



UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

BeeNotify - Sistema de Alarme para áreas onde a rede elétrica e a internet cabeadas são escassas

Trabalho de Conclusão de Curso

Gideval de Jesus Santos



São Cristóvão – Sergipe

2024

UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO

Gideval de Jesus Santos

**BeeNotify - Sistema de Alarme para áreas onde a rede elétrica
e a internet cabeadas são escassas**

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Rafael Oliveira Vasconcelos

São Cristóvão – Sergipe

2024

*Este trabalho é dedicado a Lucas Souza,
amigo, primo, compadre, apicultor e idealizador da ABMEL,
que de forma indireta, proporcionou a idealização deste projeto.*

*Também gostaria de dedicar este trabalho à memória do meu tio Lourenço, que, mesmo após seu
falecimento, permanece como o pioneiro na criação de abelhas em nossa família.*

Agradecimentos

Gostaria de agradecer, primeiramente, a Deus, por me guiar e me iluminar em todos os momentos de minha vida. Cada oportunidade e vitória, eu devo a Ele.

Agradeço também à minha família que me educou, me apoiou, me incentivou e me deu todas as condições necessárias para que eu pudesse chegar até aqui.

Agradeço à minha esposa, que me deu todo o apoio e incentivo para que eu concluisse a minha graduação.

Agradeço à minha tia Maria da Conceição, pelo apoio e por ter me acolhido, se tornando minha segunda mãe.

Agradeço aos meus irmãos do grupo de oração Caminho de Luz do bairro Rosa Elze, que se tornaram minha família enquanto vive longe da minha cidade natal.

Agradeço ao meu orientador Rafael Oliveira Vasconcelos por ter aceitado me conduzir nesse trabalho, por todas dicas e conselhos compartilhados.

Aos colegas de trabalho Leonardo Trindade, Igor Carvalho, Felipe Ventorim, Paula Schffer, Daniele Carvalho, Julia Aquino, Luca Luz e Gabriel Queiroz, que me auxiliaram neste projeto, me estendendo o braço sempre que solicitado.

A todos: Muito obrigado!

Resumo

Com o avanço da Internet das Coisas (IoT), dispositivos inteligentes têm ganhado destaque, permitindo a automação de tarefas e a troca de informações entre equipamentos, o que contribui para a melhoria da eficiência humana. No agronegócio, essas tecnologias têm sido adotadas para o monitoramento do solo, das lavouras e das máquinas agrícolas, resultando na redução de perdas e no aumento da produtividade. No entanto, muitos pequenos agricultores ainda enfrentam desafios devido à limitação de recursos financeiros. Este projeto visa desenvolver um sistema de alarme de segurança para áreas rurais, utilizando os recursos da IoT, em regiões onde o acesso à internet cabeadas e à rede elétrica é limitado, mas com sinal de telefonia móvel disponível. O sistema utiliza um microcontrolador ESP32 programado em linguagem C, responsável por gerenciar os componentes eletrônicos, incluindo um modem GSM 4G, um sensor de luminosidade e um sensor magnético. O sensor magnético aciona o sistema quando suas partes são desconectadas, e o ESP32, por meio de comandos AT, envia sinais de alerta via protocolo MQTT para um aplicativo móvel, que emite um aviso sonoro sobre a violação no perímetro. Os resultados indicaram que o sistema é eficaz no envio de alertas em tempo real, mesmo em condições de baixa captação de sinal telefônico, e pode ser aplicado para monitoramento preciso em áreas com infraestrutura limitada. Conclui-se que a solução proposta é inovadora, viável e atende às necessidades de segurança do setor rural.

Palavras-chave: sistema de alarme; Internet das Coisas; MQTT, segurança, agronegócio, ESP32, IoT.

Abstract

With the advancement of the Internet of Things (IoT), smart devices have gained prominence, enabling the automation of tasks and the exchange of information between equipment, which contributes to improving human efficiency. In agribusiness, these technologies have been adopted for monitoring soil, crops and agricultural machinery, resulting in reduced losses and increased productivity. However, many small farmers still face challenges due to limited financial resources. This project aims to develop a security alarm system for rural areas, using IoT resources, in regions where access to wired internet and the power grid is limited, but with a mobile phone signal available. The system uses an ESP32 microcontroller programmed in C language, responsible for managing the electronic components, including a 4G GSM modem, a light sensor and a magnetic sensor. The magnetic sensor activates the system when its parts are disconnected, and the ESP32, through AT commands, sends alert signals via the MQTT protocol to a mobile application, which emits an audible warning about a violation of the perimeter. The results indicated that the system is effective in sending real-time alerts, even in conditions of low telephone signal reception, and can be applied for precise monitoring in areas with limited infrastructure. It is concluded that the proposed solution is innovative, viable and meets the security needs of the rural sector.

Keywords: alarm system; Internet of Things; MQTT, security, agribusiness, ESP32, IoT.

Lista de ilustrações

Figura 1 – Versões do Módulo ESP32	16
Figura 2 – Placa de Desenvolvimento ESP32	16
Figura 3 – Placa LiLyGO T-A7670E	17
Figura 4 – Sensor de Luminosidade LDR	18
Figura 5 – Tabela comandos AT	19
Figura 6 – Módulo GSM/GPRS	20
Figura 7 – Sensor Magnético Reed Switch	20
Figura 8 – Sensor Magnético	20
Figura 9 – <i>Power Bank</i>	21
Figura 10 – Arquitetura MQTT <i>Publish/Subscribe</i>	22
Figura 11 – Cobertura 4G	27
Figura 12 – Cobertura 5G	28
Figura 13 – Porteira do Apiário	28
Figura 14 – Entrada do Apiário	28
Figura 15 – Lateral Direito	28
Figura 16 – Fundo	28
Figura 17 – Lateral Esquerdo	29
Figura 18 – Frente	29
Figura 19 – Visão Geral do Sistema	29
Figura 20 – Diagrama de Caso de Uso do Sistema de Alarme	34
Figura 21 – Tela Inicial	39
Figura 22 – Tela Inicial ao clicar no botão para atualizar data	40
Figura 23 – Tela de Alarme	41
Figura 24 – Diagrama do <i>Hardware</i>	42
Figura 25 – Placa do Circuito	51
Figura 26 – Conexão Sensores com T-A7670G	51
Figura 27 – Lógica do <i>Hardware</i>	52
Figura 28 – Lógica do Aplicativo	53
Figura 29 – Impressão LOG Aplicativo	53
Figura 30 – Verifica Conexão Servidor MQTT	54
Figura 31 – Envio Sinal Alerta	55

Lista de quadros

Quadro 1 – Requisitos Funcionais <i>Hardware</i>	30
Quadro 2 – Requisitos Funcionais Aplicativo Móvel	31
Quadro 3 – Requisitos Não Funcionais <i>Hardware</i>	32
Quadro 4 – Requisitos Não Funcionais Aplicativo Móvel	32

Lista de códigos

Código 1 – Conexão com Google Firebase	44
Código 2 – Registro do Usuário	44
Código 3 – Conexão com Servidor MQTT Broker	45
Código 4 – Inscrição em Tópico e Recebimento de Mensagem	45
Código 5 – Publicação em Tópico	45
Código 6 – Verifica Conexão	46
Código 7 – Busca Mensagem de Alerta	46
Código 8 – Carrega Arquivo de Áudio	47
Código 9 – Define Constantes dos Pinos	47
Código 10 – Inicialização Sensores e Modem GSM	48
Código 11 – Conexão com Servidor MQTT	49
Código 12 – Aguarda Mensagem	49
Código 13 – verifica Conexão	50
Código 14 – Envia Comandos AT	50

Listas de abreviaturas e siglas

2G	2 ^a geração de rede móvel
3G	3 ^a geração de rede móvel
4G	4 ^a geração de rede móvel
5G	5 ^a geração de rede móvel
API	<i>Application Programming Interface</i>
Anatel	Agência Nacional de Telecomunicações
AT	<i>Attention</i>
DCOMP	Departamento de Computação
Fax	Facsimile
GPRS	<i>General Packet Radio Service</i>
GSM	<i>Global System for Mobile communications</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of Things</i>
LED	<i>Light Emitting Diode</i>
MMS	<i>Multimedia Messaging Service</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
QoS	<i>Quality of Service</i>
SIM	<i>Subscriber Identity Module</i>
SI	Sistemas de Informação
SMS	<i>Short Message Service</i>
UFS	Universidade Federal de Sergipe
UML	<i>Unified Modeling Language</i>

Sumário

1	Introdução	12
1.1	Objetivos Gerais	13
1.2	Objetivos Específicos	13
2	Materiais e Métodos	15
2.1	<i>Hardware</i>	15
2.1.1	Microcontrolador ESP32	15
2.1.2	Sensor de Luminosidade	17
2.1.3	Módulo GSM/GPRS	17
2.1.3.1	Rede GSM	17
2.1.3.2	Tecnologia GPRS	18
2.1.3.3	Comandos AT	18
2.1.3.4	Módulo GSM/GPRS	19
2.1.4	Sensor Magnético	19
2.1.5	<i>Power Bank</i>	20
2.2	Software	21
2.2.1	MQTT	21
2.2.1.1	Modelo <i>Publish/Subscribe</i> do MQTT	21
2.2.1.2	Níveis de Qualidade	22
2.2.1.3	Persistência de Mensagem	22
2.2.1.4	Considerações	23
2.2.2	React Native	23
2.2.3	NodeJS	23
2.2.4	Expo	24
2.2.5	Google Firebase	24
3	Proposta do Sistema de Alarme	26
4	Alicerces para o Desenvolvimento do Sistema	30
4.1	Requisitos Funcionais	30
4.1.1	<i>Hardware</i>	31
4.1.2	Aplicativo Móvel	31
4.2	Requisitos Não Funcionais	31
4.2.1	Hardware	32
4.2.2	Aplicativo Móvel	32
4.3	Regras de Negócio	32

4.4	Diagrama de Cado de Uso	33
5	Busca de Anterioridade	35
5.1	Resultados	35
5.1.1	Prototipagem de Sistema de Alarme de Incêndio Baseado em Sensor Nanoestruturado Integrado a IoT	35
5.1.2	Fully Automatic Infrared Pyroelectric Home Burglar Alarm Design	36
5.1.3	ESP32 CAM-based Car Security System via Telegram Integration	36
5.1.4	Sistema de geolocalización con alarma y monitoreo basado en IOT para personas con Alzheimer	36
5.2	Conclusão	37
6	Solução Proposta	38
6.1	Protótipo de Telas do Aplicativo	38
6.2	Diagrama do <i>Hardware</i>	38
7	Desenvolvimento e Resultados	43
7.1	Aplicativo	43
7.1.1	Autenticação e Banco de Dados	43
7.1.2	Comunicação com Servidor MQTT Broker	44
7.1.3	Tela Principal	45
7.1.4	Tela de Alarme	46
7.2	Hardware	47
7.2.1	Código	47
7.2.2	Integração de Componentes Eletrônicos	48
7.3	Lógica do Sistema	51
7.3.1	Hardware	52
7.3.2	Aplicativo	52
7.4	Avaliação dos Resultados	52
8	Considerações Finais	56
8.1	Conclusão	56
8.2	Trabalhos Futuros	56
Referências		58

1

Introdução

Nos últimos anos, a Internet das Coisas (IoT - *Internet of Things*) emergiu como uma revolução tecnológica que tem transformado a maneira de interagir com o ambiente ao nosso redor. A capacidade de conectar dispositivos físicos à internet e habilitá-los para coletar, transmitir e processar dados abriu portas para uma infinidade de aplicações inovadoras em diversos campos. A IoT não se limita apenas a conectar dispositivos comuns à internet; ela representa a criação de ecossistemas interconectados de dispositivos inteligentes, capazes de coletar dados em tempo real, comunicar informações vitais e tomar decisões autônomas. A conectividade dos dispositivos e a inteligência embarcada neles estão revolucionando setores como a agricultura, indústria, saúde, transporte e muito mais.

A Internet das Coisas (IoT) permite desenvolver mecanismos para diferentes áreas, como residencial, agropecuária, produção industrial, etc. Alguns exemplos de sua aplicação são: automação residencial, sistema de irrigação e sistema de monitoramento de temperatura, essa é uma pequena parcela de sua aplicabilidade. De acordo com Magrani (2018 apud KIKUCHI; BETTERI; ARAÚJO, 2021), “a IoT aborda um mundo em que os objetos estarão conectados através de redes sem fio e que se comuniquem usando a internet, tornando-se uma rede de objetos inteligentes capazes de realizar vários processos do cotidiano”.

