usabilidade. Por este motivo na próxima seção serão abordados elementos chave referente à análise de sistemas.

8. ANÁLISE DE SISTEMAS

Conforme Sommerville (2007), os requisitos funcionais em um sistema descrevem quais funções ele deve realizar. Os mesmos dependem tanto do tipo de *software* a ser desenvolvido, assim como a abordagem utilizada pela organização e os usuários que utilizarão o *software*. Como exemplos desta categoria de requisitos pode-se citar “O sistema deve permitir realizar o cadastro de produtos” ou ainda “O sistema deve permitir realizar o cadastro de vendas”.

Há também os requisitos não funcionais, sendo que estes não são relacionados diretamente com as funções fornecidas pelo sistema. Podem estar vinculados às propriedades como confiabilidade, tempo de resposta ou espaço de armazenamento. Muitas vezes pode ser difícil a verificação quanto à eficácia e funcionalidade deste tipo de requisito. Como exemplo destes pode ser citada a afirmação “O sistema deve permitir 50.000 acessos simultâneos de forma transparente para os usuários”.

Já os requisitos de domínio refletem as necessidades quanto ao domínio da aplicação e incluem uma terminologia específica ou fazem referência a algum conceito de domínio. Estes podem ser relacionados quanto à velocidade de execução, confiabilidade, interoperabilidade com algum sistema externo e leis específicas que possam estar relacionadas ao domínio da aplicação.

A partir dos requisitos levantados durante a análise é possível criar diversos artefatos como protótipos de interface, diagramas e fluxos de trabalho e de testes. Visto estes artefatos, é imprescindível salientar a importância de descrever os

requisitos de maneira clara para evitar ambiguidades no entendimento dos mesmos, sendo que estas podem gerar problemas desde artefatos mal elaborados ou incorretos, impactando, se não identificados, em erros nas futuras funcionalidades do sistema (SOMMERVILLE, 2007).

A arquitetura do sistema especifica informações importantes quanto ao mesmo, como a organização e disposição em camadas dos componentes e demais elementos do mesmo. Um dos padrões mais utilizados é o *Model-View-Controller* (MVC), que prega a separação do código em três camadas:

- *View*: Responsável pela visualização gráfica dos elementos e *layout*;
- *Controller*: Responsável por fazer o meio de campo entre a *view* e o *model*;
- *Model*: Camada responsável pelo acesso ao banco de dados e desenvolvimento da lógica de negócio.

Classes de análise representam abstrações de classes ou subsistemas que devem aparecer no projeto sistêmico. Estas classes têm o papel de representar a solução em mais alto nível por meio da qual é possível descrever as funcionalidades do sistema a ser desenvolvido. Uma de suas características é o foco em requisitos funcionais para que os requisitos não funcionais sejam levantados na fase de design. (GONÇALVES; CORTÉS, 2015)

9. TRABALHOS CORRELATOS

Neste capítulo serão discutidos os trabalhos relacionados ao objetivo deste documento. Com o aumento da quantidade de dispositivos conectados, associado à ampla possibilidade de atuação do LoRa, há diversas iniciativas de utilização da mesma em vastas áreas da sociedade. A discussão a seguir é feita com base em três protótipos de aplicação da plataforma em ambientes diferentes, onde todos utilizam princípios vinculados a este trabalho.

9.1 TRABALHO 1: DESENVOLVIMENTO DE UM SISTEMA DE COLETA AUTOMÁTICA DE DADOS AGRÍCOLAS BASEADO EM REDE LORA E NO MICROPROCESSADOR ESP32

Pereira e Cruvinel (2019) propõem um protótipo de dispositivo com a função de coletar e transmitir dados climáticos à uma base central utilizando a rede LoRaWAN e *Wi-fi*, além de um ESP 32. Durante o experimento os circuitos foram montados em uma *protoboard* a fim de realizar testes iniciais. Durante um teste do sistema com 12 horas de funcionamento onde o equipamento se encontrava a 30 metros de distância do receptor, não houve grandes problemas de conexão, confirmando a eficácia da rede utilizada. Ao longo do trabalho foram realizados diversos ajustes no *firmware* instalado no ESP 32 a fim de criar tratamentos para problemas de transmissão entre emissor e receptor, como o reenvio de pacotes e verificação da conectividade das pontas, visando assim aumentar a confiabilidade do sistema.

Com o sucesso dos testes, Pereira e Cruvinel (2019) defendem uma placa definitiva para o circuito do sistema, além da acomodação dos dispositivos em caixas resistentes às intempéries, utilizando estes no campo para monitoramento das alterações meteorológicas e seus impactos na agricultura.

9.2 TRABALHO 2: SMART CITY: CONTROLE DE SEMÁFORO UTILIZANDO A TECNOLOGIA LORA

Bonotto (2018) propôs uma solução com o propósito de melhorar a fluidez do trânsito com semáforos inteligentes, permitindo que o usuário possa mudar o

comportamento do semáforo remotamente para controlar o fluxo de veículos nas vias relacionadas, o que pode ser importante para desviar o trânsito em caso de acidentes, por exemplo.

Em um primeiro momento foi criado um protótipo utilizando a rede LoraWAN e *Wi-fi*, além de um módulo ESP8266. A primeira etapa realizada foi a criação de um *firmware* para comunicação LoRa, envolvendo um equipamento central e um cliente, sendo este a representação do semáforo, além de outras placas utilizadas no projeto. Para visualização e controle da aplicação foi criada uma página HTML, permitindo visualização da localização dos semáforos sobreposta sobre um mapa, aumentando assim a usabilidade do sistema. Ao clicar sobre uma das *tags* que representam os dispositivos são exibidas as configurações possíveis, seja ela fechar e abrir o semáforo, ou ainda deixá-lo intermitente.

9.3 TRABALHO 3: UMA APLICAÇÃO DA TECNOLOGIA LORA EM UM AMBIENTE HOSPITALAR

Em outra área de atuação, Ribeiro (2019) propõe um protótipo de rede Lora para testar sua eficácia dentro do ambiente hospitalar, utilizando as mesmas redes de comunicação dos trabalhos anteriores, juntamente com um Raspberry Pi e uma placa LoRa Dragino V95. Um fato importante é os testes terem levado em consideração a quantidade de paredes entre o *gateway* e o cliente.

Ao longo do trabalho foram realizados diversos testes com distâncias e quantidade de obstáculos variados, chegando a mais de 70 metros de distância em alguns casos e a 8 paredes em outros. O próprio autor sugere diversas melhorias a serem feitas no projeto, o que pode aumentar a qualidade e amplitude do sinal de comunicação. Como resultado chegou-se a confirmação da eficiência da rede LoRa para transmissões no ambiente hospitalar, com comunicação e transferências de dados sendo feitas com sucesso.

10. MODELO PROPOSTO

Este capítulo tem por objetivo descrever o modelo desenvolvido, com sua estrutura física e lógica, com o fim de suprir as necessidades de sensoriamento e controle do sistema. Em um primeiro momento serão identificados os requisitos funcionais para o projeto, seguido da estrutura de dados desenvolvida e o *hardware* necessário para a aplicação e funcionamento do sistema como um todo.

Todos os códigos desenvolvidos no projeto estão disponíveis na plataforma *GitHub*, acessível através do endereço online https://github.com/leandrorasch/TCCFeevale. A hospedagem pública e remota tem o objetivo de facilitar trabalhos futuros e a possibilidade de obter o objeto atualizado conforme sejam feitos *commits* de novas versões.

10.1. REQUISITOS FUNCIONAIS

Em um primeiro momento foram levantados alguns requisitos funcionais que o sistema deveria atender, sendo eles:

- permitir ao usuário vincular seus dispositivos, possibilitando assim encontrar posteriormente de forma rápida todos os dados gerados por estes dispositivos;
- os dados devem trafegar em uma estrutura de tópicos, onde as informações são publicadas nos mesmos e as pontas leem tais dados;
- permitir a tomada de ações por parte do usuário, como ativar uma máquina, através do sistema;
- permitir o gerenciamento de alarmes, sendo que quando um tópico/sensor tiver determinado dado lido possa disparar uma ação.

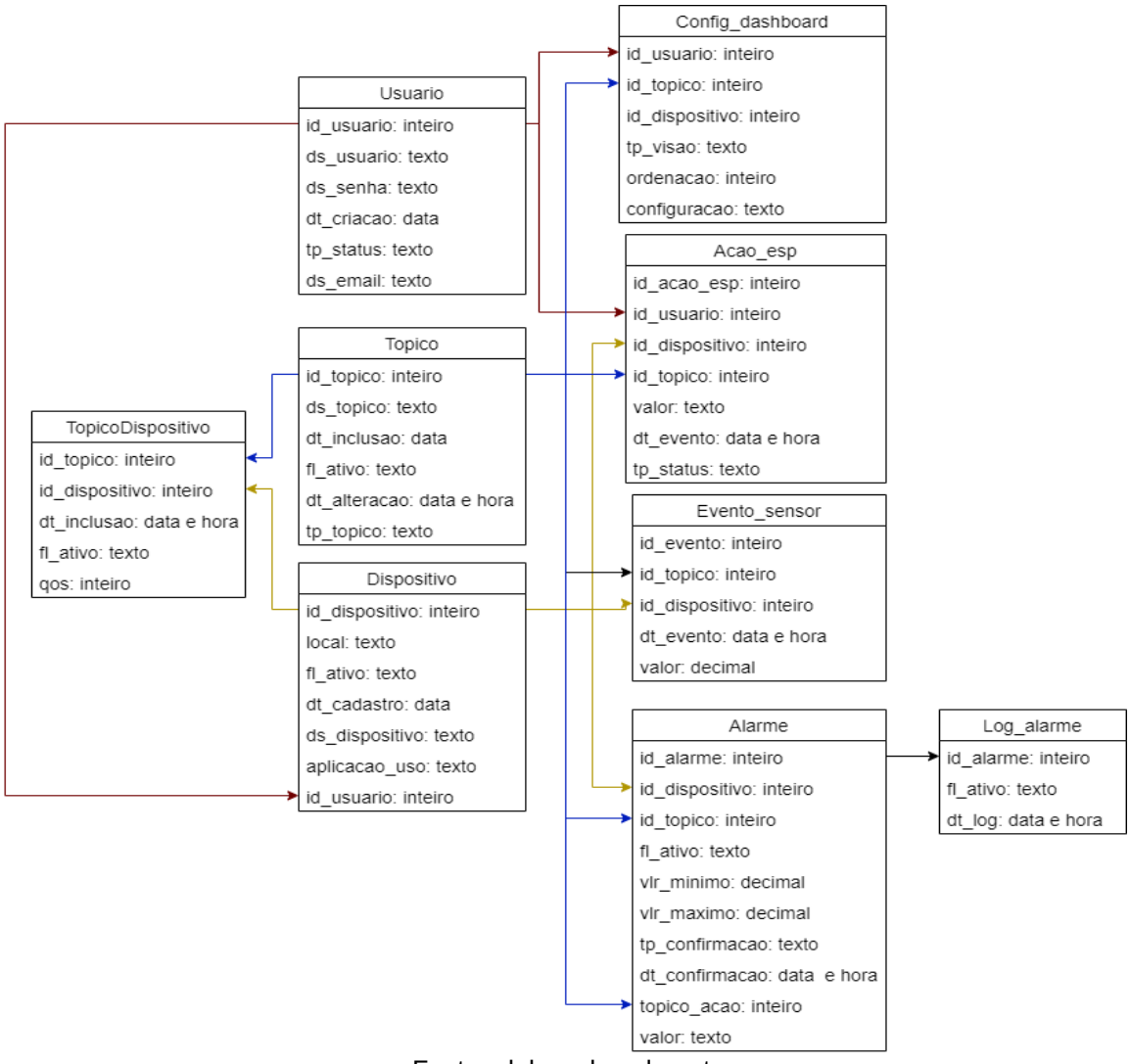
Atendendo sugestões de autoridades sanitárias de todos os níveis quanto à realização de um distanciamento social neste ano de 2020, devido à pandemia de COVID-19, decidiu-se não realizar um estudo aprofundado do perfil de possíveis usuários, utilizando então as necessidades e experiências do autor para definição dos processos e cenários de testes.

A estrutura de dados abordada a seguir foi desenvolvida e pensada levando em consideração os requisitos macro acima.

10.2. ESTRUTURA DE DADOS

Em um primeiro momento havia a expectativa de utilizar um *Raspberry* como centralizador das informações, porém ao longo do desenvolvimento optou-se por criar um *Web Service* e assim trazer mais robustez ao sistema em função deste estar conectado à uma base de dados e permitir hospedagem remota para atendimento a diversos clientes de forma simultânea. Para tanto se utilizou o banco de dados PostgreSQL, em função de possuir robustez e já haver familiaridade do autor com a ferramenta. Ao desenvolver o sistema foram levadas em consideração possibilidades de ampliação do sistema e a utilização de um único centralizador para diversos clientes. Com base nos requisitos descritos anteriormente criou-se o seguinte modelo:

Figura 17 – Modelo de dados do sistema



Fonte: elaborado pelo autor

Na figura 17 é possível perceber três estruturas principais, as quais são utilizadas como cadastro e base para as demais estruturas.