Neste contexto, dispositivos inteligentes desempenham um papel fundamental. São projetados para serem autônomos, adaptáveis e eficientes, e podem ser aplicados em uma variedade de cenários para melhorar a qualidade de vida, otimizar processos e resolver desafios complexos. A interação entre dispositivos inteligentes e sistemas de IoT está redefinindo a forma como problemas tradicionais são abordados, criando soluções inovadoras e acessíveis.

Na zona rural a presença da IoT também está ganhando espaço, mas especificamente na agropecuária, ela está se tornando uma forte aliada do agronegócio.

Com o avanço do estudo sobre a IoT e percepção de sua utilização dentro do agronegócio, principalmente nas atividades agropecuárias, surgem cada vez

mais estudos relacionando estas duas variáveis. Diferentes estudos buscam desenvolver, aplicar ou até observar a utilização e impactos da tecnologia de IoT dentro de atividades pecuárias, agrícolas, de comercialização, logística entre outras ([SILVA; ESPEJO, 2020](#)).

Alinhado ao meio tecnológico, o acesso a internet está se tornando cada vez mais globalizado, em contanto a um smartphone a população brasileira consegue se conectar a rede mundial de computadores, dados do período junho 2023 da Agência Nacional de Telecomunicações (Anatel) mostram que a densidade de acesso à internet banda larga móvel a cada 100 habitantes é de 98,6, enquanto a densidade de acesso à internet via banda larga fixa a cada 100 habitantes é de 21,7 ([ANATEL, 2023b](#)). Fazendo uma análise entre os dados obtidos podemos concluir que a conexão com a internet é mais acessível através dos aparelhos celulares.

O presente trabalho tem como finalidade utilizar o conceito e mecanismos da IoT para desenvolver um sistema de segurança com alarme, voltado para a zona rural. Onde em algumas localidades o acesso a energia elétrica e internet banda larga fixa é mais complexo, mas que disponhem de sinal de telefonia móvel na maioria dos casos. Por meio de conexão via rede da companhia de celular, o dispositivo emitirá sinais de alerta para informar que o local foi violado.

O protótipo do dispositivo que será desenvolvido baseia-se em um caso real, em uma reserva de floresta natural no município de Simão Dias Sergipe, existe um apiário que tem sido alvo de furtos em repetidas ocasiões, gerando prejuízo ao apicultor. Da disposição de onde se encontra o apiário (imagens no [Capítulo 3](#)) e com base nas informações e recursos tecnológicos e eletrônicos para o meio IoT na agropecuária e alcance da rede de telefonia móvel, como mostra a [Anatel \(2023a\)](#) no período de março 80,70% dos moradores de Simão Dias tem cobertura a tecnologia 4G, o equipamento em questão demonstra ser um aliado promissor na mitigação dos furtos. Esse é um caso específico, mas o sistema de alarme não está sujeito apenas a isso, podendo ser aplicado em outras áreas que têm suporte a infraestrutura de comunicação sem fio para dispositivos móveis.

1.1 Objetivos Gerais

Implementar um protótipo de sistema de alarme de segurança para zona rural, tendo conectividade com a internet via rede móvel, o qual emitirá sinais de alerta para o aplicativo de celular quando o perímetro for violado.

1.2 Objetivos Específicos

- Desenvolver o aplicativo móvel do sistema;
- Montar o *hardware* que compõe o sistema com base nos componentes eletrônicos utilizados;

- Testar a solução desenvolvida;
- Implantar e testar a solução em ambiente real.

2

Materiais e Métodos

Este capítulo versará sobre os recursos utilizados para o desenvolvimento do sistema de alarme de segurança.

2.1 *Hardware*

2.1.1 Microcontrolador ESP32

Desenvolvido pela Espressif¹, o microcontrolador ESP32 é uma opção mais avançada em comparação com o Arduino e o ESP8266, oferecendo mais recursos e flexibilidade. Ele possui um número maior de pinos de I/O (entrada/saída), o que permite conectar uma quantidade maior de sensores e periféricos. Dependendo do modelo, sua memória de armazenamento pode chegar a 16MB, possibilitando o desenvolvimento de programas maiores e mais complexos.

Uma das principais vantagens do ESP32 é a conectividade sem fio, suportando Wi-Fi e Bluetooth integrados, o que o torna ideal para projetos de IoT e automação. Além disso, ele pode ser programado em diversas linguagens, incluindo C/C++, MicroPython, Go, Rust e Lua, oferecendo uma ampla gama de opções para desenvolvedores com diferentes níveis de experiência e preferências.

O ESP32 é um poderoso microcontrolador SoC (System on Chip) com Wi-Fi 802.11 b/g/n integrado, Bluetooth de modo duplo versão 4.2 e uma variedade de periféricos. Ele é um sucessor avançado do chip 8266 principalmente pela implementação de dois núcleos com clock em versão diferente que pode chegar a até 240 MHz. Comparado ao seu antecessor, além desses recursos, ele também amplia o número de pinos GPIO de 17 para 36, o número de canais PWM para 16 e é equipado com 4 MB de memória flash (BABIUCH; FOLTYNEK; SMUTNÝ, 2019).

A Figura 1 exibe as duas versões do módulo ESP32 desenvolvido pela Espressif

¹ <https://www.espressif.com/en>

Figura 1 – Versões do Módulo ESP32

Fonte: [espressif \(2023\)](#)

O microcontrolador do ESP32 pode ser encontrado em diversos tipos de placas de desenvolvimento que atende a necessidades específicas de cada projeto, algumas delas são:

- ESP32S NodeMcu ESP-12;
- ESP32 TTGO Mini;
- ESP32 com Suporte de Bateria;
- ESP32 LoRa SX1276 868/915Mhz com OLED;

A [Figura 3](#) exibe um dos modelos de placa mais comum encontrado no mercado

Figura 2 – Placa de Desenvolvimento ESP32

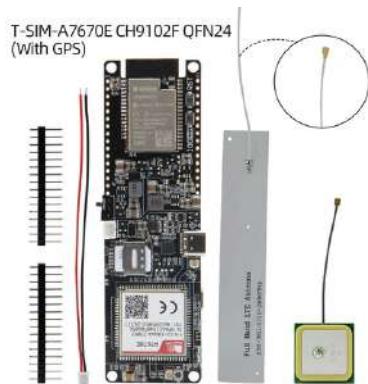
Fonte: [usinainfo \(2024\)](#)

A empresa [LILYGO²](#) desenvolve e comercializa diversas placas de desenvolvimento baseadas no ESP32. Entre elas, destaca-se a T-A7670, pertencente à série T-SIM / T-PCIE. Essa

² <https://www.lilygo.cc/>

placa integra um ESP32 com 24 pinos de comunicação, um modem GSM A7670 4G com antena, um *slot* para cartão microchip, conector USB tipo C, suporte para bateria, além de pinos de alimentação. Dependendo da aplicação, também pode incluir uma antena GPS. A figura abaixo ilustra um modelo dessa placa.

Figura 3 – Placa LiLyGO T-A7670E



Fonte: [LILYGO \(2024\)](#)

2.1.2 Sensor de Luminosidade

Um sensor de luminosidade é um componente eletrônico que capta a intensidade da luz ambiente gerando sinais de saída proporcional a intensidade da iluminação detectada. Um sensor de fotocélula é um tipo de resistor que muda sua resistência com base na quantidade de intensidade de luz experimentada. Ele converte a energia da luz em energia elétrica para produzir tensão ou corrente. A resistência do aparelho é inversamente proporcional à quantidade de intensidade luminosa, ou seja, a resistência diminui com o aumento da quantidade de luz (KHAN, 2022). A Figura 4 apresenta um de vários modelos de sensores de luminosidade presentes no mercado:

2.1.3 Módulo GSM/GPRS

Antes de ser abordado o módulo propriamente dito, será discutido sobre as tecnologias e recursos por ele utilizados.

2.1.3.1 Rede GSM

A rede *Global System for Mobile communications* (GSM) é um padrão de tecnologia mundial, mais conhecido comumente com rede 2G, através dela, usuários de telefone celular podem se comunicar com pessoas de diferentes lugares. Para ter acesso a rede GSM é preciso que o dispositivo móvel possua um cartão SIM (*Subscriber Identity Module*) de uma operadora de

Figura 4 – Sensor de Luminosidade LDR



Fonte: [Eletrônica \(2023\)](#)

celular, isso lhe garante se conectar a rede da companhia móvel. *Global System for Mobile* (GSM) é um padrão de sistema celular de segunda geração. É o primeiro sistema celular a especificar modulação digital e arquiteturas e serviços em nível de rede ([GU; PENG, 2010](#)).

O sistema GSM opera em duas bandas pareadas de 25 MHz cada: 890–915 MHz para *uplink* ou *link* reverso onde o celular transmite para a estação base, e 935–960 MHz para *downlink* ou link direto onde a transmissão é da estação base para o celular. Essas duas bandas são divididas em canais de 200 KHz de largura ([GU; PENG, 2010](#)).

2.1.3.2 Tecnologia GPRS

O *General Packet Radio Service* (GPRS) é uma tecnologia que surgiu para complementar o GSM, o GPRS tem como finalidade fornecer *internet* móvel aos dispositivos celulares, ele tem suporte aos protocolos de rede, tais como: HTTP, TCP/IP, MQTT, etc. O GPRS foi projetado para transmitir pacotes de dados e deve obter seus recursos de rádio do conjunto de canais não utilizados pelos serviços de voz GSM ([NI; HAGGMAN, 1999](#)).

2.1.3.3 Comandos AT

Os comandos AT (AT do inglês *attention*) são usados para controlar módulos GSM/GPRS, alguns desses comandos fornecem serviços de SMS (*Short Message Service*), *internet*, etc. Os comandos AT com um MODEM GSM/GPRS ou telefone celular podem ser usados para acessar as seguintes informações e serviços ([AGNIHOTRI, 2023](#)):

1. Informações e configurações relativas ao dispositivo móvel ou modem e cartão SIM.
2. Serviços de SMS.
3. Serviços MMS.
4. Serviços de fax.

5. Link de dados e voz pela rede móvel.

A imagem a seguir apresenta alguns comandos AT.

Figura 5 – Tabela comandos AT

SMS Text mode :	
Command	Description
AT+CSMS	Select message service
AT+CPMS	Preferred message storage
AT+CMGF	Message format
AT+CSCA	Service centre address
AT+CSMP	Set text mode parameters
AT+CSDH	Show text mode parameters
AT+CSCB	Select cell broadcast message types
AT+CSAS	Save settings
AT+CRES	Restore settings
AT+CNMI	New message indications to TE
AT+CMGL	List messages
AT+CMGR	Read message
AT+CMGS	Send message
AT+CMSS	Send message from storage
AT+CMGW	Write message to memory
AT+CMGD	Delete message

Fonte: [Agnihotri \(2023\)](#)

2.1.3.4 Módulo GSM/GPRS

Acoplado a um microcontrolador, o módulo GSM/GPRS funciona como um celular, como seu próprio nome diz, ele utiliza acesso a rede GSM/GPRS e para isso aconteça, sua estrutura física possui *slot* para armazenar um cartão SIM e também tem uma antena para captar os sinais da rede de telefonia, para poder executar ações como enviar mensagem de texto, acessar a *internet* e fazer ligações, este módulo conta com suporte aos comandos AT, a seguir, é exibido um exemplo de módulo GSM/GPRS.

2.1.4 Sensor Magnético

O sensor magnético é constituído por duas placas metálicas que se movimentam conforme a ação de um campo magnético, possibilitando os aberto e fechado, quando o sensor se encontra no estado fechado as hastes estão conectadas possibilitando a passagem de corrente elétrica, quando está aberto, ou seja, as placas estão distantes uma da outra impedindo que a eletricidade passe de um lado ao outro. Existem vários modelos de sensores magnéticos, alguns são encapsulados em vidro [Figura 7](#) outros em plástico [Figura 8](#), mas ambos têm a mesma característica, um sensor que detecta se há ou não a existência de um campo magnético. Os sensores magnéticos consistem em duas placas ferromagnéticas dispostas em separação e encapsuladas por vidro. Quando um

Figura 6 – Módulo GSM/GPRS



Fonte: Saravati (2023)

Figura 7 – Sensor Magnético Reed Switch



Fonte: Macedo (2022)

Figura 8 – Sensor Magnético



Fonte: Oliveira (2021)

campo magnético (ímã) se aproxima do sensor, as suas placas alinham-se com o campo externo, estabelecendo um contacto elétrico e permitindo a passagem de corrente. Ao afastar o campo magnético, as placas voltam a se separar, interrompendo o contacto, e o sensor altera o seu estado para aberto (MARCHESAN, 2012).

2.1.5 Power Bank

O *power bank* é uma bateria portátil, ele serve para recarregar dispositivos como celulares ou tablets quando há ausência de tomadas elétricas. A fim de resolver a falta de bateria, o *power*

bank foi projetado para aparelhos portáteis para carregar a bateria quando a tomada não estiver próxima ou disponível ([SUEN; CHAN; HUNG, 2017](#)).