- Usuario: possui o cadastro do usuário, bem como as credenciais (senha e e-mail) para *login* no aplicativo;
- Topico: utilizada para cadastro dos tópicos em que as informações devem trafegar. Vale sinalizar que o campo *fl_ativo* identifica se o registro está ativo e disponível para visualização e utilização do usuário, enquanto *tp_topico* distingue quanto à sua funcionalidade, podendo ser para transmissão de sensores (valor S) ou para configurações e tomada de ações (valor C). Apenas tópicos do tipo C podem ser enviados do centralizador para os dispositivos da ponta.
- Dispositivo: com a função de manter o cadastro do dispositivo, possui campo específico para parametrização do usuário quanto à localização do *device* dentro da propriedade e sua aplicação de uso (respectivamente *local* e *aplicação_uso*). *Fl_ativo* identifica se o registro está ativo quanto ao recebimento e envio de mensagens, enquanto *id_usuario* identifica no sistema de qual usuário o dispositivo pertence.

Além das estruturas principais, existem algumas outras nas quais ocorrem parametrizações complementares e o armazenamento de todas as movimentações de dados provenientes do dispositivo LoRa e do aplicativo *Android*:

- TopicoDispositivo: utilizado para vínculo entre tópicos e dispositivos, possui uma chave para ativar o vínculo (*fl_ativo*) e um atributo que define o QoS que deve ser utilizado ao trafegar o dado via MQTT.
- Config_dashboard: Define a estrutura com que deve se dar a apresentação de dados no *dashboard* do aplicativo. Através do usuário, dispositivo e tópico é possível identificar os eventos utilizados para determinada visualização. A coluna *tp_visao* identifica como será feita a visualização (valor T = textual; valor G = gráfico; valor S = chave liga/desliga; valor P = barra de progresso) – até o momento apenas o modo textual foi desenvolvido no sistema Android. *Ordenacao* define a ordem que a exibição deve ser feita e o campo *configuracao* (ainda não utilizado) permite configurações como parâmetros de visualização do gráfico, por exemplo.

- Acao_esp: esta estrutura armazena as ações tomadas pelo usuário e enviadas para as placas de sensores, possuindo ligação com as estruturas do usuário que tomou a ação, o dispositivo e o tópico a que pertence. O campo *valor* identifica o comando enviado. Já o atributo *tp_status* por padrão assume o valor P (pendente de retorno) e então o comando é enviado. Ao receber o retorno então o status é alterado para C (confirmado) e a data de confirmação é atualizada.
- Evento_sensor: utilizada para armazenar informações provenientes dos sensores, armazena o tópico e dispositivo vinculado, além do valor lido.
- Alarme: permite a configuração de alarmes junto às placas de sensores. É possível identificar o tópico e dispositivo onde será configurada a função, se o mesmo está ativo ou não, além de valores mínimos e máximos. Quando o valor do tópico estiver entre estes valores é então tomada a ação parametrizada (ajustando o *valor* para o tópico *topico_acao)*.
- Log_alarme: armazena o histórico de ativações e desativações de alarmes.

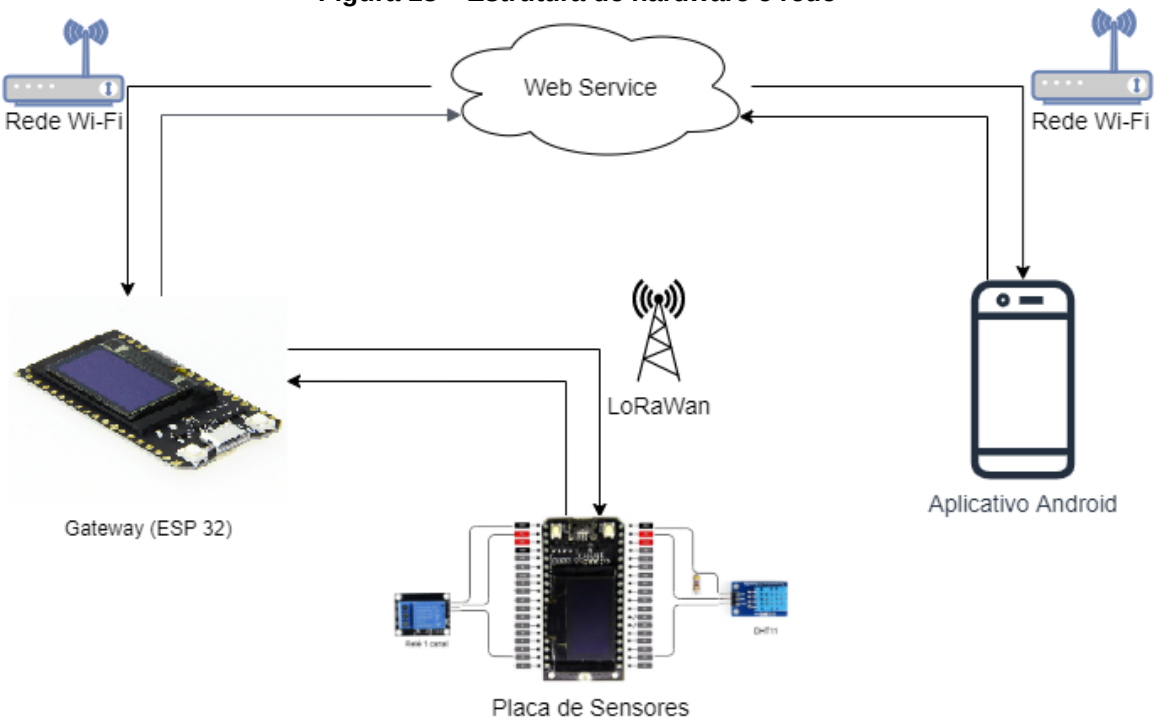
A estrutura de *software*, que será apresentada no próximo capítulo, foi desenvolvida de forma a utilizar da melhor forma a estrutura de dados deste capítulo.

10.3. ESTRUTURAS DE HARDWARE E SOFTWARE

O modelo de hardware é constituído por alguns elementos, os quais utilizam diferentes tecnologias para conexão entre si. A utilização do módulo ESP32 como centralizador e também como placa de sensores se mostrou benéfico pelo baixo custo e alta possibilidade de conexão com dispositivos periféricos. A utilização da rede LoRa para comunicação entre placas, e *Wi-Fi* para as demais ligações do sistema, se mostrou efetiva devido ao alcance, confiabilidade e utilização de energia de cada uma destas, de acordo com cada um dos cenários expostos.

A figura 18 mostra uma visão geral da estrutura, seguida de uma breve descrição.

Figura 18 – Estrutura de hardware e rede



Fonte: elaborado pelo autor

1. Placa de Sensores: Um dispositivo ESP32 responsável pelo sensoriamento e aplicação das ações tomadas pelo usuário. Este dispositivo conecta a um *gateway* (descrito a seguir) através da LoRaWAN, permitindo assim a comunicação mesmo em distâncias onde outras tecnologias não seriam eficientes. No projeto esta estrutura foi representada por um único dispositivo, porém a lógica empregada na programação permite diversos destes conectados ao mesmo *gateway* de forma simultânea.
2. Gateway: O *gateway* tem como função realizar o intermédio entre as placas de sensores e as demais partes do sistema. Utilizando a interface *Wi-Fi* para conexão ao ambiente externo, é feita a conexão com o *Web Service*, onde então o protocolo MQTT auxilia na troca de informações. Além das funções citadas, também é possível utilizar este dispositivo para sensoriamento e tomada de ações.
3. Web Service: O *Web Service* é uma plataforma em nuvem para armazenar as informações vindas do *gateway* e suportar então os acessos dos usuários via aplicativo.
4. Aplicativo *Android*: O aplicativo, desenvolvido na plataforma *Android*, permite ao usuário final realizar *login* no sistema e acessar as informações dos dispositivos vinculados ao mesmo. Além de consultas é permitida a

tomada de ações e gerenciamento de alarmes (processo melhor descrito à frente).

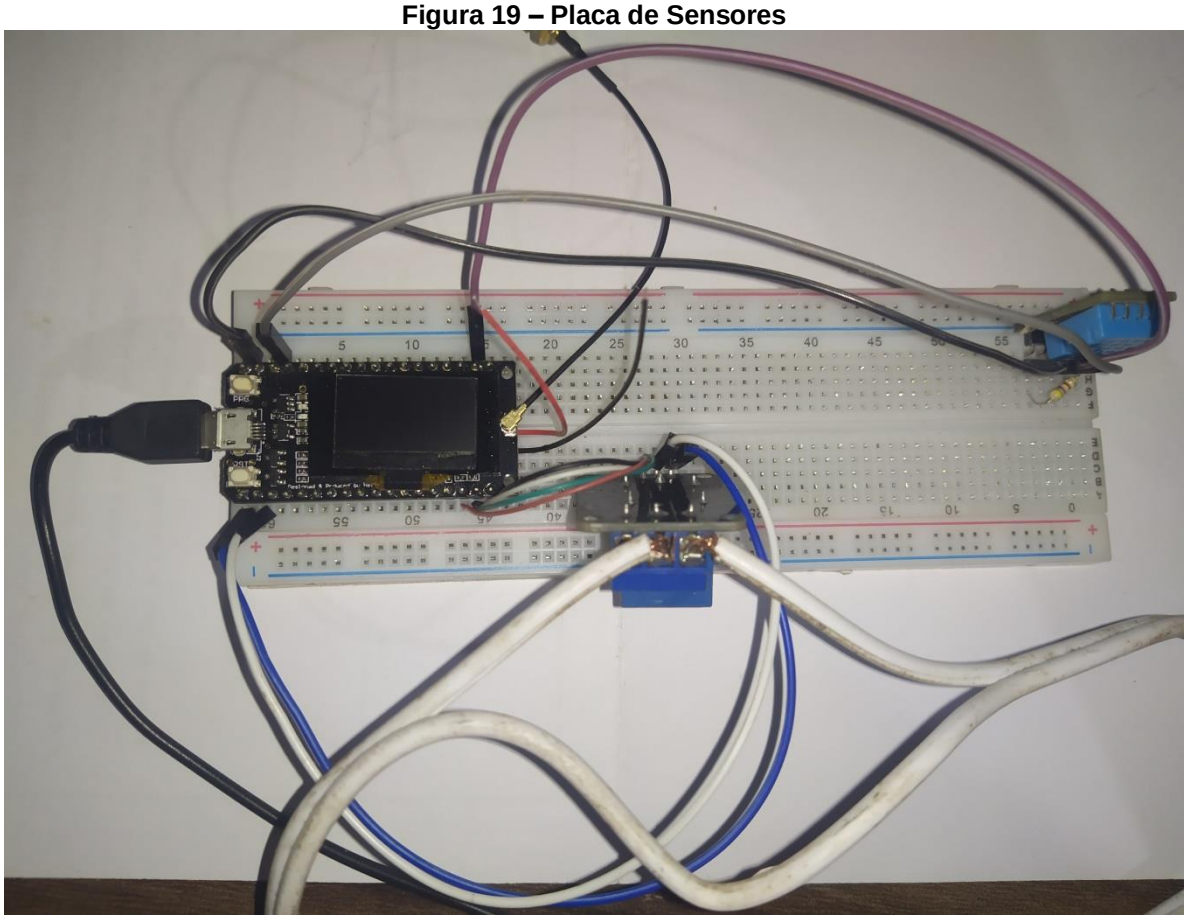
Os dispositivos e partes do sistema citados serão mais bem abordados a seguir, com o detalhamento de fluxos de dados e arquiteturas específicas.

10.3.1. Placa de sensores

A placa de sensores é constituída de um módulo ESP32 com LoRaWAN integrado, capaz de se conectar ao *gateway* para envio e recebimento de dados, além de contar com o incremento de sensores e relés.

A fim de proporcionar os testes do sistema foram inclusos dois periféricos na placa, sendo um sensor de umidade e temperatura (DHT11) e um relé de 1 canal para ativação de equipamentos elétricos. As conexões foram realizadas da seguinte forma durante a prototipagem:

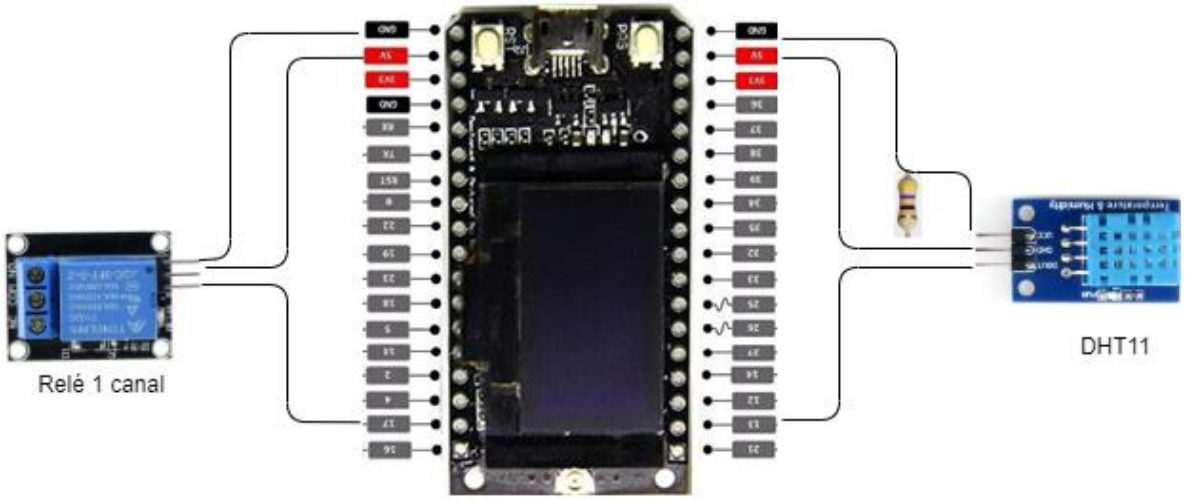
Figura 19 – Placa de Sensores



Fonte: elaborado pelo autor

A figura 19 contém a placa de sensores, com as conexões do sensor de umidade e temperatura e o relé. As conexões são simples, não necessitando de grande estrutura eletrônica. O esquema exibido na figura 20 identifica de forma mais detalhada a conexão das portas do ESP32 tanto para leitura do sensor DHT11 quanto definição da ativação ou não do relé.