Figura 9 – *Power Bank*



Fonte: [amazon \(2023\)](#)

2.2 Software

Nesta seção será abordado sobre os *softwares* que serão utilizado tanto na comunicação do *hardware* como para o desenvolvimento do aplicativo móvel.

2.2.1 MQTT

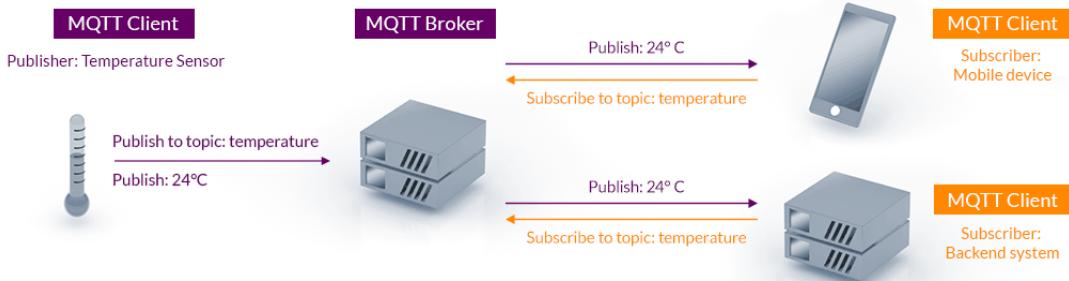
O protocolo de comunicação utilizado para trocas de mensagens entre o Arduino e dispositivo móvel é o *Message Queuing Telemetry Transport* (MQTT). Ele foi escolhido por possuir foco na IoT que funciona em cima do protocolo TCP/IP ([NERI; LOMBA; BULHÕES, 2019](#)) e tendo como particularidades sua simplicidade e baixo consumo de banda larga. Pela sua documentação, ele é resumido como um protocolo de transporte de mensagens de publicação/assinatura (*publish/subscribe*) do servidor para o cliente, sendo leve, aberto, simples e projetado para ser fácil de implementar ([OASIS, 2015](#)). "Essas características o tornam adequado para uso em muitas situações, incluindo ambientes restritos, como para comunicação em contextos de máquina para máquina (M2M) e IoT"([OASIS, 2015](#)).

2.2.1.1 Modelo *Publish/Subscribe* do MQTT

A metodologia do envio de mensagem é baseada em tópicos e assinatura (*Publish/Subscribe*), onde um cliente MQTT se inscreve em um tópico para poder receber os dados transmitidos pelo tópico. Os tópicos são hierárquicos e podem conter vários níveis separados por barras, como um caminho de arquivo conforme o exemplo: *cervejaria/galpão/tanque/temperatura*.

Um cliente pode assinar vários tópicos, como um tópico também pode ter vários inscritos. Para que essa comunicação seja realizada, há um servidor *broker* que serve como um intermediário que encaminha as informações transmitidas pelo publicador para seus assinantes. A [Figura 10](#) apresenta um modelo *Publish/Subscribe* do MQTT.

Figura 10 – Arquitetura MQTT Publish/Subscribe

Fonte: [MQTT \(2022\)](#)

2.2.1.2 Níveis de Qualidade

O MQTT possui três níveis de qualidade de serviço (QoS - *Quality of Service*): QoS 0, QoS 1 e QoS 2. Conforme explicado por [HiveMQ \(2015\)](#), os níveis de QoS funcionam do seguinte modo:

- QoS 0 : Este nível fornece entrega “no máximo uma vez”, onde as mensagens são enviadas sem confirmação e podem ser perdidas. Esse é o nível mais baixo de QoS e normalmente é usado em situações em que a perda de mensagem é aceitável ou em que a mensagem não é crítica.
- QoS 1 : Este nível fornece entrega “pelo menos uma vez”, onde as mensagens são confirmadas e reenviadas se necessário. Com QoS 1, o publicador envia a mensagem para o *broker* e aguarda a confirmação antes de prosseguir. Se o *broker* não responder dentro de um tempo definido, o publicador reenvia a mensagem.
- QoS 2 : Este nível fornece entrega “exatamente uma vez”, onde as mensagens são confirmadas e reenviadas até que sejam recebidas exatamente uma vez pelo assinante. QoS 2 é o nível mais alto de QoS e normalmente é usado em situações em que a perda ou duplicação de mensagens é inaceitável.

2.2.1.3 Persistência de Mensagem

A persistência de mensagem no protocolo MQTT se refere à capacidade de garantir que as mensagens enviadas por um cliente MQTT sejam armazenadas de forma confiável até que sejam entregues aos destinatários corretos.

Para uma comunicação confiável, o MQTT apresenta três opções de persistência de mensagem ([HIVEMQ, 2015](#)):

- **Não Persistente:** Opção padrão no MQTT. Nesse modo, as mensagens não são armazenadas no servidor e são perdidas se o servidor ou a rede falhar.
- **Persistente na fila:** neste modo, as mensagens são armazenadas no servidor até serem entregues ao assinante. Se o assinante não estiver disponível, as mensagens serão enfileiradas até que o assinante se reconecte.
- **Persistente com confirmação:** Este modo fornece o nível mais alto de persistência de mensagem. Nesse modo, as mensagens são armazenadas no servidor até serem entregues ao assinante, e o assinante deve acusar o recebimento da mensagem. Se o assinante não acusar o recebimento, a mensagem é reenviada até que o assinante confirme o recebimento.

2.2.1.4 Considerações

Por ser simples, de baixo consumo de largura de banda e oferecer opções confiáveis de entrega de dados, o protocolo MQTT é bastante utilizado no mercado IoT, fazendo com que os dispositivos que o utilizam possam enviar maior quantidade de informações em relação a outros protocolos de comunicação, sendo esta também uma razão pela qual o MQTT foi escolhido para este projeto.

2.2.2 React Native

O React Native³ é uma *framework* para criação de aplicativos móveis, inicialmente ela foi desenvolvida pela Meta (Facebook) atualmente é de código aberto onde quem desejar pode contribuir com melhorias, sua principal característica é a criação de aplicativos multiplataforma, ou seja, basta desenvolver um único app e ele funcionará em plataformas diferentes como Android e iOS.

O React Native é capaz de renderizar os componentes do React Native em visualizações nativas reais para Android ou visualizações de interface do usuário para iOS. Isso é possível devido à camada de abstração conhecida como "bridge" que permite ao React Native invocar as APIs de renderização em Java para Android ou Objective-C para iOS. Além disso, o *framework* revela a interface do JavaScript, permitindo que o aplicativo acesse recursos específicos da plataforma como a bateria ou a localização (EISENMAN, 2015 apud DANIELSSON, 2016).

2.2.3 NodeJS

Node.js⁴ é uma tecnologia que executa código JavaScript, ela é autônoma não depende de um navegador para rodar um algoritmo, uma de suas características é sua execução ser *single-thread*, ou seja, necessita de apenas uma *thread* para executar código JavaScript, fazendo

³ <https://reactnative.dev/>

⁴ <https://nodejs.org/en>

com que o servidor não utilize muitos recursos e melhore sua performance. Node.js escrito em linguagem C++, é um ambiente operacional JavaScript. Node.js é um ambiente de tempo de execução JavaScript. O Node.js usa o mecanismo Google Chrome V8 para obter um bom desempenho e também fornece várias APIs no nível do sistema, como operações de arquivo, programação da Web e assim por diante. O código JavaScript no lado do navegador está sujeito a várias restrições de segurança em tempo de execução e a operação do sistema cliente é limitada. O Node.js usa programação assíncrona orientada a eventos e projetada para serviços de rede (LIANG; ZHU; SHANG DONGYU FENG, 2017).

2.2.4 Expo

O Expo⁵ é uma plataforma de desenvolvimento de aplicativos móveis e web baseada no React Native. Com ele, é possível criar aplicativos usando JavaScript e React Native. O Expo oferece APIs pré-construídas que simplificam o acesso a recursos nativos dos dispositivos, como câmera, galeria, localização, entre outros. Uma das vantagens do Expo é a capacidade de emular o funcionamento do aplicativo em diversos sistemas, incluindo iOS e Android. Isso significa que um desenvolvedor pode criar um aplicativo iOS em um computador com Windows ou Linux.

Expo é uma ferramenta de linha de comando que encapsula projetos React Native em "projetos Expo". Esses projetos Expo organizam todos os recursos do *framework* de forma mais estruturada. Além disso, o Expo oferece aos desenvolvedores recursos de depuração da aplicação e a capacidade de visualizar o aplicativo em tempo real. Isso é viabilizado pelo Expo Client, um aplicativo disponível para Android e iOS que utiliza uma conexão Wi-Fi para estabelecer uma ligação direta entre o smartphone e o ambiente de desenvolvimento. (ARAUJO, 2019).

2.2.5 Google Firebase

O Google Firebase⁶ é uma plataforma que dispõe de várias ferramentas em nuvem que tem como objetivo auxiliar no desenvolvimento de aplicativos móveis e web. O Google Firebase é um software de desenvolvimento de aplicativos apoiado pelo Google que permite aos desenvolvedores desenvolver aplicativos IOS, Android e Web. O Firebase fornece ferramentas para rastrear análises, gerar relatórios e corrigir falhas de aplicativos, criar experimentos de marketing e produtos (CHOUGALE; YADAV; GAIKWAD, 2022).

Alguns dos serviços fornecidos pelo Google Firebase são:

- *Analytics*
- *Authentication*
- *Cloud Messaging*

⁵ <https://expo.dev/>

⁶ <https://firebase.google.com/?hl=pt-br>

- ***Realtime Database***
- ***Crashlytics***
- ***Performance***

O aplicativo móvel deste projeto fará uso de dois recursos do Firebase:

- ***Authentication*** - Para autenticação e recuperação de senhas;
- ***Realtime Database*** - Armazenamento de e-mails e data de recarga.

3

Proposta do Sistema de Alarme

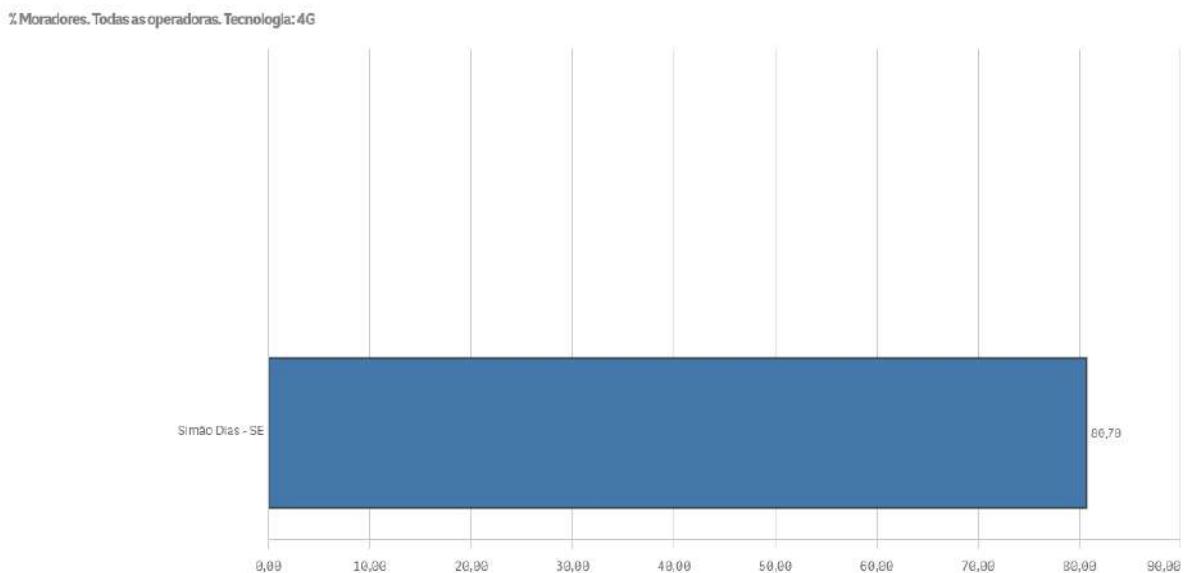
Conforme mencionado anteriormente, o sistema de alarme proposto é baseado em um caso real, tendo sua concepção originada devido a uma série de furtos ocorridos em um apiário no município de Simão Dias. Ele é composto por duas partes distintas: um dispositivo eletrônico e um aplicativo para celular. Através da conectividade pela internet, essas duas partes podem se comunicar e informar quando o perímetro é violado.

O componente eletrônico principal é o microcontrolador ESP32, responsável por conectar os diversos módulos e executar o algoritmo que gerencia e controla os outros componentes. O sensor magnético desempenha o papel de enviar sinais ao Arduino quando suas travas são desengatadas. O sensor de luminosidade atende a uma regra de negócio que determina que o sinal de alerta só pode ser ativado durante o período noturno, essa regra foi estabelecida com base no fato de que, durante o período diurno, há movimentação nas proximidades do apiário, as abelhas tendem a ser mais agressivas, e o estresse de um enxame pode impactar outras colmeias. Esses fatores tornam a ação de furto mais difícil durante o dia. O módulo GSM tem como função estabelecer conexão com a rede 4G e transmitir as informações para um servidor. O *powerbank* fornece energia elétrica para os demais componentes eletrônicos, enquanto o regulador de tensão gerencia a voltagem e a corrente elétrica direcionadas ao módulo GSM. A [Figura 24](#) no [Capítulo 6](#) apresenta o diagrama do circuito eletrônico. Essa solução integra tecnologia e segurança para lidar com o problema de segurança em apiários, permitindo uma resposta eficiente e oportunamente em caso de violação do perímetro.

Devido à disponibilidade de sinal de operadoras de celular na região, este projeto apresenta viabilidade para implantação. A conectividade desempenha um papel crucial, pois permite a transmissão das informações necessárias. A figura [Figura 11](#) exibe dados de março de 2023 da Anatel, mostrando a porcentagem de cobertura da tecnologia 4G em Simão Dias, separando por operadoras, temos Claro com 66,06%, OI 0%, Tim 62,80% e VIVO com 78,46% Fonte: ([ANATEL, 2023c](#)). Conforme demonstrado, a rede 4G abrange a maior parte do município. Isso

foi um dos critérios na escolha desta tecnologia como meio de conectividade com a *internet*. Outros motivos para sua escolha incluem o fato de que as redes 2G e 3G estão programadas para serem desativadas em um futuro próximo, o que tornaria o projeto obsoleto. Além disso, a tecnologia 5G, que ainda está em processo de expansão, não está disponível na cidade, como ilustrado na [Figura 12](#).

Figura 11 – Cobertura 4G



Fonte: [Anatel \(2023c\)](#)

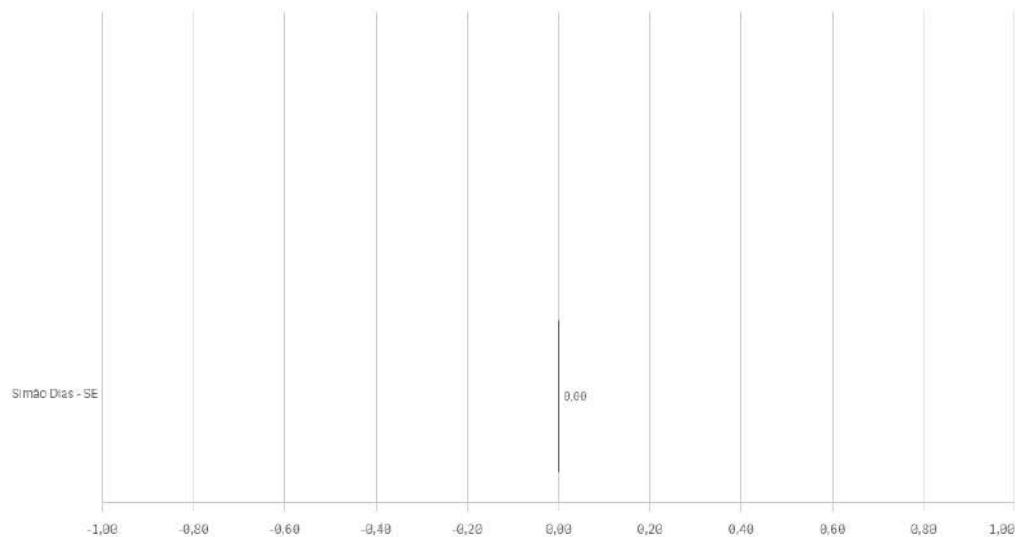
A forma como o apiário foi construído também contribui para a eficiência do alarme, pois, existe apenas um caminho para acessá-lo, o que induz os invasores utilizarem a porteira, consequentemente desconectando as partes do sensor magnético. As figuras [Figura 13](#), [Figura 14](#), [Figura 15](#), [Figura 16](#), [Figura 17](#) e [Figura 18](#) dão uma visão de como é o apiário.

O aplicativo móvel será um *software* simples, sua principal funcionalidade é emitir sinais sonoros quando receber informações do servidor alertando sobre uma possível existência de invasão, outras funcionalidades serão implementadas ao longo do projeto caso seja necessário.

A [Figura 19](#) apresenta uma visão geral de como deve ser o sistema e sua comunicação bidirecional.

Figura 12 – Cobertura 5G

% Moradores. Todas as operadoras. Tecnologia: 5G



Fonte: Anatel (2023c)

Figura 13 – Porteira do Apiário



Fonte: Autor

Figura 15 – Lateral Direito



Fonte: Autor

Figura 14 – Entrada do Apiário



Fonte: Autor

Figura 16 – Fundo



Fonte: Autor

Figura 17 – Lateral Esquerdo



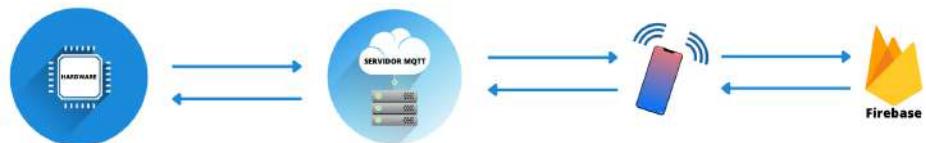
Fonte: Autor

Figura 18 – Frente



Fonte: Autor

Figura 19 – Visão Geral do Sistema



Adaptado de: [IO-Imagens \(2023\)](#), [pixabay \(2023\)](#), [Tavares \(2023\)](#), [Inaara \(2023\)](#)

4

Alicerces para o Desenvolvimento do Sistema

4.1 Requisitos Funcionais

Os requisitos funcionais na engenharia de *software* representam funções que um sistema deve exercer para atingir os objetivos esperados. De acordo com Sommerville (2007), os requisitos funcionais são as declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.

O [Quadro 1](#) apresenta os requisitos funcionais do dispositivo eletrônico do sistema, enquanto que o [Quadro 2](#) exibe os requisitos do aplicativo móvel.

Quadro 1 – Requisitos Funcionais *Hardware*

Código	Identificação	Classificação	Objetivo
RF01	Detectar abertura	Essencial	O sistema deve identificar quando a portaria é aberta.
RF02	Detectar luminosidade	Essencial	O sistema deve detectar o nível de luminosidade para poder disparar o alarme.
RF03	Emitir sinal	Essencial	O sistema deve ser capaz de enviar sinal de alerta quando o perimetro for violado.

Fonte: Autor

Quadro 2 – Requisitos Funcionais Aplicativo Móvel

Código	Identificação	Classificação	Objetivo
RF01	Receber sinal	Essencial	O sistema deve receber sinal de alerta enviado pelo <i>hardware</i> .
RF02	Emitir sinal	Essencial	O sistema deve emitir sinais sonoros para informar que o perímetro foi violado.
RF03	Desligar alarme	Essencial	O sistema deve possibilitar o desligamento da notificação sonora.
RF04	Cadastrar, alterar data	Essencial	O sistema deve possibilitar o cadastro e alteração da data da recarga.
RF05	Emitir lembrete	Essencial	O sistema deve emitir notificações para lembrar o usuário de colocar crédito no chip.
RF06	Verificar data	Essencial	O sistema deve possibilitar a verificação da data em que foi feita a recarga do chip.
RF07	Notificar perda de comunicação	Essencial	O sistema deve notificar quando há perda de comunicação com o <i>hardware</i> .
RF08	Verificar conectividade	Essencial	O sistema deve permitir a verificação da conectividade do <i>hardware</i> .
RF09	Fazer login	Essencial	O sistema deve solicitar autenticação do usuário.
RF10	Realizar Cadastro	Essencial	O sistema deve fornecer opção de cadastro.

Fonte: Autor

4.1.1 *Hardware*

4.1.2 Aplicativo Móvel

4.2 Requisitos Não Funcionais

Os requisitos não funcionais descrevem características e qualidade do sistema, alguns exemplos seriam: disponibilidade do sistema, usabilidade, processamento, etc.

Os requisitos não funcionais são restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo ([SOMMERVILLE, 2007](#)).

Os quadros [Quadro 3](#) e [Quadro 4](#) apresentam respectivamente os requisitos não funcionais do *hardware* e do aplicativo de celular.

Quadro 3 – Requisitos Não Funcionais *Hardware*

Código	Identificação	Classificação	Objetivo
RNF01	Baixo consumo	Essencial	O <i>hardware</i> do sistema deve possuir baixo consumo de energia.
RNF02	Fonte ajustável	Essencial	O <i>hardware</i> do sistema deve ter uma fonte de alimentação energética ajustável.
RNF03	Acesso a rede móvel	Essencial	O <i>hardware</i> do sistema deve poder se conectar a rede de telefonia móvel.
RNF04	Integração com servidor MQTT	Essencial	O <i>hardware</i> do sistema deve poder se conectar ao servidor MQTT.

Fonte: Autor

Quadro 4 – Requisitos Não Funcionais Aplicativo Móvel

Código	Identificação	Classificação	Objetivo
RNF01	Interface simples e intuitiva	Essencial	O sistema deve possuir uma interface simples e intuitiva para que o usuário possa usá-lo sem complicações.
RNF02	Responsivo	Essencial	O sistema deve ser responsivo e se adaptar aos diferentes tamanhos de telas.
RNF03	Integração com servidor MQTT	Essencial	O sistema deve ser capaz de se comunicar com o servidor MQTT.
RNF04	Compatibilidade de SOs	Essencial	O sistema deve ser compatível a diferentes tipos de sistemas operacionais (Android e IOs).

Fonte: Autor

4.2.1 Hardware

4.2.2 Aplicativo Móvel

4.3 Regras de Negócio

Regras de negócio são especificações que definem e delimitam as operações do sistema ou projeto.

uma regra de negócio pode ser definida segundo duas perspectivas: a perspectiva do negócio e a perspectiva dos sistemas de informação (SI). Segundo a perspectiva do negócio, uma regra de negócio é uma diretiva destinada a influenciar ou guiar o comportamento do negócio, como suporte à política de negócio que é formulada em resposta a uma oportunidade. Segundo a perspectiva dos sistemas de informação, uma regra de negócio é uma sentença que define ou restringe algum aspecto do negócio. Pretende-se garantir a estrutura do negócio ou controlar a influência do comportamento do mesmo (PROJECT, 1997 apud SILVA et al., 2001).

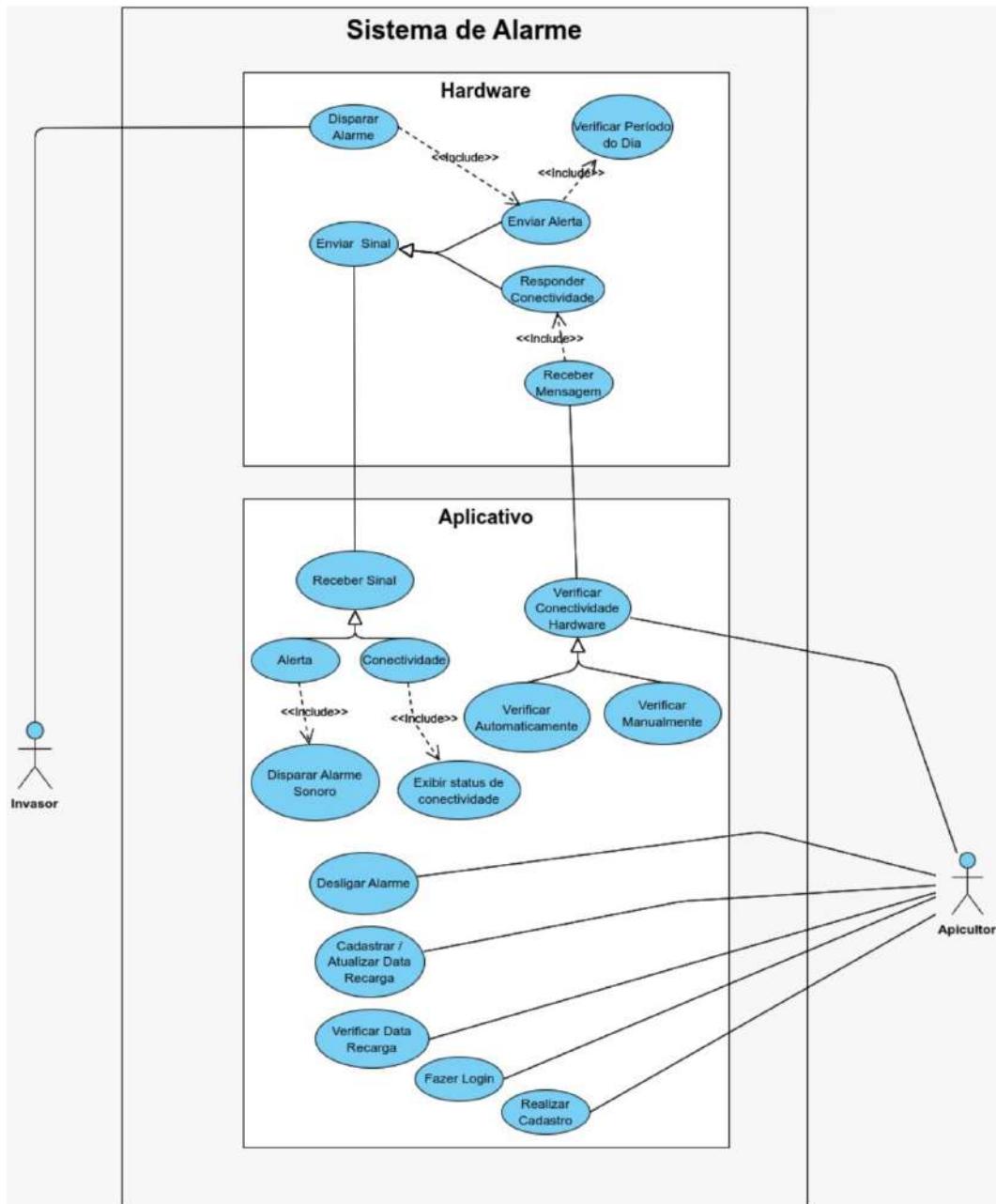
Abaixo, seguem as regras de negócio do sistema:

- O sinal de alerta só pode ser emitido durante o período noturno;
- A cada intervalo de tempo determinado, o aplicativo deve mandar mensagem para o hardware, caso não obtenha resposta, repete o procedimento em um intervalo mais curto, se a falta de comunicação persistir, emitir notificação;

4.4 Diagrama de Cado de Uso

O diagrama de caso de uso é uma representação gráfica que modela um sistema, ele é implementado usando a *Unified Modeling Language* (UML). O diagrama de caso de uso é criado para demonstrar uma possível interação entre o usuário e o sistema. De acordo com Pender (2004 apud FILHO, 2011), o diagrama de caso de uso (Use Case) é um elemento gráfico exclusivo, pois é um diagrama usado para modelar o modo como as pessoas esperam usar um sistema. O diagrama descreve quem serão os usuários relevantes, os serviços que eles exigem do sistema e os serviços que eles precisam oferecer ao sistema. A Figura 20 apresenta o diagrama de caso de uso do sistema de alarme.

Figura 20 – Diagrama de Caso de Uso do Sistema de Alarme



Fonte: Autor

5

Busca de Anterioridade

A busca de anterioridade ou estado da arte tem como objetivo realizar pesquisas para verificar se uma determinada tecnologia já existe, caso seja algo inovador, seu processo de patenteamento é iniciado. Segundo [Dias e Porto \(2013\)](#), busca de anterioridade - busca nas bases de patentes nacionais e internacionais "sobre eventuais tecnologias semelhantes ou que buscam solucionar o mesmo tipo de problema, bem como a identificação do diferencial tecnológico da criação em relação às soluções existentes".

Para o sistema de alarme, foram realizadas consultas em sites como ieee, acm, inep e google acadêmico, algumas palavras chaves utilizadas nas buscas e suas respectivas tradução para inglês foram: sistema de alarme IoT, sistema de alarme com ESP32 e GSM, sistema de alarme para zona rural, IoT alarm system, Alarm system with ESP32 and GSM, rural area alarm system. Alguns dos resultados encontrados são apresentados a seguir.

5.1 Resultados

5.1.1 Prototipagem de Sistema de Alarme de Incêndio Baseado em Sensor Nanoestruturado Integrado a IoT

[Freitas \(2021\)](#) apresenta um protótipo de sensor de incêndio com maior precisão com monitoramento em tempo real, esse modelo é desenvolvido utilizando o conceito de IoT, seu hardware é composto por um NodeMCU, um sensor nanoestruturado, sua comunicação acontece através do protocolo MQTT e os sinais de alerta são recebidos por um aplicativo Android.

Análise: ambos sistemas utilizam o mesmo protocolo de comunicação e os dados são recebidos por um aplicativo móvel, mas seus objetivos de monitoramento são diferentes, eles também se diferem no fato que um é para ambientes externo e outro interno, para se conectar com a *internet*, o protótipo do [Freitas \(2021\)](#) utiliza a rede wifi disponível no ambiente enquanto

que o BeeNotify faz essa conexão através da rede GSM.

5.1.2 Fully Automatic Infrared Pyroelectric Home Burglar Alarm Design

[Bao et al. \(2024\)](#) apresenta um sistema de alarme residencial que utiliza o microcontrolador STC89C52 como componente central para gerenciar o hardware. Um dos principais dispositivos integrados é o sensor infravermelho humano DYP-ME003, responsável por detectar movimentos no ambiente e acionar o alarme quando necessário.

Análise: Ambos os sistemas têm o mesmo objetivo, que é servir como sistemas de alarme. No entanto, eles se destinam a ambientes diferentes e utilizam hardwares compostos por dispositivos eletrônicos distintos.

5.1.3 ESP32 CAM-based Car Security System via Telegram Integration

[Monica, Kumar e Vemulapalli \(2023\)](#) apresentam um sistema de alarme automotivo que utiliza o ESP32 integrado a uma câmera e um módulo GPS, acoplado ao volante do veículo. O sistema é capaz de enviar fotos e a localização do veículo em tempo real para um bot no Telegram. Com essas informações, o proprietário pode decidir remotamente, por meio de comandos enviados pelo Telegram, se concede ou não acesso ao veículo.

Análise: Este projeto compartilha com o BeeNotify apenas o uso do microcontrolador ESP32. No entanto, as formas de comunicação com o sistema e os objetivos de proteção diferem significativamente entre os dois projetos.

5.1.4 Sistema de geolocalización con alarma y monitoreo basado en IOT para personas con Alzheimer

[Chavez e Choque \(2022\)](#) apresentam um projeto de geolocalização projetado para monitorar pessoas com Alzheimer e seus cuidadores. O sistema utiliza uma placa TTGO T-Call ESP32 equipada com o módulo GSM SIM800L, responsável por enviar dados de localização para o banco de dados do Google Firebase. A comunicação entre o ESP32 e o módulo GSM é realizada por meio de comandos AT. Posteriormente, as informações armazenadas no Firebase são consumidas por um aplicativo móvel, que, com o auxílio da API do Google Maps, gerencia os dados e fornece a localização em tempo real do indivíduo.

Análise: Esse projeto utiliza uma placa de desenvolvimento do mesmo fabricante da placa empregada no BeeNotify. Contudo, o sistema apresentado pelo autor faz uso de um módulo GSM compatível com a tecnologia 2G, que está em vias de descontinuação. Em contrapartida, a placa utilizada no BeeNotify adota o módulo GSM A7670G, que oferece suporte à tecnologia 4G, considerada uma das mais modernas e amplamente utilizadas atualmente. Além disso, os dois projetos possuem finalidades distintas no que diz respeito ao monitoramento.

5.2 Conclusão

Os projetos e protótipos encontrados na busca de anterioridade, têm como objetivo em comum o uso do conceito e ferramentas da internet das coisas, todos fazem uso de um microcontrolador para gerenciar os outros componentes eletrônicos, mas sempre se diferem em algo comparando-se com o BeeNotify, o trabalho que possui uma similaridade maior é o da [subseção 5.1.4](#), pois, seu hardware é composto por um ESP32, modem GSM, faz uso dos comandos AT, da plataforma do Google Firebase e de um aplicativo móvel, mas ele não faz uso de servidor MQTT e seu objetivo de monitoramento é diferente, deste modo, o BeeNotify apresenta indícios de ser um projeto inovador como um sistema de alarme de segurança para o mercado agropecuário.

6

Solução Proposta

Neste capítulo será apresentado o protótipo das telas do aplicativo móvel e o diagrama da montagem do circuito.

6.1 Protótipo de Telas do Aplicativo

A [Figura 21](#) ilustra a tela inicial do aplicativo, na qual são exibidas informações relevantes. Descrevendo de cima para baixo, a primeira seção apresenta o nome do apicultor, seguido pelo nome do apiário. Em sequência, encontramos informações sobre a conectividade do hardware, permitindo ao usuário verificar se o equipamento eletrônico está conectado à rede de internet móvel. O status é atualizado automaticamente, entretanto, o usuário também pode optar por verificar a conexão no momento atual clicando no botão correspondente. Por último, a data da última recarga do chip da operadora de telefonia é exibida, sendo possível atualizá-la mediante um clique no botão apropriado.

Na [Figura 22](#), é exibida como ficaria a tela ao clicar no botão de atualizar a data. Por fim, a [Figura 23](#) mostra a tela de disparo do alarme quando o perímetro monitorado for violado. Nesta tela será emitido um sinal sonoro e a disponibilização de dois botões onde o primeiro desativa o alarme, já o segundo é um atalho que permite a vítima ligar para as autoridades locais.

6.2 Diagrama do Hardware

No desenvolvimento de hardware deste projeto, será utilizada a placa LILYGO¹ modelo T-A7670G². Com base em sua arquitetura, foi elaborado o diagrama de hardware do projeto

¹ <https://www.lilygo.cc/>

² <https://www.lilygo.cc/products/t-sim-a7670e?srsltid=AfmBOoooNkUWyw4pEpGZOcf2Gv2AsoxK74UTB2WYxluWnyLL3O>

Figura 21 – Tela Inicial



Adaptado de: [iftikharalam \(2021\)](#)

que é apresentado na [Figura 24](#). Nele é possível observar as conexões entre os componentes eletrônicos.

Para informações técnicas mais detalhadas sobre o sobre a placa T-A7670G e sobre o modem GSM A7670G, consulte o GitHub do fabricante³.

³ <https://github.com/Xinyuan-LilyGO/LilyGO-T-A76XX?tab=readme-ov-file>

Figura 22 – Tela Inicial ao clicar no botão para atualizar data

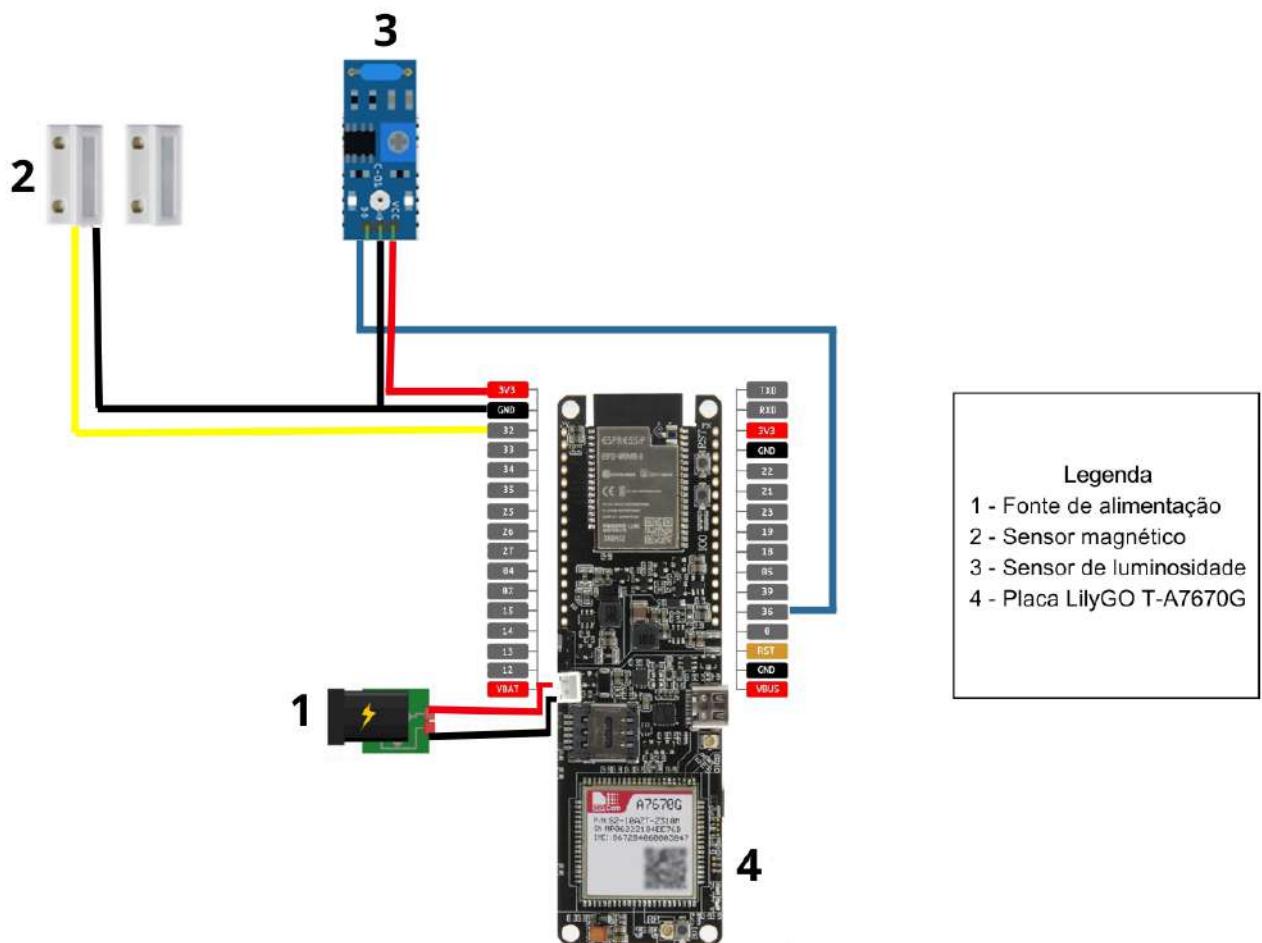


Adaptado de: [iftikharalam \(2021\)](#)