Figura 20 – Ligação de sensores e relé à placa de sensores

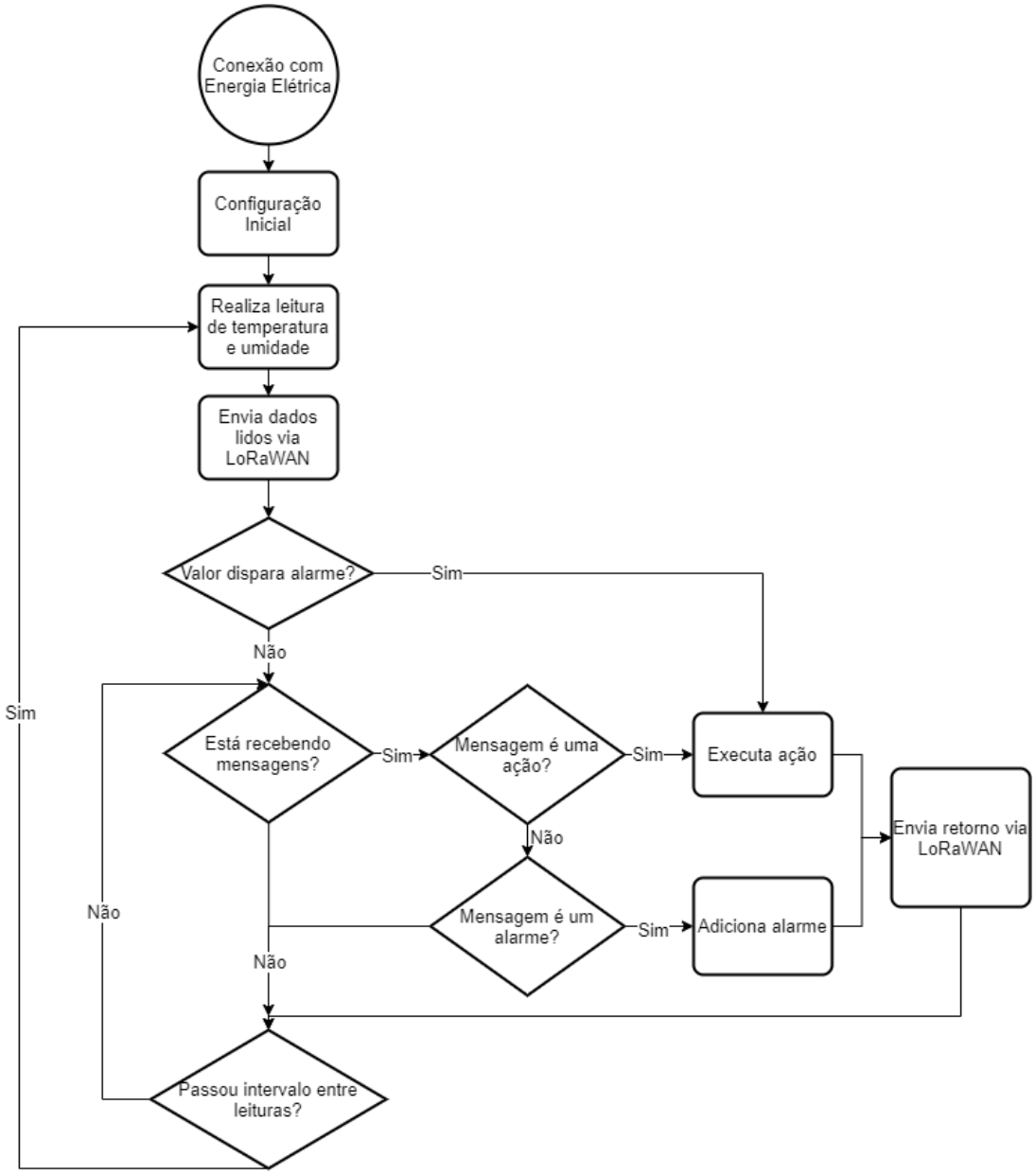


Fonte: elaborado pelo autor

De forma a ficar aderente à estrutura de dados apresentada, há a premissa de que cada placa que se conecte na rede tenha fixo em sua memória um ID único, sendo este utilizado como identificador do dispositivo em todos os fluxos de dados do sistema, incluindo processos internos do *Web Service* e envio de configurações.

Em função do projeto se tratar de um protótipo do sistema, algumas variáveis como as portas onde os periféricos estão conectados são fixas no código, sem possibilidade de parametrização remota ou realizada em tempo de execução.

Figura 21 – Fluxo de processamento da placa de sensores



Fonte: elaborado pelo autor

Conforme figura 21, assim que conectado à energia elétrica, é realizada a configuração inicial da placa de sensores (definindo pinos que serão utilizados pelos sensores e relé, além da inicialização das bibliotecas responsáveis pelo controle do LoRa e dos sensores). Após esta etapa, o sistema passa a realizar ciclos, realizando a leitura dos sensores, identificando se estes disparam algum alarme e processando mensagens recebidas, caso existam. Caso as mensagens sejam comandos de ação ou definição de alarmes o processo é executado e um pacote de retorno é enviado confirmando que a mensagem foi recebida. O formato das mensagens será detalhado à frente.

Figura 22 – Fragmento de código com ciclos da placa de sensores

```
void loop() {
  float temperatura_lida;
  float umidade_lida;
  delay(500);

    temperatura_lida = dht.readTemperature();
    umidade_lida = dht.readHumidity();

    if ( isnan(temperatura_lida) || isnan(umidade_lida) ) {
      oled(true, "Erro ao ler sensor!");
    }
    else
    {
      atualiza_temperatura_max_e_minima(temperatura_lida);
      escreve_temperatura_umidade_display(temperatura_lida, umidade_lida);
    }

  ultimo_envio = millis();
  while (millis() < ultimo_envio + INTERVALO){
    recebe_lora();
  }
}
```

Fonte: elaborado pelo autor

Na figura 22 é possível visualizar de forma codificada o ciclo de funcionamento da placa de sensores. Ao iniciar o ciclo são lidas as informações de temperatura e umidade, as quais passam por uma validação a fim de verificar se não há qualquer erro de leitura, sendo que caso exista alguma oportunidade é exibida mensagem descrevendo a mesma. Com as informações lidas então ocorre a chamada de uma função para atualizar variáveis globais utilizadas para identificar a temperatura máxima e mínima do programa. Assim que todas as informações estão gravadas é então executado processo que exibe texto na tela OLED do dispositivo com as informações coletadas. Este mesmo processo hoje já executa tanto o envio das informações via LoRa quanto a ativação de alarmes.

Após a exibição o processo busca novas mensagens recebidas através do *recebe_lora*.

Figura 23 – Fragmento de código com processamento de tópicos na placa de sensores

```
void escreve_temperatura_umidade_display(float temp_lida, float umid_lida)
{
  char   v_numero[20] = {0};
  String str_temp         = "";
  String str_umid         = "";
  String str_temp_max_min = "";
  /* formata para o display as strings de temperatura e umidade */
  str_temp = "Temperatura: " + float_para_string(temp_lida) + " C";
  str_umid = "Umidade: " + float_para_string(umid_lida) + "%";
  str_temp_max_min = "Min/Max: " + float_para_string(temperatura_min) + "C
                   + float_para_string(temperatura_max) + "C";

  oled(true,  str_temp);
  oled(false, str_umid);
  oled(false, str_temp_max_min);
  envia_lora (TOP_TEMPERATURA, float_para_string(temp_lida));
  envia_lora (TOP_UMIDADE    , float_para_string(umid_lida));

  executa_alarme(TOP_TEMPERATURA, temp_lida);
  executa_alarme(TOP_UMIDADE    , umid_lida);
}
```

Fonte: elaborado pelo autor

A figura 23 evidencia o processo citado para processamento das informações lidas (visível na figura 21) formata as informações para que sejam exibidas no display, exibição esta feita pelo processo *oled*, chamado com um parâmetro indicando se a tela deve ser limpa ou não (respectivamente valores *true* e *false*), além do texto que deve ser impresso. Os processos *envia_lora* e *executa_alarme* serão abordados de forma profunda na seção 10.4.

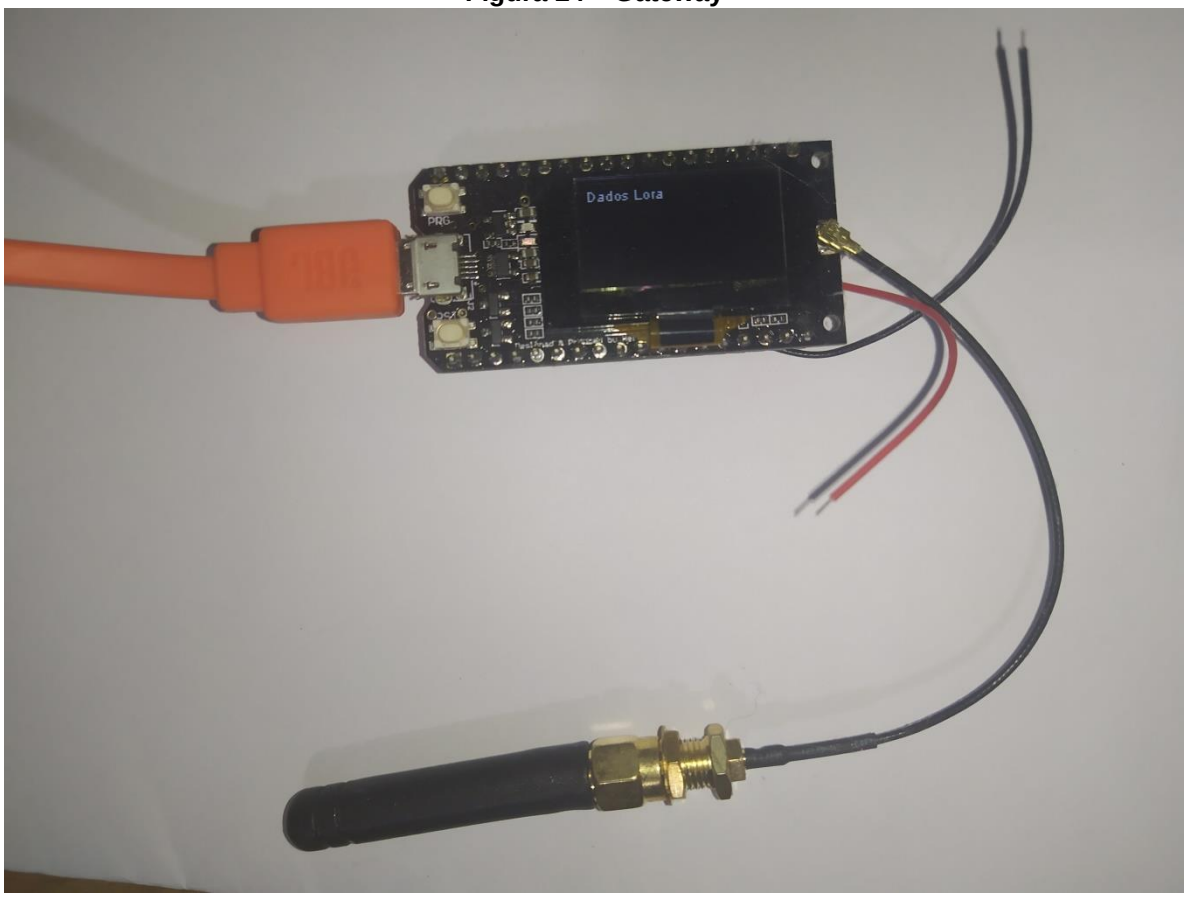
Sempre que um comando de definição de alarmes é recebido, o mesmo é armazenado em um vetor de alarmes. Ao realizar qualquer leitura de sensores o vetor é lido e verificado posição a posição, confrontando com o tópico e valor lido. Para que a ação configurada do alarme seja executada é necessário que o valor do sensor esteja entre os valores mínimos e máximos parametrizados. Caso seja recebido um comando para desativar determinado alarme, o mesmo é excluído do vetor através do atributo *id_alarme*.

10.3.2. Gateway

Desenvolvido também sobre um módulo ESP32 LoRaWAN, permite sua utilização para sensoriamento e gerenciamento de alarmes assim como a placa de

sensores. Em função deste recurso não ter sido utilizado no dispositivo, não há qualquer adição de *hardware* ao mesmo, como é possível visualizar na figura 24. A antena LoRa utilizada é a padrão do fabricante.