Figura 23 – Tela de Alarme



Adaptado de: [iftikharalam \(2021\)](#)

Figura 24 – Diagrama do *Hardware*

Adaptado de: [LILYGO \(2024\)](#), [Fritzing \(2023\)](#), [Components101 \(2023\)](#), [UsinaInfo \(2023\)](#)

7

Desenvolvimento e Resultados

Neste capítulo, são abordados os métodos utilizados para o desenvolvimento do projeto tanto a parte da aplicação móvel como o hardware.

7.1 Aplicativo

A [seção 2.2](#) apresenta as tecnologias necessárias para a implementação do aplicativo. Fazendo uso desses recursos tecnológicos, são apresentados alguns pontos chaves do código da aplicação.

7.1.1 Autenticação e Banco de Dados

Para agilizar o desenvolvimento de autenticação e armazenamento de dados, foi utilizado o Google Firebase ([subseção 2.2.5](#)), ele fornece uma estrutura completa para o serviço de cadastro, login e recuperação de senha do usuário. O [Código 1](#) demonstra como configurar uma seção no Firebase.

O [Código 2](#) apresenta de forma simples como armazenar e criar o login do usuário. Nessa função, além do email e senha utilizados no cadastro do login, são passados outros parâmetros para serem armazenados no banco de dados (linha 2) que ficarão disponíveis para consultas posteriores. Por possuir poucas informações cadastrais, o banco NoSQL do Firebase é o suficiente para fazer os tratamentos e consultas necessárias.

Para acessar o código completo e ver outras funcionalidades como recuperar senha, fazer login, buscar e atualizar informações cadastrais, acesse no GitHub o arquivo [firebaseDB.js](#).

Código 1 – Conexão com Google Firebase

```

1 const firebaseConfig = {
2   apiKey: API_KEY,
3   authDomain: AUTH_DOMAIN,
4   projectId: PROJECT_ID,
5   storageBucket: STORAGE_BUCKET,
6   messagingSenderId: MESSAGING_SENDER_ID,
7   appId: APP_ID,
8   measurementId: MEASUREMENT_ID
9 };
10
11 const initializeDB = initializeApp(firebaseConfig);
12 const db = getFirestore(initializeDB);
13 const auth = getAuth(initializeDB);

```

Código 2 – Registro do Usuário

```

1 async function signupApp (userName, date, userEmail, password) {
2   await addData(userName, userEmail, date);
3
4   console.log(password)
5
6   var message = '';
7
8   try {
9     await createUserWithEmailAndPassword(auth, userEmail,
10       password);
11
12     message = 'Usuário cadastrado com sucesso'
13     alert('Usuário cadastrado com sucesso')
14   }
15   catch (error) {
16     message = 'Falha ao cadastra usuário';
17     alert('Falha ao cadastra usuário', error)
18   }
19
20   return (message)
21 }

```

7.1.2 Comunicação com Servidor MQTT Broker

Para comunicação via protocolo MQTT ([subseção 2.2.1](#)), o projeto faz uso da versão de testes do servidor Mosquitto.org¹, os recursos por ele fornecidos são suficientes para a realização dos testes do sistema. Os códigos apresentados a seguir, demonstram como é realizada a conexão com o servidor, inscrição e publicação em tópicos.

¹ <https://test.mosquitto.org/>

Código 3 – Conexão com Servidor MQTT Broker

```
1 const client = mqtt.connect('ws://test.mosquitto.org:8080/mqtt');
```

O [Código 4](#) demonstra como é feita a inscrição e o recebimento de mensagem em um tópico.

Código 4 – Inscrição em Tópico e Recebimento de Mensagem

```
1 client.on('connect', () => {
2   client.subscribe('Hardware/Status');
3   client.subscribe('Hardware/Alert');
4
5
6   client.on('message', (topic, message) => {
7     `${message}`);
8     if(messageCallback) {
9       messageCallback(message.toString());
10    }
11  });
12});
```

Para publicação de mensagens, é utilizado o [Código 5](#), o parâmetro “message” representa o conteúdo da mensagem que será enviada.

Código 5 – Publicação em Tópico

```
1 const publishMessage = (message) => {
2   client.publish('App/Status', message);
```

Para consultar o código completo, acesse o arquivo [serverMQTT.js](#).

7.1.3 Tela Principal

Na tela da [Figura 21](#) encontra-se a maioria das funcionalidades do aplicativo, através da sua interface é possível verificar o status de conectividade do *Hardware* com a rede móvel, atualizar data da recarga dos créditos da companhia telefônica, no seu *backend*, há funcionalidades que rodam em segundo plano, algumas delas são:

- Verificar conectividade do hardware automaticamente a cada um minuto ([Código 6](#));
- Verificar a cada segundo se o hardware enviou um sinal de alerta ([Código 7](#));

As demais funcionalidades do algoritmo, como a atualização da data e do status da conexão, estão disponíveis no repositório². Código completo está disponível em [alarmScreen.js](#).

² <https://github.com/Gideval/BeeNotify/blob/main/AppBeeNotify/src/screens/mainScreen.js>

Código 6 – Verifica Conexão

```

1  useEffect(() => {
2      const interval = setInterval(async () => {
3          publishMessage('Status');
4
5          setMessageCallBack(message => {
6              setMessageMQTT(message);
7          });
8
9          if(cont === 3) {
10              setCont(0);
11              alert('Dispositivo Desconectado!');
12          }
13          else if(messageMQTT !== 'Alert' && messageMQTT !==
14              'Connected') {
15              setCont(prevCont => prevCont + 1);
16          }
17          else if(messageMQTT === 'Connected') {
18              setCont(0);
19          }
20      }, 1 * 60 * 1000);
21
22      return () => {
23          clearInterval(interval);
24      };
25  }, [connectionMqtt]);

```

Código 7 – Busca Mensagem de Alerta

```

1  useEffect(() =>{
2      const intervalId = setInterval(() => {
3          setMessageCallBack(message => {
4              if (message === 'Alert') {
5                  callAlarmScreen();
6              }
7          });
8      }, 1000);
9
10     return () => {
11         clearInterval(intervalId);
12     };
13 }, []);

```

7.1.4 Tela de Alarme

A tela da Figura 23 possui poucas funcionalidades, sua função principal é emitir um sinal sonoro para informar que o perímetro foi violado, ela também disponibiliza um atalho para fazer ligações emergenciais (linhas 14 à 17 do Código 8).

Código 8 – Carrega Arquivo de Áudio

```

1  async function playSound() {
2      const { sound } = await
3          Audio.Sound.createAsync(require('../../../../assets/Alarm.mp3'));
4          setSound(sound);
5
6      await sound.playAsync();
7
8  async function pauseSound() {
9      if(sound) {
10         await sound.pauseAsync();
11     }
12 }
13
14 const phoneCall = () => {
15     let phoneNumber = '190';
16     Linking.openURL('tel:${phoneNumber}');
17 }
```

As demais funcionalidades (login, recuperação de senha, cadastro, etc) estão disponível no [repositório no GitHub](#).

7.2 Hardware

Esta seção apresenta o código desenvolvido para executar as instruções no microcontrolador, além da configuração da placa de circuito para a integração dos componentes com o ESP32. Por fim, será discutido o resultado da comunicação entre o aplicativo móvel e o dispositivo físico.

7.2.1 Código

Na programação do microcontrolador, foi utilizada a linguagem C/C++. O código inclui funções para ligar e reiniciar o módulo GSM, verificar a desconexão do sensor magnético, enviar e receber mensagens, monitorar a conexão móvel, entre outras funcionalidades.

O trecho do [Código 9](#) define os pinos dos sensores (magnético e luminosidade).

Código 9 – Define Constantes dos Pinos

```

1  #define SENSOR_PIN 32
2  #define LUZ_PIN 36
```

A inicialização dos pinos dos sensores e do modem GSM é mostrada no [Código 10](#), por exemplo, a variável SENSOR_PIN é responsável por receber os sinais enviados pelo sensor

magnético, que está conectado ao pino número 32. [Código 11](#), por meio de comandos AT, é estabelecida a conexão com a rede móvel e com o servidor *MQTT broker*, além de permitir que o cliente se inscreva em um tópico.

Código 10 – Inicialização Sensores e Modem GSM

```

1 void setup() {
2     SerialMon.begin(115200);
3     delay(10);
4
5     pinMode(SENSOR_PIN, INPUT_PULLUP);
6     pinMode(LED_PIN, OUTPUT);
7
8     analogReadResolution(12);
9     analogSetAttenuation(ADC_11db);
10
11    modemPowerOn();
12    SerialAT.begin(UART_BAUD, SERIAL_8N1, PIN_RX,
13                  PIN_TX); //Inicia SimA7670G
14
15    SerialMon.println("Initializing modem... ");
16    modem.restart();
17
18    // ...
}
```

No [Código 12](#), o cliente MQTT aguarda mensagens enviadas ao tópico "App/Status" em que está inscrito. [Código 13](#), em intervalos de tempo predefinidos, essa função verifica a conexão com o servidor broker. Caso a conexão tenha caído, a função tenta realizar a reconexão e reinscrever o cliente no tópico.

Através da função do [Código 14](#) são enviados comandos AT que serão interpretados e executados pelo módulo GSM. Para saber mais sobre os comandos AT e suas funcionalidades, recomenda a leitura do site da fabricante SIMCOM³.

O código completo do microcontrolador encontra-se no GitHub em [main.cpp](#).

7.2.2 Integração de Componentes Eletrônicos

Conforme discutido na [seção 6.2](#), o projeto utiliza a placa T-A7670G, que facilita a integração entre o microcontrolador ESP32 e o módulo GSM, já que ambos estão incorporados na mesma placa. Essa integração reduz possíveis problemas de comunicação e alimentação entre as partes.

³ <https://www.simcom.com/product/A7670X.html>

Código 11 – Conexão com Servidor MQTT

```

1 // Comandos iniciais enviado para o modem
2 SerialAT.println("AT"); // verifica se o modem esta respondendo
3 delay(1000);
4 SerialAT.println("AT+CSQ"); // Verifica sinal rede movel
5 delay(1000);
6 SerialAT.println("AT+NETOPEN"); // conecta com a internet
7 delay(1000);
8
9 // Inicia configuracao para trocas de mensagens via protocolo MQTT
10 delay(5000);
11
12 // Inicia serviço MQTT
13 if (!sendATCommand("AT+CMQTTSTART", "+CMQTTSTART: 0")) {
14     Serial.println("Failed to start MQTT service");
15     return;
16 }
17 delay(3000);
18
19 // Configura Cliente MQTT
20 if (!sendATCommand("AT+CMQTTACQ=0,\"client_Val1\"")) {
21     Serial.println("Failed to acquire MQTT client");
22     return;
23 }
24 delay(3000);
25
26 // Se conecta ao servidor broker
27 if (!sendATCommand("AT+CMQTTCONNECT=0,\"tcp://test.mosquitto.org:1883\",60,1",
28     "+CMQTTCONNECT: 0,0", 20000)) {
29     Serial.println("Failed to connect to MQTT broker");
30     return;
31 }
32 delay(3000);
33
34 // Se inscreve em um topico
35 sendATCommand("AT+CMQTTSUB=0,9,1", "+CMQTTSUB: 0,0", 5000);
36 delay(2000);
37 sendATCommand("App/Status", "+CMQTTSUB: 0,0", 10000);
38 delay(5000);

```

Código 12 – Aguarda Mensagem

```

1 static String response = "";
2 while (SerialAT.available()) {
3     SerialMon.println("Entrou no serialAT");
4     SerialMon.println("Response: " + response);
5     char c = SerialAT.read();
6     response += c;
7     if (response.indexOf("Status") != -1) {
8         processResponse();
9         response = "";
10    }
11 }

```

A Figura 25 mostra a placa de circuito onde serão conectados os sensores. Essa placa foi projetada para simplificar a manutenção e a substituição dos componentes, que serão apenas encaixados e não soldados.

Código 13 – verifica Conexão

```

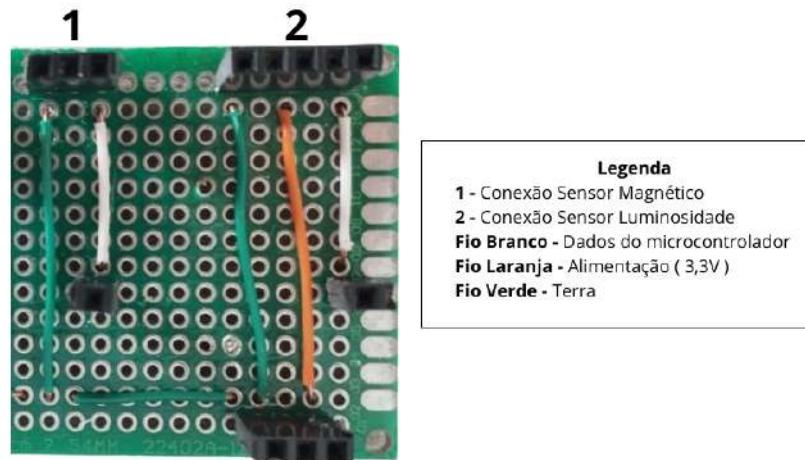
1 if (currentMillis - previousMillis >= interval) {
2     previousMillis = currentMillis;
3
4     if (!checkMQTTConnection()) {
5         Serial.println("Try Reconnected MQTT broker");
6
7         sendATCommand("AT+CMQTTCONNECT=0,\"tcp://test.mosquitto.org:1883\",60,1",
8                     "+CMQTTCONNECT: 0,0", 20000);
9
10    // delay(3000);
11    // Se inscreve em um topico
12    sendATCommand("AT+CMQTTSUB=0,9,1", "+CMQTTSUB: 0,0", 5000);
13    // delay(500);
14    sendATCommand("App/Status", "+CMQTTSUB: 0,0", 10000);
15    // delay(2000);
16 }
```

Código 14 – Envia Comandos AT

```

1 bool sendATCommand(const char* command, const char*
2 expectedResponse, unsigned long timeout) {
3     Serial.print("Sending: ");
4     Serial.println(command);
5
6     SerialAT.write(command, strlen(command));
7     SerialAT.write("\r\n");
8
9     unsigned long startTime = millis();
10    String response = "";
11    bool success = false;
12
13    while (millis() - startTime < timeout) {
14        if (SerialAT.available()) {
15            char c = SerialAT.read();
16            response += c;
17            Serial.print(c);
18            if (response.endsWith("\r\n")) {
19                if (response.indexOf("ERROR") != -1) {
20                    success = false;
21                    break;
22                }
23                if (expectedResponse == NULL ||
24                    response.indexOf(expectedResponse) != -1) {
25                    success = true;
26                    break;
27                }
28                response = "";
29            }
30        }
31    }
32 }
```

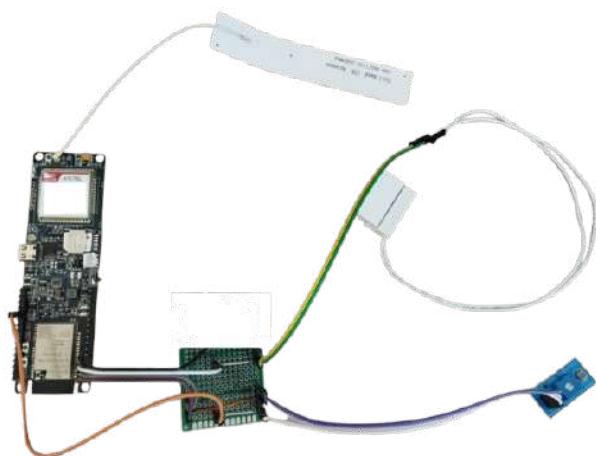
Figura 25 – Placa do Circuito



Fonte: Autor

A conexão entre a placa de desenvolvimento T-A7670G e os sensores através da placa de circuito é ilustrada na [Figura 26](#).

Figura 26 – Conexão Sensores com T-A7670G



Fonte: Autor

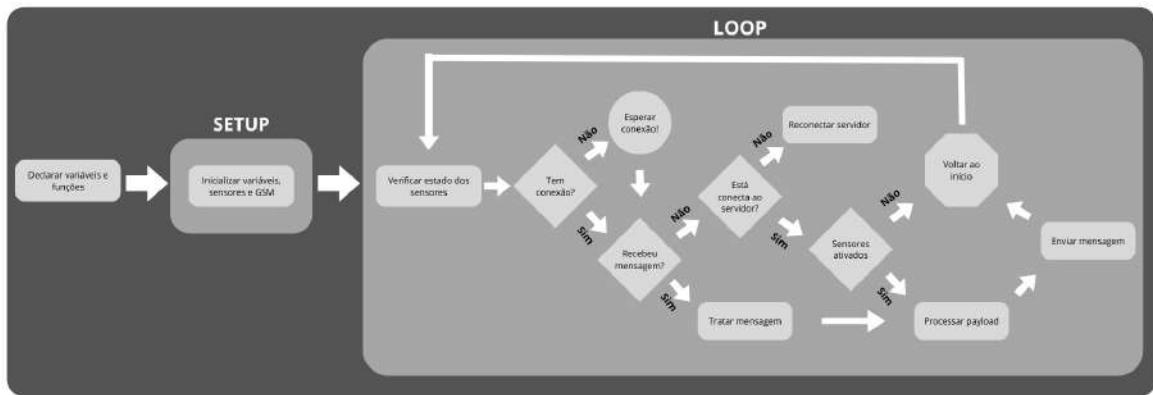
7.3 Lógica do Sistema

Nesta seção, é apresentado o funcionamento lógico do sistema.

7.3.1 Hardware

A Figura 27 apresenta o funcionamento lógico do *hardware*. Após ser ligado, o sistema inicializa as variáveis e funções. No bloco de configuração (*setup*), são definidos os tipos das variáveis, indicando se são de entrada ou saída. Nesse estágio, o modem GSM também é ativado. No bloco de repetição (*loop*), o hardware monitora continuamente os estados dos sensores de luminosidade e magnético, avaliando as condições estabelecidas. Caso uma condição seja atendida, uma tarefa específica é executada, como o envio de mensagens, conexão com o servidor MQTT, entre outras ações.

Figura 27 – Lógica do *Hardware*



Fonte: Autor

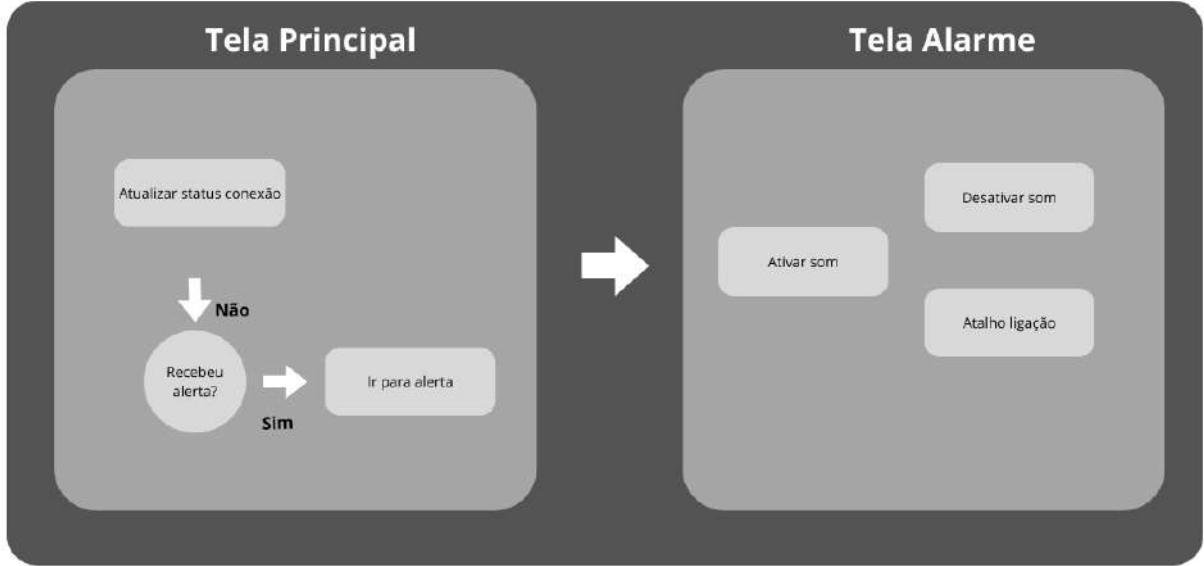
7.3.2 Aplicativo

A apresentação desta lógica foca apenas na comunicação entre o aplicativo e o *hardware* por meio do protocolo MQTT, omitindo etapas como login, cadastro e recuperação de senha. Como mostrado na Figura 28, na tela principal é possível verificar se o dispositivo ainda está conectado à rede de telefonia móvel. Paralelamente, o aplicativo verifica se recebeu alguma mensagem de alerta. Caso não tenha recebido, o sistema permanece no *loop*. Se uma mensagem for recebida, o aplicativo é direcionado para a tela do alarme. Na tela de alarme, um sinal sonoro é emitido, permitindo que o usuário desligue o alarme e acesse um atalho para realizar uma ligação telefônica.

7.4 Avaliação dos Resultados

Para monitorar a eficiência do projeto, foram inseridos comandos de impressão tanto no código do microcontrolador quanto no da aplicação móvel. Assim, é possível verificar o comportamento e a comunicação entre as partes envolvidas.

Figura 28 – Lógica do Aplicativo



Fonte: Autor

Para visualizar os logs do aplicativo no terminal, utiliza-se o comando “npx expo start”, que executa a aplicação em um ambiente de teste, exibindo todos os logs no terminal. A Figura 29 apresenta as mensagens recebidas pelo protocolo MQTT, destacando as quatro últimas impressões de log que são geradas ao receber os dados enviados pelo hardware.

Figura 29 – Impressão LOG Aplicativo

```

const auth = initializeAuth(app, {
  persistence: getReactNativePersistence(ReactNativeAsyncStorage)
});
[Component Stack]
WARN fontFamily "Inter_800ExtraBold" is not a system font and has not been loaded through expo-font. [Component Stack]
WARN fontFamily "Inter_400Regular" is not a system font and has not been loaded through expo-font. [Component Stack]
WARN fontFamily "Inter_900Black" is not a system font and has not been loaded through expo-font. [Component Stack]
WARN fontFamily "Inter_900Black" is not a system font and has not been loaded through expo-font. [Component Stack]
WARN fontFamily "Inter_800ExtraBold" is not a system font and has not been loaded through expo-font. [Component Stack]
WARN fontFamily "Inter_800ExtraBold" is not a system font and has not been loaded through expo-font. [Component Stack]
LOG true
LOG Received message on topic Hardware/Alert: Alert

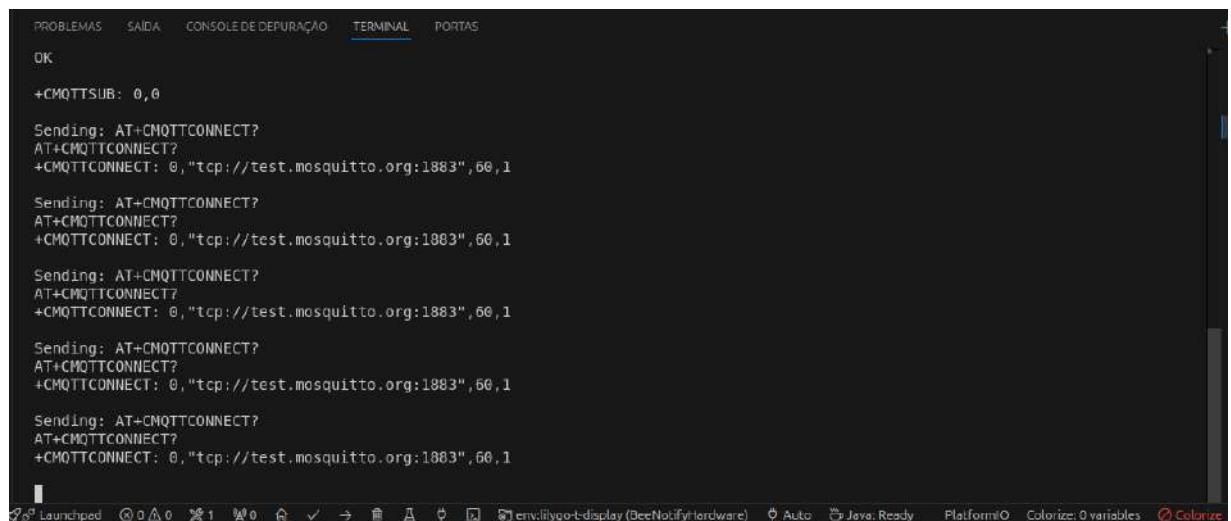
```

Fonte: Autor

Para monitorar as impressões do ESP32, a placa é conectada ao computador via cabo USB.