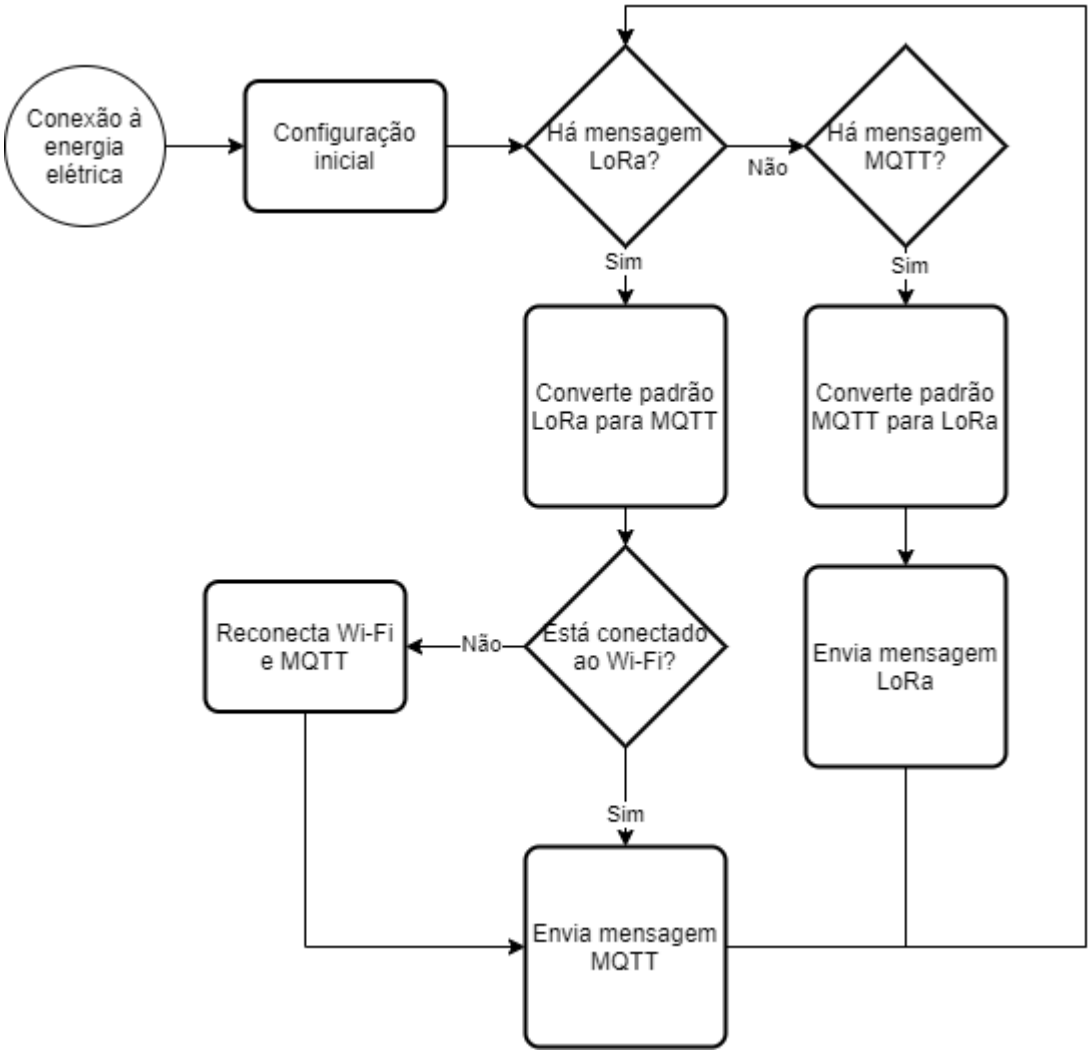
Figura 24 – Gateway



Fonte: elaborado pelo autor

Sua principal função no contexto é a concentração e transmissão de valores entre as placas de sensores e o *Web Service*, sendo esta segunda etapa realizada através do protocolo MQTT.

Figura 25 – Fluxo de processamento do gateway



Fonte: elaborado pelo autor

Conforme figura 25, ao iniciar o *gateway*, automaticamente é feita a configuração inicial, onde é realizada a conexão com o *Wi-Fi* e servidor MQTT, além da inicialização do rádio LoRa. Em seguida são iniciados ciclos para gerenciamento das mensagens trafegadas pelo dispositivo. Quando é detectada uma mensagem LoRa, a mesma é convertida para o padrão MQTT (ambos os padrões serão detalhados à frente) e enviada a mensagem via rede *Wi-Fi*. Caso por algum motivo a conexão com o *Wi-Fi* seja perdida neste meio tempo então esta é restabelecida antes do envio.

Ao receber uma mensagem via MQTT, a mesma é recebida, convertida de MQTT para o padrão LoRa e então enviado via rádio LoRa para a placa de sensores.

10.3.3. Web Service

O *Web Service* é uma das partes fundamentais da plataforma. Além de realizar todo o armazenamento dos dados acaba por suportar o usuário do aplicativo *Android*. Desenvolvido na linguagem *Java* através da IDE *Netbeans IDE 8.2* e *JDK 1.8*, possui instanciado um cliente MQTT que ao ser iniciado se inscreve nos tópicos necessários para que receba as informações provenientes do *gateway*. O servidor utilizado para executar o serviço é o *GlassFish Server*.

A estrutura do desenvolvimento possui alguns pacotes, onde as classes Java criadas foram agrupadas conforme sua funcionalidade, conforme citado no quadro 5.

Quadro 5 – Estruturas de classes Java do Web Service

(continua)

Pacote (função)	Arquivo	Observação
EFS.conexao (classes utilizadas para conexão à base de dados)	Conexao.java	Responsável por gerar a conexão à base de dados
	ParametrosSGBD.java	Retorna os parâmetros para conexão
	ParametrosSGBD.xml	Possui os parâmetros para conexão
EFS.estruturas (estruturas de cadastro	Acao_esp.java	
	Alarme.java	
	Config_Dashboard.java	
	Dispositivo.java	
	EfsDataSet.java	
	EstruturaBase.java	
	EventoSensor.java	
	Registros.java	

Quadro 5 – Estruturas de classes Java do Web Service

(conclusão)

Pacote (função)	Arquivo	Observação
EFS.estruturas (estruturas de cadastro)	Topico.java	
	Usuario.java	
EFS.mqtt (utilizadas para conexão MQTT)	Ouvinte.java	Cria *listenner* MQTT
	PadraoMQTT.java	Possui Padrões para a comunicação com *gateway*
	clienteMQTT.java	Configura o cliente MQTT
EFS.server (*Web Service*)	EfsService.java	possui os processos para chamada SOAP
EFS.uteis	EfsException.java	Classe para tratamento de erros
	EfsUtil.java	Processos úteis como conversão de valores

Fonte: elaborado pelo autor

Para a criação do cliente MQTT foi utilizada a biblioteca *Eclipse Paho MQTT versão 3.1*, disponível no site oficial da *Eclipse*. Este permite a criação de um *listenner* capaz de se inscrever em tópicos junto ao servidor MQTT e então passa a receber as mensagens dos mesmos. Ainda permite o envio de informações, também no formato tópico e mensagem.

Em função de o presente trabalho ter como objetivo permitir ao usuário visualizar os valores lidos pela placa de sensores e a tomada de ações, foram desenvolvidas algumas funções para apoiar o funcionamento do dispositivo, como:

- Funções de buscas de estruturas:
    - Dispositivos vinculados a determinado usuário;
    - Busca de determinado dispositivo;
    - Listagem de tópicos existentes;
    - Busca de determinado tópico;
    - Tópicos associados a um dispositivo;

  - Busca de alarmes existentes, ativos ou não, para determinado usuário;
  - Busca de determinado usuário por e-mail e senha;
- Tomada de ações:
  - Cadastro de um novo usuário;
  - Ativação ou desativação de alarmes;
  - Cadastro de ações a serem enviadas para a placa de sensores.

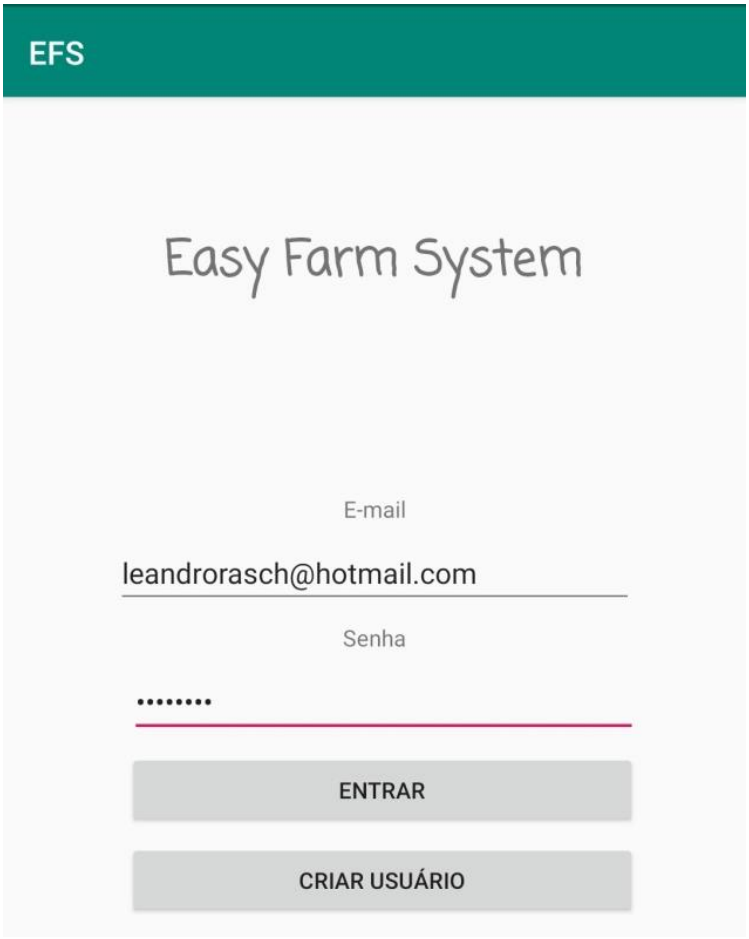
O funcionamento das tomadas de ações será abordado no tópico 10.4.

Na próxima seção serão abordadas algumas características do aplicativo Android (quarta parte da plataforma), a qual utiliza as funções acima citadas.

10.3.4. Aplicativo Android

O aplicativo desenvolvido na plataforma *Android* é o ponto visível do usuário, através deste que o mesmo visualiza as informações sensoriadas e toma as ações no aplicativo. A seguir serão observadas as telas desenvolvidas, assim como a explanação a respeito do funcionamento de cada uma das funções.

Figura 26 – Tela de logindo aplicativo



Fonte: elaborado pelo autor

A tela de login desenvolvida (figura 26) é simples, permitindo a inserção de e-mail e senha para identificação do usuário e acesso à plataforma. Após inserir as informações e clicar no botão “Entrar” os dados são enviados ao *Web Service* que, ao identificar o usuário retorna um registro completo com as demais informações do mesmo. Caso não haja sucesso na busca então é exibido erro informando que as credenciais são inválidas.

Ao clicar no botão “Criar usuário” é aberto novo *canvas*, onde é possível criar um usuário novo para acesso.

Figura 27 – Tela de cadastro de usuários

Fonte: elaborado pelo autor

Na tela de criação de usuários (figura 27), o informar o nome de usuário, e-mail para acesso e senha (a qual é necessário repetir), é possível clicar no botão “Cadastrar usuário”, onde as informações são enviadas ao *Web Service*, que retorna com um registro de usuário a fim de confirmar o cadastro. Após o processo então é fechado o *canvas*, permitindo o *login* conforme já descrito anteriormente.

Ao cadastrar novo usuário são feitas algumas validações: 1 – todos os campos devem estar preenchidos; 2 - o e-mail informado é validado para que contenha “@” em seu texto; 3 - os dois campos para digitação da senha são comparados a fim de garantir que possuem o mesmo valor. Caso alguma das validações não seja atendida então é exibida mensagem indicando o que deve ser corrigido.

Figura 28 – Visualização do dashboard e tomada de ações

Fonte: elaborado pelo autor

Ao efetuar *login* é carregada a tela principal (figura 28), onde é exibido o *dashboard*, que por sua vez é carregado com base na estrutura *config_dashboard* (detalhes no tópico 10.2), acessando as informações dos tópicos necessários no servidor. Até o momento foi utilizado apenas a configuração de exibição textual, onde no exemplo da imagem é identificado o tópico e o último valor lido. O modelo de dados já foi desenvolvido visando permitir que sejam feitas visualizações com o histórico de forma gráfica, porém esta não foi desenvolvida devido ao foco do projeto ser a comunicação entre as diferentes partes do sistema.

Após a exibição dinâmica há um espaço onde o usuário pode cadastrar ações para serem enviadas à placa de sensores. Ao informar o código do tópico e o valor que deve ser enviado e clicar em “Salvar” o comando é enviado ao servidor, que por sua vez grava a informação na base de dados e envia ao *gateway* para que seja entregue para a placa de sensores. Até o momento há apenas a confirmação de que o comando foi salvo no servidor, mas não que de fato tenha sido executado na placa de sensores.

O botão “Alarmes” abre uma nova tela, onde é possível visualizar e gerenciar todos os alarmes vinculados ao usuário.

Figura 29 – Tela de controle de alarmes

Fonte: elaborado pelo autor

Ao carregar a tela, é buscado junto ao servidor um vetor com os registros de alarme, sendo exibidos então o dispositivo onde o mesmo é executado, o tópico que passa a ser monitorado, a faixa de valores mínimos e máximos e se está ativo. No exemplo do tópico “tempera” (referente à temperatura) da figura 29 o disparo será realizado quando o sensor ler valores entre 25,00 e 100,00.

Ao alterar o status do *switch* e clicar em “Salvar” é enviado então um comando para o *Web Service* onde o registro do alarme é atualizado e o comando de ativação é enviado ao *gateway* via MQTT e este por sua vez replica à placa de sensores.

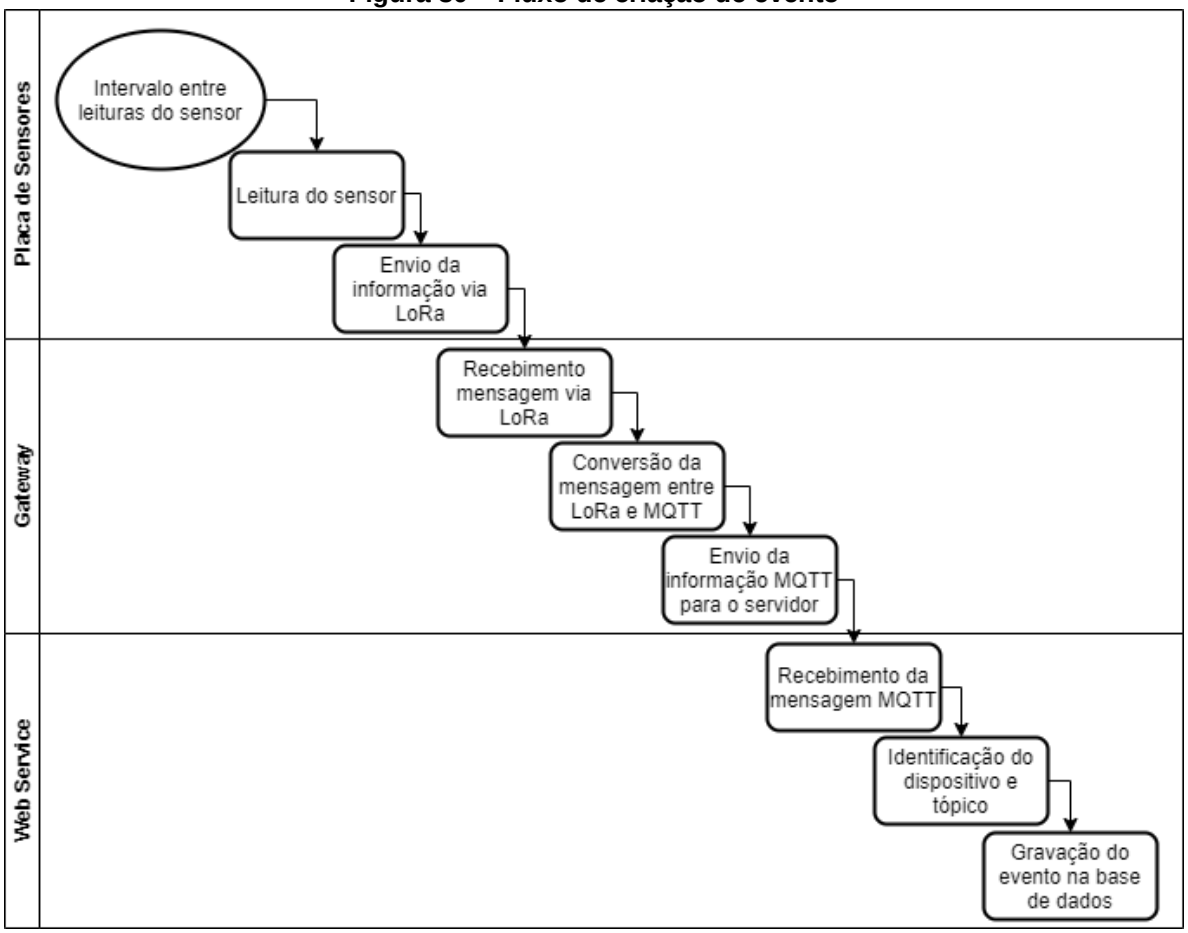
10.4. COMUNICAÇÃO ENTRE ESTRUTURAS DE HARDWARE E SOFTWARE

Esta seção tem por objetivo descrever o fluxo da informação entre as estruturas do sistema, desde a placa de sensores até o aplicativo Android, separadas entre eventos, ações e alarmes. O fluxo de dados, apesar de transparecer baixa complexidade, necessita de um protocolo específico para que todas as informações necessárias sejam contempladas na mensagem e entendível em todos os subsistemas.

10.4.1. Eventos

Os eventos, assim como sinalizados anteriormente, são gerados sempre que um sensor é lido pela placa de sensores.

Figura 30 – Fluxo de criação de evento



Fonte: elaborado pelo autor

Conforme é possível visualizar no fluxo da figura 30, um evento sempre nasce na placa de sensores. Após a validação interna identificar que já passou o intervalo parametrizado desde a última leitura é então lido o valor do sensor, já ligado internamente a um tópico. Após a leitura do sensor a informação é colocada em um padrão do sistema (visível na figura 31) e então enviado via LoRaWAN para o *gateway*.

Figura 31 – Fragmento de código com envio de mensagem LoRa

```
void envia_lora (String ptopico, String pmensagem){
  oled(false, "Enviando Lora");
  String dispositivo = DEVICE + SEPARADOR;
  if (ptopico == TOP_ALARME || ptopico == TOP_ACAO || ptopico == TOP_CONFIGU){
    dispositivo = "";
  }

  LoRa.beginPacket();
  LoRa.print(SETDATA + SEPARADOR + dispositivo + ptopico + SEPARADOR_TOP + pmensagem);
  LoRa.endPacket();
  oled(false, SETDATA + SEPARADOR + dispositivo + ptopico + SEPARADOR_TOP + pmensagem);
  oled(false, "Mensagem enviada");
  delay(500);
}
```

Fonte: elaborado pelo autor

O processo que envia as informações via LoRa (figura 31) possui como parâmetro o tópico e a mensagem que será enviada. A comunicação LoRa foi desenvolvida já utilizando o conceito de tópicos. Para a leitura de um sensor é enviada a palavra “SETDATA”, seguida do tópico, o qual é concatenado com o código do dispositivo, e a mensagem. Importante sinalizar que entre as informações existem separadores configurados tanto na placa de sensores quanto no *gateway*. Ponto e vírgula indica o separador de campos dentro do tópico ou mensagem, enquanto o ponto de exclamação é utilizado para separação entre tópico e mensagem.

Utilizando um cenário em que o sensor de umidade da placa com código 1 leu o valor de 88.00, o comando enviado será formatado conforme quadro 6.

Quadro 6 – Composição de mensagem LoRa para eventos

(continua)

Informação	Valor
Identificador de envio de mensagem	SETDATA
Separador	;
Dispositivo	1
Separador	;
Tópico	umidade

Quadro 6 – Composição de mensagem LoRa para eventos

(conclusão)

Informação	Valor
Separador entre tópico e mensagem	!
Mensagem	88.00
Mensagem LoRa	SETDATA;1;umidade!88.00

Fonte: elaborado pelo autor

A partir do momento em que o *gateway* recebe a mensagem, é verificado se o comando é do tipo SETDATA. Caso sim a mensagem é transformada para o envio via MQTT.

Figura 32 – Fragmento de código com recepção de mensagem LoRa

```
void recebe_lora (){
  int     tam_pacote = LoRa.parsePacket();
  String  mensagem = "";
  int     contador = 0;

  oled(true, "Dados Lora");
  if (tam_pacote > SETDATA.length()){
    while(LoRa.available()){
      mensagem += (char) LoRa.read();
    }
    int index = mensagem.indexOf(SETDATA);
    if (index >= 0){
      String dado_recebido = mensagem.substring(SETDATA.length());
      int index_msg = mensagem.indexOf(SEPARADOR_TOP);

      String device        = mensagem.substring(SETDATA.length()+1, mensagem.indexOf(SEPARADOR));
      String topico        = mensagem.substring(SETDATA.length()+1, index_msg);
      String valor         = mensagem.substring(index_msg+1);
      oled(false, topico);
      oled(false, valor);
      mqtt_publica (topico, valor);
    }
  }
}
```

Fonte: elaborado pelo autor

Na figura 32 é possível verificar a visão sistêmica, onde ao receber os dados, os mesmos são armazenados *byte* a *byte* em uma variável do tipo *String*, e a partir disto é verificado se a mesma inicia com o comando SETDATA. Caso positivo a mesma é quebrada em tópico e mensagem (através do separador de tópicos) e então publicada via MQTT. A mensagem citada anteriormente no envio via LoRa é formatada conforme quadro 7.

Quadro 7 – Quebra da mensagem LoRa

Mensagem	Valor
Mensagem inicial	SETDATA;1;umidade!88.00
Tópico	1;umidade
Mensagem	88.00

Fonte: elaborado pelo autor

Figura 33 – Fragmento de código com publicação MQTT

```
void mqtt_publica (String ptopico, String pmensagem) {
  ptopico = mqtt_sufixo+ptopico;
  if (WiFi.status() != WL_CONNECTED) {
    conecta_wifi();
  }
  while (!mqtt_cliente.connected()) {
    if (mqtt_cliente.connect("tcfevlrESP32Client", mqtt_usuario, mqtt_senha)) {
      oled(true, "Conectado ao Broker!");
    }
    else{
      oled(true, "Falha de conexão ao broker!");
    }

  }
  mqtt_cliente.publish(ptopico.c_str(), pmensagem.c_str(), mqtt_qos);
}
```

Fonte: elaborado pelo autor

No processo visível na figura 33, ao publicar a mensagem MQTT é feita a validação se o *Wi-Fi* e o cliente MQTT estão conectados, sendo que se um dos pontos não obtiver sucesso a conexão é refeita para o envio da mensagem. O QoS utilizado por parte do *gateway* é sempre 2 (dois), por este trazer a garantia de recebimento por parte dos leitores do tópico.

Figura 34 – Fragmento de código com recepção de mensagem MQTT

```
try{
    Map<String, String> t = PadraoMQTT.topicoRetorno(topico);
    Topico      topic       = new Topico(t.get("topico").toUpperCase());
    Map<String, String> m = PadraoMQTT.mensagemRetorno(topic, mensagem);

    if (topic.getTp_topico().equals("S")){
        Dispositivo dispositivo = new Dispositivo(EfsUtil.StringToInteger(t.get("dispositivo")));
        EventoSensor evento = new EventoSensor(dispositivo.getId_dispositivo(), topic.getId_topico(),
        try{
            evento.getSQLInsert().execute();
        }catch (SQLException e){
            System.out.println(evento.getSQLInsert().toString());
            System.out.println("Erro ao inserir evento - ID: "          + evento.getId_evento() +
                                                    " Dispositivo: " + evento.getId_dispositivo()
                                                    " Tópico: "      + evento.getId_topico() +
                                                    " Valor: "       + evento.getValor() +
                                                    " - "            + e.getMessage());
        }
```

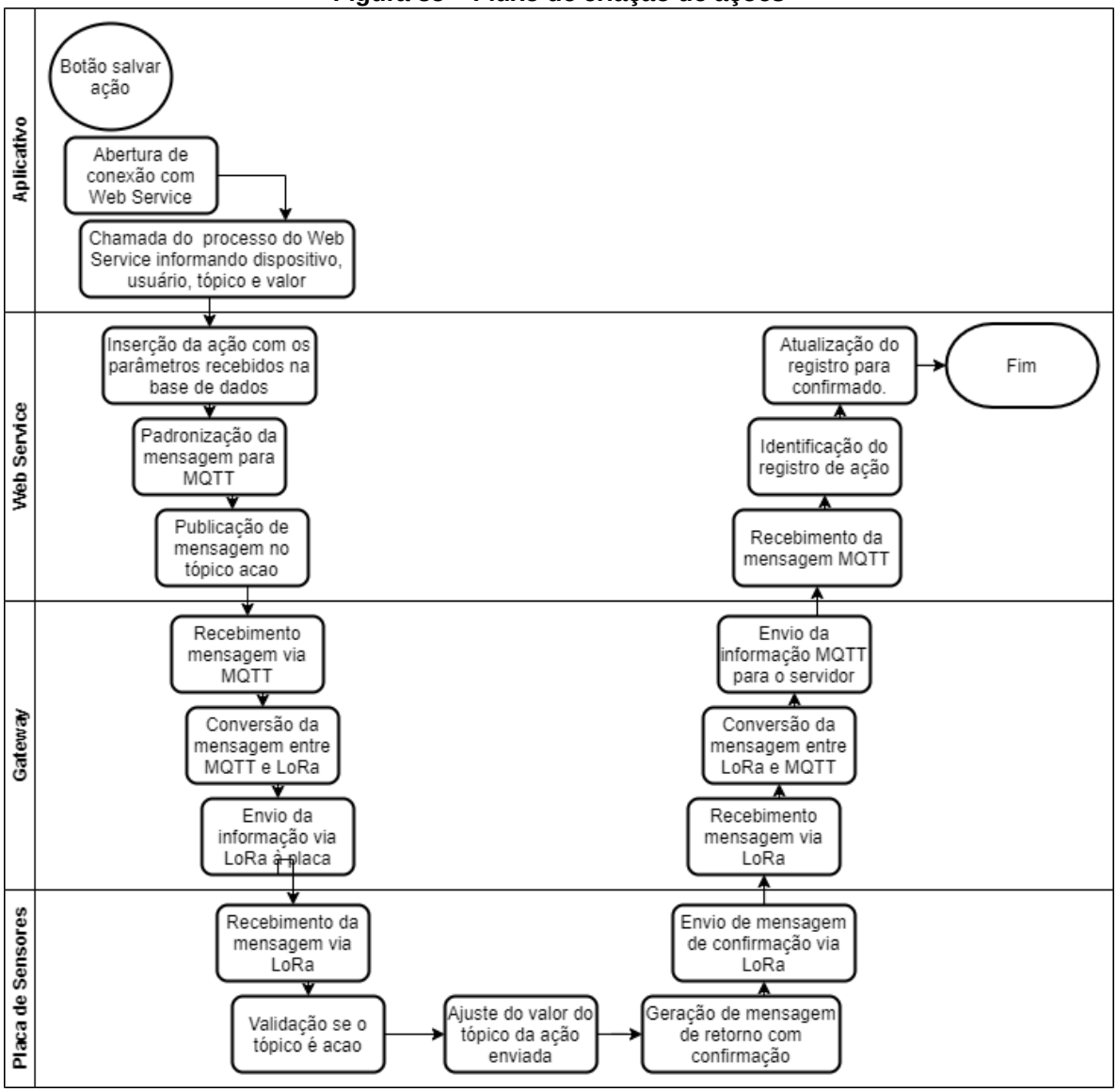
Fonte: elaborado pelo autor

Quando a mensagem chega ao *Web Service* o ouvinte quebra o tópico em um *Map*, podendo haver os campos “dispositivo” e “topico” (figura 34). Ao buscar o cadastro do tópico na base de dados é verificado se o tipo do mesmo é sensor (S), e, sendo positivo busca também o dispositivo origem, salvando o evento na base de dados. Caso ocorra qualquer erro no processo é salvo erro no *log* do servidor, porém não há ainda visibilidade disto para o usuário ou retorno à placa de sensores.

10.4.2. Ações

Através do aplicativo Android é possível tomar ações como, por exemplo, ativar um dispositivo através da placa de sensores. Este é um tipo de informação que necessita de todas as partes do sistema e conta com o rastreamento da informação, sabendo se o fluxo foi completo através de confirmação no *Web Service*. Embora não haja grandes desvios no fluxo de dados do envio de uma ação, gera certa complexidade devido às diferentes tecnologias envolvidas e ao controle necessário para que não se perca qualquer parte da informação.

Figura 35 – Fluxo de criação de ações



Fonte: elaborado pelo autor

A figura 35 deixa claro o fluxo, que parte da tomada de ação do usuário via aplicativo, passa pelo *Web Service* onde é salvo o registro, chegando até a placa de sensores, onde a ação é de fato executada e uma mensagem de confirmação é gerada.

Conforme visto na seção 10.3.4, na tela principal do aplicativo é possível identificar o tópico para o qual será enviada a ação e qual o valor que será adicionado. Ao clicar em “Salvar” é aberta a conexão com o *Web Service* e as informações são enviadas. Na figura 36 é possível visualizar parte do código onde é realizado o processamento.

Figura 36 – Fragmento de código com geração de mensagem de configuração de ação

```
Acao_esp acao = null;
try{
    acao = new Acao_esp(id_dispositivo, id_usuario, id_topico, valor);
    acao.getSQLInsert().execute();

    String topico = "acao";
    String mensagem = PadraoMQTT.getSTATUS_ENVIO()+PadraoMQTT.getSeparador()
                    + acao.getDispositivo().getId_dispositivo()+PadraoMQTT.getSeparador()
                    + acao.getId_acao_esp()+PadraoMQTT.getSeparador()
                    + acao.getTopico().getDs_topico().toLowerCase()+PadraoMQTT.getSeparador()
                    + acao.getValor();

    cliente.publicar(topico, mensagem.getBytes(), 2, true);
}catch(EfsException e){
    System.out.println(e.getMessage());
}catch(SQLException e){
    System.out.println(e.getMessage());
}
```

Fonte: elaborado pelo autor

Conforme figura 46, o primeiro passo ao receber as informações é salvar o registro de ação com o status P (pendente) na base de dados. A partir disto é gerado o comando da mensagem com os campos necessários, sempre com o ponto e vírgula como separador. Um elemento adicional utilizado nos registros de evento é o status de envio, onde é equalizado entre os sistemas que o valor 0 (zero) é mensagem de envio e 1 (um) é um registro de confirmação.

Quadro 8 – Composição de mensagem MQTT de configuração de ação

(continua)

Mensagem	Valor
Status de envio	0
Separador	;
Dispositivo	1
Separador	;
Identificador da ação	45
Separador	;
Descrição do tópico que sofrerá a ação	push

Quadro 8 – Composição de mensagem MQTT de configuração de ação

(conclusão)

Mensagem	Valor
Separador	;
Valor da ação	L
Mensagem final	0;1;45;push;L

Fonte: elaborado pelo autor

A mensagem simulada no quadro 8 se trata de um comando para definir como “L” o tópico relé do dispositivo 1. No protótipo desenvolvido este tópico está atrelado a um relé na placa de sensores, sendo que quando recebe o valor “L” liga o dispositivo. Desta forma a mensagem é enviada via MQTT utilizando o tópico acao, do qual o *gateway* é cliente no *broker* MQTT.

Figura 37 – Recebimento de mensagem MQTT pelo gateway

```
void mqtt_recebe(char* ptopico, byte* payload, unsigned int length)
{
    String topico;
    String mensagem;
    char c;

    for(int i = 0; i < strlen(ptopico); i++)
    {
       c = (char)ptopico[i];
       topico += c;
    }
    /* Obtem a string do payload (dados) recebido */
    for(int i = 0; i < length; i++)
    {
       c = (char)payload[i];
       mensagem += c;
    }
    topico = topico.substring(mqtt_sufixo.length());
    oled(false, topico);
    oled(false, mensagem);
    delay(1000);

    if (topico == mqtt_topico_config_esp) {
      String device = mensagem.substring(0, DEVICELENGTH);
      String conf   = mensagem.substring(DEVICELENGTH);
      envia_lora(device + topico + conf);
    }else if (topico == mqtt_topico_alarme || topico == mqtt_topico_acao){
      envia_lora(topico+SEPARADOR_TOP+mensagem);
    }
}
```

Fonte: elaborado pelo autor

Quando o *gateway* recebe a mensagem, este armazena o tópico e mensagem recebidos em variáveis *String*. Caso o tópico corresponda à palavra “acao” as duas variáveis são concatenadas (separadas com o separador de tópico) e enviadas via LoRa. Na figura 37 é possível verificar um tratamento para excluir os caracteres de “mqtt_sufixo” do tópico. Este é um tratamento adicionado para permitir também o uso de *broker* MQTT público, adicionando uma espécie de chave para identificar os tópicos que, embora contraindicado devido a questões de segurança dos dados trafegados, é suportado pela plataforma.

Figura 38 – Fragmento de código com tratamento de ação na placa de sensores

```
if (topico.equals(TOP_ACAO)){
  String mensagem_recebida = dado_recebido.substring(dado_recebido.indexOf(SEPARADOR_TOP)+1);

  index = mensagem_recebida.indexOf(SEPARADOR);
  String status = mensagem_recebida.substring(0,index);

  mensagem_recebida = mensagem_recebida.substring(index+1);
  index = mensagem_recebida.indexOf(SEPARADOR);
  String id_dispositivo = mensagem_recebida.substring(0, index);

  mensagem_recebida = mensagem_recebida.substring(index+1);
  index = mensagem_recebida.indexOf(SEPARADOR);
  String id_acao = mensagem_recebida.substring(0, index);

  mensagem_recebida = mensagem_recebida.substring(index+1);
  index = mensagem_recebida.indexOf(SEPARADOR);
  String topico_acao = mensagem_recebida.substring(0, index);

  mensagem_recebida = mensagem_recebida.substring(index+1);
  index = mensagem_recebida.indexOf(SEPARADOR);
  String valor = mensagem_recebida.substring(0, index);

  if (id_dispositivo == DEVICE){
    trata_acao(topico_acao, valor);

    String mensagem_retorno = STATUS_OK + SEPARADOR + id_acao;
    envia_lora (topico, mensagem_retorno);
  }
```

Fonte: elaborado pelo autor

Conforme figura 38, a partir do momento que a placa de sensores identifica o recebimento da mensagem, está é quebrada entre tópico e mensagem através do separador de tópicos. Caso o tópico lido tenha a palavra “acao” a mensagem é quebrada para obter os valores do status de envio, dispositivo, identificador da ação, o tópico para o qual será executado e qual o valor a ser atribuído. O código de dispositivo lido sendo o mesmo do receptor, a ação é executada e uma mensagem de retorno é enviada com o mesmo tópico (“acao”) e a mensagem contendo o valor 1 (um) sendo a confirmação como status de retorno e código da ação.

O intermédio realizado pelo *gateway* entre a mensagem recebida da placa de sensores e a mensagem enviada ao *Web Service* é o mesmo descrito anteriormente para o envio de eventos (identificando tópico e mensagem para o envio). Este por sua vez quebra da mesma forma tanto o tópico quanto mensagem para obter acesso às informações enviadas.

Figura 39 – Recebimento de confirmação do processamento da ação

```
if (topic.getDs_topico().equals("ACAO")){
    if (m.get("status").equals("1")){
        Acao_esp acao = new Acao_esp(EfsUtil.StringToInteger(m.get("id_acao")));
        acao.setDt_confirmacao(EfsUtil.getDataAtualTimestamp());
        acao.setTp_status("C");
        try{
            acao.getSQLUpdate().execute();
        } catch (SQLException e){
            System.out.println("Erro ao atualizar ação: "
                    + acao.getSQLUpdate().toString() + e.getMessage());
        }
    }
```

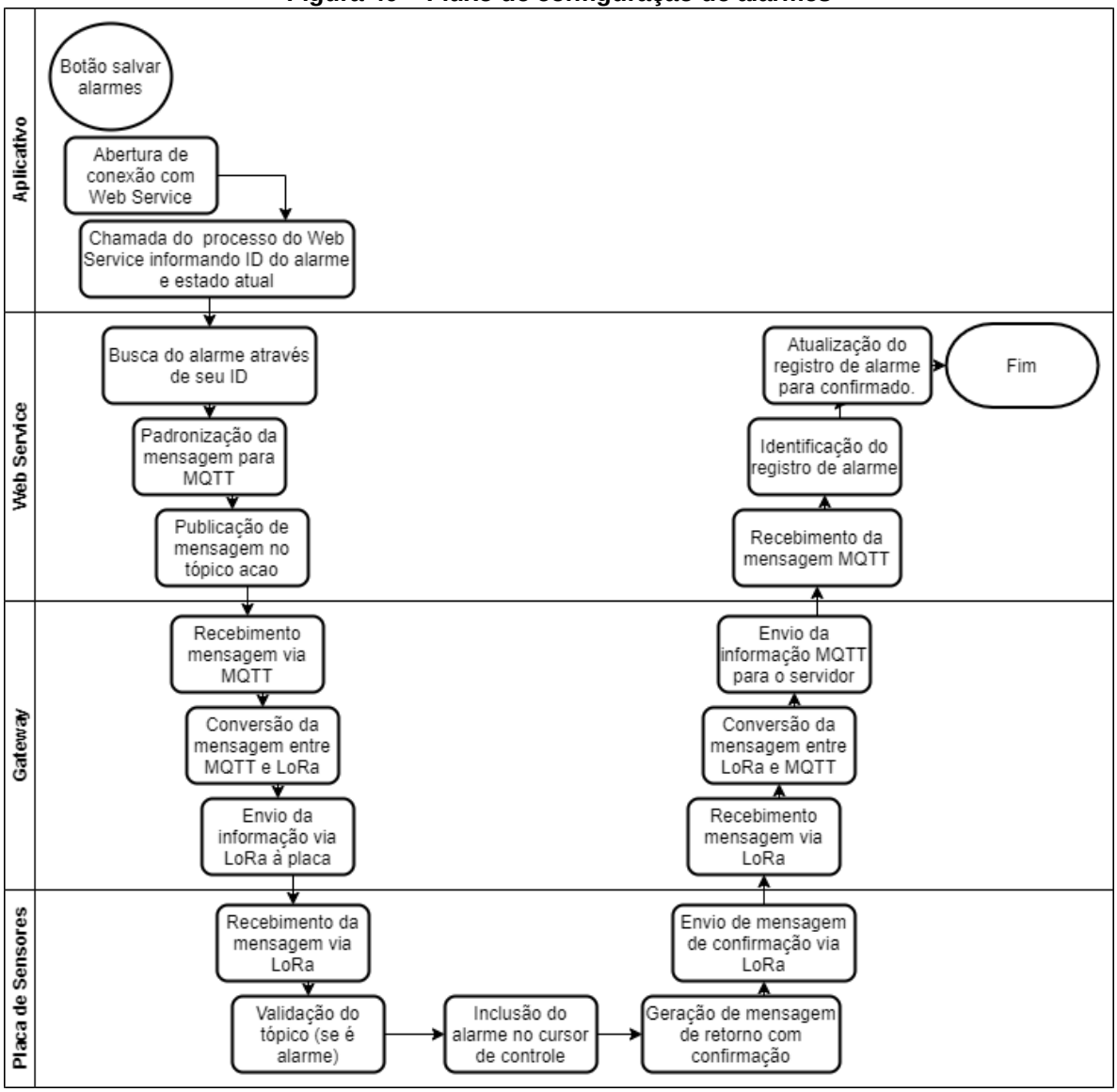
Fonte: elaborado pelo autor

Ao identificar que a mensagem MQTT recebida é do tópico “acao” e o status de envio é 1 (um) identificando o sucesso do processo, o registro da ação é carregado da base de dados através de seu identificador e a atualização é feita com a confirmação do envio do comando e processamento da placa de sensores (figura 39).

10.4.3. Alarmes

O envio do comando de ativação e desativação do alarme é muito similar ao das ações (visto anteriormente), tendo poucas alterações no processo de envio da mensagem pelo *Web Service* para o *gateway* e no tratamento destas pela placa de sensores.

Figura 40 – Fluxo de configuração de alarmes



Fonte: elaborado pelo autor

Segundo fluxo da figura 40, ao clicar no botão “Salvar” na tela de alarmes do aplicativo toda a lista é verificada e, caso tenha sido alterada, é realizada a conexão com o *Web Service* informando o identificador do alarme e seu novo estado.

Figura 41 – Fragmento de código com geração da mensagem MQTT de alarmes

```
Alarme alarme = null;
try{
    alarme = new Alarme(id_alarme);
    alarme.setFl_ativo(fl_ativo);
    alarme.setTp_confirmacao("P");

    String topico   = "alarme";
    String mensagem = PadraoMQTT.getSTATUS_ENVIO()+PadraoMQTT.getSeparador()
                    + alarme.getDispositivo().getId_dispositivo()+PadraoMQTT.getSeparador()
                    + id_alarme+PadraoMQTT.getSeparador()
                    + alarme.getTopico().getDs_topico().toLowerCase()+PadraoMQTT.getSeparador()
                    + alarme.getVlr_minimo()+PadraoMQTT.getSeparador()
                    + alarme.getVlr_maximo()+PadraoMQTT.getSeparador()
                    + alarme.getFl_ativo()+PadraoMQTT.getSeparador()
                    + alarme.getTopico_acao().getDs_topico().toLowerCase() + PadraoMQTT.getSeparador()
                    + alarme.getValor();
    cliente.publicar(topico, mensagem.getBytes(), 2, true);

    System.out.println(alarme.getSQLUpdate().toString());
    alarme.getSQLUpdate().execute();
} catch (SQLException e){
    System.out.println("Erro Busca Alarmes por Usuario: " + e.getMessage());
} catch (EfsException e){
    System.out.println("Erro Busca Alarmes por Usuario: " + e.getMessage());
}
```

Fonte: elaborado pelo autor

Através do identificador é buscado o registro do alarme, que é atualizado na base com o novo estado recebido (figura 41). A mensagem MQTT é enviada com tópico “alarme” e os campos separados por ponto e vírgula como nos demais fluxos, sendo enviadas as informações quanto ao *status* de envio da mensagem, o dispositivo, o identificador do alarme, a descrição do tópico que dispara o alarme, valores mínimos e máximos, o flag identificando se está ativo ou não, o tópico em que dispara a ação do alarme e ainda o valor a ser atribuído a este.

A comunicação MQTT e LoRa ocorre de forma já conhecida e, quando os dados chegam à placa de sensores, é verificado se o tópico da mensagem é “alarme”. Caso positivo, as informações citadas acima são novamente quebradas em variáveis e, caso a informação dispositivo seja igual ao código da placa de sensores então é adicionada uma posição em um vetor de alarmes (figura 42).

Figura 42 – Estrutura de alarme na placa de sensores

```
struct Alarme {
        String topico_alarme;
        String id_alarme = "0";
        float  vlr_minimo;
        float  vlr_maximo;
        String topico_acao;
        String valor;
      } alarmes[20];
```

Fonte: elaborado pelo autor

O vetor é criado com vinte posições e tem como base um *struct*, que por sua vez contém as informações do alarme. Caso o registro recebido seja com o comando para desativar o alarme a posição do vetor é limpa, evitando que seja utilizada. A partir desta alteração é criado um registro de confirmação com o objetivo de informar que o alarme foi ativado ou desativado com sucesso na placa de sensores.

Figura 43 – Fragmento de código com execução de alarmes

```
void executa_alarme(String topico, float valor){
  //oled(true, "ALARME" + topico + valor);
  for (int i = 0; i < 20; i++){
    if (alarmes[i].topico_alarme.equals(topico)){
      if (alarmes[i].vlr_minimo <= valor && alarmes[i].vlr_maximo >= valor){
        trata_acao(alarmes[i].topico_acao, alarmes[i].valor);
      }else if (alarmes[i].valor.equals("L")){
        trata_acao(alarmes[i].topico_acao, "D");
      }
    }
  }
  //delay(1000);
}
```

Fonte: elaborado pelo autor

Quando ocorre a leitura de um sensor é chamado o processo “executa_alarme”, onde todas as posições são percorridas e verificado se em alguma das posições há o tópico lido. Caso positivo é validado o valor lido, que quando estiver entre os valores mínimo e máximo do cadastro executa a ação do alarme.

11. ANÁLISE DOS RESULTADOS

Este capítulo busca trazer uma análise dos resultados obtidos através das movimentações de eventos, ações e alarmes, identificando os pontos positivos e negativos quanto aos objetivos do projeto, com base nos dados lidos pelos sensores e os eventos gerados desde que a integração entre os equipamentos de *hardware* esteve permanentemente ativa.

11.1. MOVIMENTAÇÕES DE EVENTOS

O fato de manter o histórico de eventos lidos torna possível não só uma série de ações como também uma análise de padrões ou anormalidades nas leituras. Utilizando o contexto rural, ao identificar, por exemplo, que o nível de leite gerado pelas vacas de uma propriedade diminui após determinada quantidade de dias sem precipitação ou quando a umidade do solo atinge determinado nível, é possível a tomada de ação externa como a inclusão de insumos que tragam uma maior produção. Da mesma forma este padrão pode vincular um aumento da produção à determinas condições meteorológicas ou outras variáveis que possam ser sensoriadas.

Outro ponto em que a análise da base de dados impacta é na criação de alarmes específicos de acordo com os interesses do usuário, permitindo uma flexibilidade para tomada de ações como ativar equipamentos de controle da temperatura ou ativação de um sistema de irrigação quando as leituras chegam a determinados níveis.

Na tabela 1 há a análise dos dados lidos de temperatura, gerados no período de 08/04/2020 a 18/05/2020, onde são exibidas as seguintes informações:

- Data de leitura;
- Quantidade do tópico no dia;
- Taxa de confiabilidade, calculada pela quantidade de leituras no dia dividida pela quantidade possível no dia (288);
- Valor mínimo lido;
- Valor máximo lido;
- Média das leituras do dia;

- Amplitude térmica gerada pelo valor máximo lido subtraído do valor mínimo do dia;

Tabela 1 – Resumo de movimentações de eventos

(continua)

Data	Quantidade	Confiabilidade	Mínimo	Máximo	Média	Amplitude
08/04/2020	193	67%	13,40	19,00	15,27	5,60
09/04/2020	262	91%	13,30	18,90	15,34	5,60
10/04/2020	266	92%	13,80	19,70	15,49	5,90
11/04/2020	161	56%	13,60	19,60	15,12	6,00
12/04/2020	255	89%	13,00	23,40	16,71	10,40
13/04/2020	261	91%	14,70	26,80	20,08	12,10
14/04/2020	249	86%	14,40	21,30	18,17	6,90
15/04/2020	246	85%	12,90	20,90	16,32	8,00
16/04/2020	244	85%	14,00	21,40	17,21	7,40
17/04/2020	13	5%	16,50	17,00	16,73	0,50
19/04/2020	1	0%	17,90	17,90	17,90	0,00
20/04/2020	242	84%	17,30	26,10	20,24	8,80
21/04/2020	244	85%	18,50	26,80	21,24	8,30
22/04/2020	197	68%	17,30	26,20	20,25	8,90
23/04/2020	11	4%	19,60	20,20	19,82	0,60
24/04/2020	104	36%	20,90	28,00	24,54	7,10
25/04/2020	249	86%	19,40	29,00	22,78	9,60
26/04/2020	235	82%	19,90	23,20	21,67	3,30
27/04/2020	243	84%	19,00	23,10	20,50	4,10
28/04/2020	94	33%	19,00	23,90	20,44	4,90
29/04/2020	147	51%	21,00	24,30	23,09	3,30

Tabela 1 – Resumo de movimentações de eventos

(conclusão)

Data	Quantidade	Confiabilidade	Mínimo	Máximo	Média	Amplitude
30/04/2020	236	82%	15,70	23,50	18,71	7,80
01/05/2020	154	53%	14,30	22,10	16,84	7,80
02/05/2020	27	9%	15,80	19,10	17,62	3,30
03/05/2020	202	70%	13,10	22,10	16,54	9,00
04/05/2020	290	101%	17,10	26,00	23,10	8,90
05/05/2020	68	24%	18,00	36,00	23,20	18,00
06/05/2020	259	90%	11,30	18,50	13,80	7,20
07/05/2020	242	84%	9,70	17,70	12,89	8,00
08/05/2020	255	89%	10,70	20,10	14,29	9,40
09/05/2020	172	60%	15,80	23,70	19,24	7,90
10/05/2020	269	93%	16,50	26,50	20,07	10,00
11/05/2020	253	88%	18,80	28,80	22,68	10,00
12/05/2020	231	80%	18,10	20,20	19,12	2,10
13/05/2020	273	95%	14,70	19,00	17,49	4,30
14/05/2020	211	73%	10,70	16,70	13,63	6,00
15/05/2020	226	78%	10,10	18,50	12,70	8,40
16/05/2020	266	92%	11,90	27,60	15,37	15,70
17/05/2020	187	65%	16,70	23,80	18,64	7,10
18/05/2020	157	55%	16,90	24,00	19,61	7,10
Média	**197,38**	**69%**	**15,63**	**22,77**	**18,36**	**7,13**

Fonte: elaborado pelo autor

Em um sistema de monitoramento com foco na segurança a confiabilidade das informações é de suma importância para que a tomada de ação esteja correta, bem como a consulta por parte do usuário tenha os valores atualizados. Neste primeiro período, com uma confiabilidade dos dados em torno de 69%, se

mostra necessária uma alteração no processo de garantia de entrega das informações para que estas cheguem até a ponta final. É importante salientar que em função de haver desenvolvimento e ajustes nos diversos pontos do sistema houveram diversos momentos de inoperância dos processos. Outros ofensores também foram observados no período de acompanhamento, como quedas da rede elétrica e rede Wi-Fi, além de atualizações no sistema operacional que impactaram neste quesito.

As informações de temperatura mínima, máxima, média e de amplitude térmica, por sua vez, são exemplos de informações uteis para a tomada de decisões, podendo, por exemplo, auxiliar na estimativa de um sistema de controle de temperatura para estabilidade térmica de plantas e/ou animais.

A placa de sensores foi instalada a cerca de 60 metros do *gateway*, sendo utilizada em ambos a antena padrão de fábrica. O fato desta antena não possuir grande alcance também impactou na confiabilidade do sistema e no raio de alcance para comunicação LoRa.

11.2. DEFINIÇÕES DE AÇÕES

Durante os testes das funcionalidades desenvolvidas foram enviados diversos comandos para a placa de sensores com instruções para ligar e desligar o relé fixado na mesma. Este processo é simples e, exceto em casos onde a placa de sensores está fora de alcance, é entregue corretamente.

Tabela 2 – Ações enviadas à placa de sensores

(continua)

Data envio	Valor	Envio	Status	Retorno	Tempo retorno
04/05/2020	L	00:55:29	C	00:57:51	00:02:22
04/05/2020	L	00:57:28	C	00:58:39	00:01:11
04/05/2020	D	00:59:30	C	00:59:38	00:00:08
04/05/2020	D	01:12:23	C	01:12:31	00:00:08
04/05/2020	L	21:11:52	P		
04/05/2020	L	21:13:52	C	21:14:01	00:00:09

Tabela 2 – Ações enviadas à placa de sensores

(conclusão)

Data envio	Valor	Envio	Status	Retorno	Tempo retorno
04/05/2020	D	21:53:26	C	21:53:35	00:00:09
04/05/2020	L	21:53:45	C	21:53:54	00:00:09
04/05/2020	D	22:38:56	C	22:39:05	00:00:09
04/05/2020	D	23:38:49	P		
04/05/2020	D	23:39:54	C	23:39:58	00:00:04
04/05/2020	L	23:40:12	P		
05/05/2020	L	00:33:49	C	00:33:53	00:00:04
05/05/2020	D	01:02:03	P		
05/05/2020	L	22:46:19	P		
05/05/2020	L	22:47:16	P		
05/05/2020	L	22:52:05	C	22:52:09	00:00:04
05/05/2020	D	22:52:25	C	22:52:28	00:00:03
06/05/2020	L	11:59:53	P		
06/05/2020	D	12:00:15	P		
14/05/2020	L	21:40:02	P		
16/05/2020	L	14:49:14	C	14:51:33	00:02:19
16/05/2020	D	14:52:07	C	14:52:12	00:00:05
16/05/2020	L	14:52:22	C	14:52:26	00:00:04
16/05/2020	L	14:53:14	C	14:53:18	00:00:04
16/05/2020	D	14:54:02	C	14:54:07	00:00:05
16/05/2020	L	14:55:47	C	14:55:52	00:00:05
16/05/2020	D	14:55:59	C	14:56:03	00:00:04
16/05/2020	L	15:02:09	C	15:02:52	00:00:43
16/05/2020	D	15:03:11	C	15:03:16	00:00:05
16/05/2020	L	15:03:50	C	15:04:09	00:00:19

Fonte: elaborado pelo autor

Durante os testes, evidenciados na tabela 2, observou-se que quando a mensagem enviada pelo *gateway* não era entregue corretamente à placa de sensores novas mensagens também não eram enviadas, sendo necessário reiniciar o *gateway* para normalização do processo. Após alguns ajustes no processo, colocando alguns pontos de espera no código, o problema foi reduzido de forma significativa, o que pode ser visto nos comandos enviados em 16/05/2020. Embora não tenha sido encontrada bibliografia que aborde um possível motivo para esta oportunidade houve sucesso no ajuste.

O fluxo da ação (desde que a mensagem chega ao *Web Service*, chegando à placa de sensores, e retornando o comando de confirmação) se mostrou rápido na maior parte das execuções, completando todo o processo em menos de 10 segundos. Em outros momentos o comando ficou retido no *gateway*, levando mais tempo para a execução completa.

Quanto à confiabilidade dos comandos que tiveram o retorno de confirmação, estes foram de fato positivos, onde todos os registros enviados de fato foram interpretados pela placa de sensores e houve a alteração de estado do relé.

11.3. ACIONAMENTOS DE ALARMES

Assim como ocorrido com as ações, o envio dos comandos para ativação ou exclusão do alarme na placa de sensores ocorreu com sucesso na maior parte dos testes, porém com alguns momentos onde a distância da placa de sensores impactou na entrega do mesmo.

Quanto ao funcionamento do alarme na placa de sensores não foram observados problemas. Um dos testes da funcionalidade foi feito com o apoio de uma fonte de calor (representada por um secador de cabelo), um ventilador para refrigeração e as seguintes configurações:

Quadro 9 – Configuração do alarme utilizado nos testes

Informação	Valor
Tópico disparador	*tempera* (temperatura)
Valor mínimo	25,00
Valor máximo	100,00
Tópico que deve ser executado	7 (configurado como *rele*)
Valor	L

Fonte: elaborado pelo autor

Conforme é visível no quadro 9, o tópico *rele* deveria ser atualizado com o valor *L* sempre que a temperatura estivesse entre 25 e 100 (graus célsius). Com a temperatura ambiente em torno de 18ºC a fonte de calor foi acionada de forma manual, gerando assim uma temperatura lida no sensor de mais de 30ºC. Desta forma a leitura atendia as condições do alarme, que alterava o estado do relé para ligado, que por sua vez ligava o ventilador já anteriormente conectado ao mesmo. Assim que a leitura possuía valores abaixo de 25ºC, o estado do relé era novamente alterado (desta vez para desligado), desativando assim o ventilador. O teste foi feito diversas vezes, onde sempre a execução ocorreu de forma correta.

Ao longo destas validações identificou-se que pelo fato da leitura estar programada para que ocorra a cada 5 minutos houve momentos em que a temperatura ficou bem elevada em função do aguardo para a leitura.

12. CONCLUSÃO

Conforme discutido ao longo do trabalho, a Internet das Coisas vem crescendo de forma extraordinária nos últimos anos, atingindo cada vez mais áreas da sociedade com o objetivo de automatizar e permitir maior controle em diversas áreas, como saúde, educação, mobilidade urbana e agricultura.

A alta capacidade e confiabilidade das redes sem fio atuais, aliada à capacidade cada vez maior dos microcomputadores e microcontroladores, permitem hoje desenvolver aplicações customizáveis e de baixo custo. Estes pontos associados indicam uma alta aderência das ferramentas ao problema proposto.

O objetivo maior de desenvolver um modelo de sistema para monitoramento de propriedades rurais, permitindo a tomada de decisões com o apoio de dispositivos IoT foi alcançado. Através do uso de diversas tecnologias foi possível contar com uma plataforma onde o usuário pode, de forma remota, monitorar os diferentes sensores que estejam ligados ao *gateway* e *Web Service*, mantendo além das últimas informações recebidas também todo um histórico de dados que auxilia de forma significativa na correta tomada de decisão. Esta por sua vez pode ser feita também de forma remota, gerando comandos a serem utilizados para ativar equipamentos ou então configurar o sistema para que se auto gerencie e ative dispositivos de acordo com as variáveis configuradas.

A forma de concepção do projeto leva em consideração as previsões feitas por Evans (2011) e Carrion e Quaresma (2019), com um aumento exponencial da utilização de IoT nos próximos anos. A estrutura de dados foi desenvolvida com o intuito de atender diversos usuários e diversos *gateways* conectados. A identificação do dispositivo por um ID único e o vínculo deste com o usuário torna possível sua identificação em todos os pontos do sistema.

Evans (2011) também traz em sua bibliografia o fato da quantidade enorme de dados gerados pelos dispositivos de IoT, o que é tratado no projeto pela base de dados que pela sua robustez gera facilidade e rapidez no acesso aos dados, além de evitar a perda dos mesmos por qualquer sinistro.

Por utilizar a conexão *Wireless* entre *gateway* e *Web Service* é possível minimizar problemas de conexão conforme citado por Morimoto (2011). A utilização da tecnologia, atrelada ao uso da *internet* torna possível a utilização de

um *Web Service* centralizado atendendo *gateways* instalados em locais distantes entre si. A rede LoRa por sua vez, utilizada entre o *gateway* e a placa de sensores proporciona longas distâncias, gerando facilidade nas propriedades e a possibilidade de utilização de um único centralizador para todas as placas desta propriedade.

Pelo fato de ser utilizado o *hardware* ESP32 é possível todo o processamento de monitoramento e execução das ações por um baixo custo, além da utilização de comandos no processo que consumam ainda menos energia e assim passem a utilizar baterias para seu funcionamento.

O Mosquitto, apontado por Yuan (2017), traz uma grande facilidade na comunicação, uma vez que por se tratarem de mensagens simples possam tratar as mesmas de forma rápida, podendo integrar outros clientes para recebimento das informações em projetos futuros e complementares.

Através dos diversos testes realizados e abordados é possível concluir que a ferramenta é aderente à necessidade, podendo ainda receber uma série de melhorias a fim de aumentar a confiabilidade de entrega das informações, facilitar a usabilidade por parte do usuário e dinamizar o processo melhorando o mesmo e aumentando as possibilidades de personalização.

No quesito confiabilidade, melhorias futuras que podem ser citadas são: a criação de um processo que garanta o recebimento do evento pelo centralizador; inclusão de validação para que reenvie ações ou comandos de alarmes que não tenham sido confirmados em determinado período; reportar ao usuário o sucesso ou falha no envio da ação ou configuração do alarme.

Para mitigar problemas de comunicação é possível criar um monitoramento das ações e alarmes enviados à placa de sensores, enviando novamente os registros pendentes da confirmação de recebimento a cada determinado período.

A utilização dos sensores em conjunto dos alarmes pode ser utilizada para controle da segurança da propriedade, sendo necessário para isso uma alta confiabilidade e disponibilidade do sistema. Os alarmes citados podem ser utilizados para acionar sirenes ou câmeras mediante acionamento do sensor de presença, por exemplo.

Já quanto à usabilidade do sistema, o maior ponto de melhoria é a tela principal, onde podem ser desenvolvidas novas formas de exibição gráficas e interativas com o usuário. Embora não fosse objetivo deste projeto, o cadastro do

dispositivo deve ser feito em trabalhos futuros a fim de permitir que o usuário, sabendo o código do dispositivo, possa realizar de forma rápida e fácil o vínculo com o mesmo.

Quanto à notificação dos usuários, é viável a criação de notificações a serem enviadas quando determinado alarme é disparado, ou mesmo quando uma ação ou alarme é enviada à placa de sensores.

Quanto à comunicação podem ser abordadas opções de contingência como o uso da rede celular para conexão à *internet* em momentos que a rede *Wi-Fi* estiver inoperante. Conforme abordado durante a revisão bibliográfica esta tecnologia possui boa confiabilidade e alto alcance.

REFERÊNCIAS BIBLIOGRÁFICAS

AMAZON (EUA). **ESP32.** 2019. Disponível em: <https://www.amazon.co.uk/diymore-wireless-Display-Development-Arduino/dp/B078MCR8FY>. Acesso em: 13 nov. 2019.

ANDRADE, Alexandre Acácio de; SOMA, Andrea Tiemi; NAKAMURA, Cassio Eiki Arashiro. **Automação de baixo custo baseada no Raspberry Pi**. ResearchGate**,** Berlim, maio 2016.

BONOTTO, Luiz Gabriel. **Smart City:** Controle de semáforo o utilizando a tecnologia LoRa. 2018. 84 f. TCC (Graduação) - Curso de Curso Superior de Tecnologia em Eletrônica Industrial, Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Florianópolis, 2018.

CARRION, Patrícia; QUARESMA, Manuela. Internet das Coisas (IoT): Definições e aplicabilidade aos usuários finais. **Human Factors In Design**, Florianópolis, v. 15, n. 8, p.49-66, mar. 2019.

CARVALHO, Lucas Azevedo de. **Os dados sobre a violência "do campo" no Brasil:** Análise crítica. Brasília: Câmara dos Deputados, 2018. 36 p.

DUSSAUX, Gautier et al. **Tecnologias de Rede em Telefonia Móvel.** 2010. Disponível em: <https://www.gta.ufrj.br/grad/10_1/movel/index.html>. Acesso em: 10 out. 2019.

ESPRESSIF SYSTEMS (Shanghai). **ESP32 Series:** Datasheet. 3.2 Shanghai: Espressif Systems, 2019. 61 p.

SANTOS, Dave. **A Internet das Coisas:** Como a próxima evolução da Internet está mudando tudo. San Jose: Cisco, 2011.

GHOSLYA, Sakshamma. **Tudo sobre Lora e LoraWAN.** 2017. Disponível em: <https://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html>. Acesso em: 13 nov. 2019.

GONÇALVES, Enyo José Tavares; CORTÉS, Mariela Inés. **Análise e Projeto de Sistemas.** 3. ed. Fortaleza: ED UECE, 2015. 111 p.

HABEKOST, Anderson Felipe. **Diretrizes para introdução dos conceitos da Indústria 4.0 no segmento de manufatura de veículos linha leve.** 2019. 68 f. Dissertação (Mestrado) - Curso de Pós-graduação em Engenharia de Produção e Sistemas, Universidade do Vale do Rio dos Sinos, São Leopoldo, 2019.

MARQUES, Júlio José Branquinho; BOCHIE, Kaylani. **LoRa.** 2018. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/lora/>. Acesso em: 10 nov. 2019.

MORIMOTO, Carlos Eduardo. **Redes:** Guia prático. 2. ed. Porto Alegre: Sul Editores, 2011. 571 p.

PEREIRA, Eduardo da Fonseca; YAMADA, William Jun. **Implementação do protocolo industrial de campo Devicenet em plataforma Raspberry Pi 3 para automação de Maquete Ferroviária.** 2018. 113 f. TCC (Graduação) - Curso de Engenharia Mecatrônica, Universidade de Brasília, Brasília, 2018. Disponível em: <https://bdm.unb.br/bitstream/10483/21320/1/2018_EduardoDaFonseca_WilliamJunYamada_tcc.pdf>. Acesso em: 10 nov. 2019.

PEREIRA, Maurício Fernando Lima; CRUVINEL, Paulo Estevão. **Desenvolvimento de um sistema de coleta automática de dados agrícolas baseado em rede LoRa e no microprocessador ESP32.** Cuiabá, 2019. 6 p.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do Trabalho Científico:** Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013. 277 p.

RASPBERRY PI LTD.. **Raspberry Pi 4 Model B.** Caldecote: Raspberry Pi Ltd., 2019. 13 p.

RIBEIRO, Jonatas Magno Tavares. **Uma aplicação da tecnologia LoRa em um ambiente hospitalar.** 2019. 79 f. TCC (Graduação) - Curso de Engenharia Eletrônica, Universidade Federal de Santa Catarina, Florianópolis, 2019.

ROBUN.IN. **Raspberry Pi 4 Model-B with 4 GB RAM.** 2019. Disponível em: <https://robu.in/product/raspberry-pi-4-model-b-with-4-gb-ram/>. Acesso em: 14 nov. 2019.

SANTOS, Jean Willian; LARA JUNIOR, Renato Capelin de. **Sistema de Automatização Residencial de Baixo Custo Controlado pelo Microcontrolador ESP32 e Monitorado via Smartphone.** 2019. 46 f. TCC (Graduação) - Curso de Curso Tecnológico em Automação Industrial, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2019.

SEIDEL, Álysson Raniere. **Instrumentação Aplicada.** Santa Maria: UFSM, 2011. 98 p. Disponível em: <http://estudio01.proj.ufsm.br/cadernos_automacao/setima_etapa/instrumentacao_aplicada_2012.pdf>. Acesso em: 11 nov. 2019.

SILVA, Alessandro Ramos da. **Aplicação de IoT para coleta e leitura de dados climáticos.** 2019. 68 f. Tese (Doutorado) - Curso de Pós-graduação em Mídia e Tecnologia, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, 2019.
SOMMERVILLE, Ian. **Engenharia de Software.** 9. ed. São Paulo: Pearson Education, 2007. 544 p.

WENDLING, Marcelo. **Sensores.** 2. ed. Guaratinguetá: Unesp, 2010. 19 p.
YUAN, Michael. **Conhecendo o MQTT.** Nova York: IBM, 2017. Disponível em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Acesso em: 15 nov. 2019.