A partir do monitor serial de um ambiente de desenvolvimento específico para microcontroladores, é possível visualizar as mensagens exibidas no terminal. A Figura 30 ilustra o comportamento do hardware quando os requisitos da regra de negócio não são atendidos. Em intervalos de tempo definidos, a conexão com o servidor MQTT é verificada; caso esteja desconectada, uma reconexão é iniciada. Quando a regra de negócios é satisfeita, o microcontrolador inicia o processo de envio de mensagens para o aplicativo móvel. A Figura 31 o processo de envio de comandos AT com a mensagem de alerta.

Figura 30 – Verifica Conexão Servidor MQTT



The screenshot shows a terminal window with several tabs: PROBLEMAS, SAÍDA, CONSOLE DE DEPURAÇÃO, TERMINAL (which is selected), and PORTAS. The terminal output displays a series of AT commands related to MQTT connectivity:

```
PROBLEMAS   SAÍDA   CONSOLE DE DEPURAÇÃO   TERMINAL   PORTAS
OK

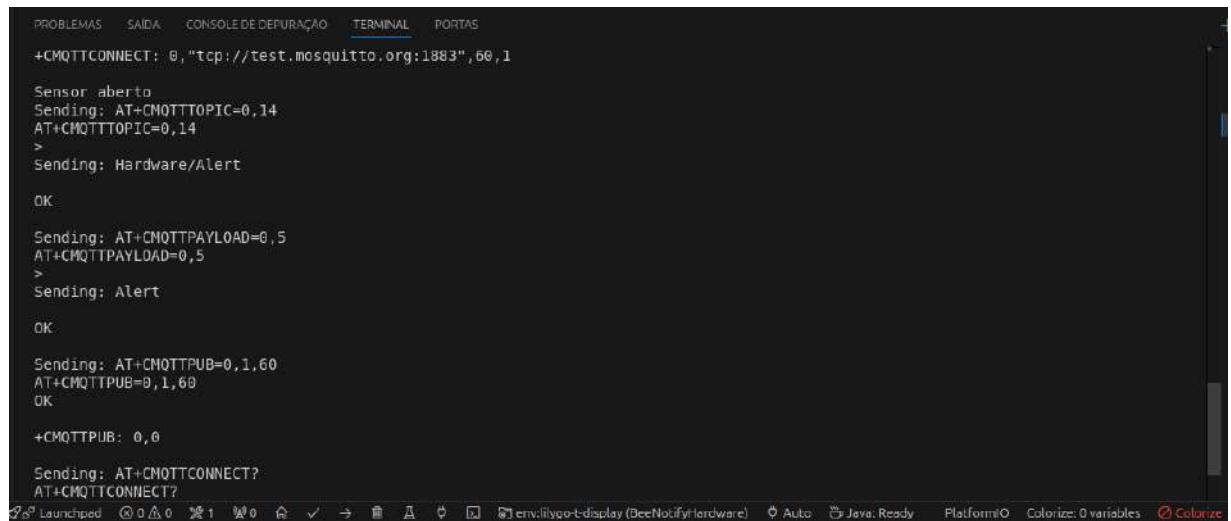
+CMOTTSUB: 0,6
Sending: AT+CMQTTCONNECT?
AT+CMQTTCONNECT?
+CMQTTCONNECT: 0,"tcp://test.mosquitto.org:1883",60,1
```

At the bottom of the terminal window, there is a toolbar with icons for file operations, a status bar showing "env:liyoo-t-display (BeeNotify hardware)" and "Auto Java Ready", and a message "PlatformIO Colorize: 0 variables Colorize".

Fonte: Autor

Com base nas evidências apresentadas nesta seção, pode-se concluir que o BeeNotify atinge seu objetivo principal, funcionando como um sistema de alarme para áreas remotas com acesso apenas à rede de telefonia móvel. Utilizando comunicação via protocolo MQTT entre seus componentes, o aplicativo desempenha a função de notificar sonoramente os alertas recebidos e verificar a comunicação com o dispositivo. O dispositivo físico, alimentado por uma bateria móvel e recarregável, monitora o perímetro designado e, em caso de violação da propriedade, envia as informações ao aplicativo. A atuação integrada dos componentes demonstra que o projeto está qualificado para cumprir suas funções conforme planejado.

Figura 31 – Envio Sinal Alerta



The screenshot shows a terminal window with several tabs at the top: PROBLEMAS, SAIDA, CONSOLE DE DEPURAÇÃO, TERMINAL (which is selected), and PORTAS. The terminal content displays the following sequence of MQTT commands:

```
+CMQTTCONNECT: 0,"tcp://test.mosquitto.org:1883",60,1
Sensor aberto
Sending: AT+CMOTTTOPIC=0,14
AT+CMOTTTOPIC=0,14
>
Sending: Hardware/Alert
OK

Sending: AT+CMQTTPAYLOAD=0,5
AT+CMQTTPAYLOAD=0,5
>
Sending: Alert
OK

Sending: AT+CMQTTPUB=0,1,60
AT+CMQTTPUB=0,1,60
OK

+CMQTTPUB: 0,0

Sending: AT+CMOTTCREATE?
AT+CMOTTCREATE?
```

At the bottom of the terminal window, there is a status bar with various icons and text, including "env:liyoo-t-display (SeeNotifyHardware)", "Auto", "Java Ready", "PlatformIO", "Colorize: 0 variables", and "Colorize".

Fonte: Autor

8

Considerações Finais

8.1 Conclusão

O uso de dispositivos em áreas remotas ou rurais, onde esses equipamentos precisam trocar informações via internet, ainda representa um desafio a ser superado. Mesmo com a expansão da internet via satélite, esse recurso pode ser custoso em algumas situações. Este trabalho tem como objetivo apresentar uma solução mais viável: utilizando a cobertura da rede de telefonia móvel, o BeeNotify mostra-se qualificado para atuar como um sistema de alarme em locais onde o acesso à internet cabeada ou à rede elétrica é limitado ou inexistente.

Os resultados apresentados na [seção 7.4](#) demonstram que o projeto atingiu seu objetivo, sendo capaz de enviar e receber informações, cumprindo sua função como sistema de alarme. O protocolo MQTT, por ser simples e leve em termos de transmissão de dados, contribui para a comunicação eficaz entre os componentes, mesmo em condições de sinal de rede móvel fraco, garantindo assim a eficiência e a confiabilidade na execução e no desempenho do sistema.

Embora o BeeNotify tenha sido originalmente projetado como um sistema de alarme para áreas de difícil acesso à internet, ele também pode ser adaptado para outras finalidades, como no agronegócio. Graças ao protocolo MQTT, é possível enviar uma quantidade maior de informações com baixo consumo de dados móveis, permitindo um monitoramento mais inteligente e eficiente das lavouras.

8.2 Trabalhos Futuros

Neste trabalho, algumas funcionalidades técnicas não foram implementadas, mas são propostas como melhorias:

- Permitir que o aplicativo rode em segundo plano, pois atualmente ele só recebe alertas

quando está aberto.

- Configurar o módulo GSM para se conectar ao servidor MQTT via protocolo SSL, aumentando a segurança na transmissão de dados. Atualmente, o código do microcontrolador se conecta a um servidor público, permitindo que qualquer dispositivo com acesso aos tópicos utilizados visualize os dados transmitidos.

Outras melhorias incluem a adição de novos tipos de sensores para atender a diferentes necessidades e a inclusão de funcionalidades extras no aplicativo, tornando o projeto mais robusto e completo.

Referências

- AGNIHOTRI, N. *AT Commands, GSM AT command set*. 2023. Disponível em: <<https://www.engineersgarage.com/at-commands-gsm-at-command-set/>>. Citado 2 vezes nas páginas 18 e 19.
- AMAZON. *Carregador Portátil Power Bank Pineng 10000 Mah V8 e Iphone*. 2023. Disponível em: <https://www.amazon.com.br/Carregador-Port%C3%A1til-Power-Pineng-Iphone/dp/B07ZRKP9KP/ref=asc_df_B07ZRKP9KP/?tag=googleshopp00-20&linkCode=df0&hvadid=379787850893&hvpos=&hvnetw=g&hvrand=244589115349551982&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcndl=&hvlocint=&hvlocphy=1031767&hvtargid=pla-900434326532&psc=1>. Citado na página 21.
- ANATEL. *Infraestrutura Panorama*. 2023. Disponível em: <<https://informacoes.anatel.gov.br/paineis/infraestrutura/panorama>>. Citado na página 13.
- ANATEL. *Panorama*. 2023. Disponível em: <<https://informacoes.anatel.gov.br/paineis/acessos/panorama>>. Citado na página 13.
- ANATEL. *Panorama*. 2023. Disponível em: <<https://informacoes.anatel.gov.br/paineis/infraestrutura/panorama>>. Citado 3 vezes nas páginas 26, 27 e 28.
- ARAUJO, G. *DESENVOLVIMENTO CROSS-PLATFORM COM REACT NATIVE: UM ESTUDO DE CASO DO APLICATIVO NAVEG*. Fortaleza, Ceará: [s.n.], 2019. Disponível em: <https://repositorio.ufc.br/bitstream/riufc/45892/3/2019_tcc_graraaujo.pdf>. Acesso em: 18 jul. 2023. Citado na página 24.
- BABIUCH, M.; FOLTÝNEK, P.; SMUTNÝ, P. Using the esp32 microcontroller for data processing. In: *2019 20th International Carpathian Control Conference (ICCC)*. [S.l.: s.n.], 2019. p. 1–6. Citado na página 15.
- BAO, Q. et al. Fully automatic infrared pyroelectric home burglar alarm design. In: *2024 IEEE 4th International Conference on Power, Electronics and Computer Applications (ICPECA)*. [s.n.], 2024. p. 163–167. Disponível em: <<https://ieeexplore-ieee-org.ez20.periodicos.capes.gov.br/document/10471139>>. Citado na página 36.
- CHAVEZ, J. H.; CHOQUE, D. C. Sistema de geolocalización con alarma y monitoreo basado en iot para personas con alzheimer. *Journal Boliviano de Ciencias, UNIVALLE*, v. 18, n. 53, p. 48–63, 2022. Disponível em: <<https://doi.org/10.52428/20758944.v18i53.373>>. Citado na página 36.
- CHOUGALE, P.; YADAV, V.; GAIKWAD, A. Firebase - overview and usage. *International Research Journal of Modernization in Engineering Technology and Science*, v. 3, n. 1, p. 1178–1183, 2022. Disponível em: <https://www.researchgate.net/profile/Anil-Gaikwad-12/publication/362539877_FIREBASE_-_OVERVIEW_AND_USAGE/links/62efc738505511283e9a5318/FIREBASE-OVERVIEW-AND-USAGE.pdf>. Citado na página 24.
- COMPONENTS101. *MC-38 - Magnetic Switch for Home Alarm System*. 2023. Disponível em: <<https://components101.com/sensors/>>

- [mc38-magnetic-switch-sensor-pinout-features-datasheet-working-alternative-application>](https://www.digitalsignage.com.br/mc38-magnetic-switch-sensor-pinout-features-datasheet-working-alternative-application.pdf). Citado na página 42.
- DANIELSSON, W. *React Native application development – A comparison between native Android and React Native*. Linköping, Sucia: [s.n.], 2016. Disponível em: <<https://www.diva-portal.org/smash/get/diva2:998793/FULLTEXT02.pdf>>. Acesso em: 16 jul. 2023. Citado na página 23.
- DIAS, A. A.; PORTO, G. S. Gestão de transferência de tecnologia na inova unicamp. *Scielo*, v. 17, n. 3, p. 263–284, 2013. Disponível em: <<https://www.scielo.br/j/rac/a/JCLpShSMZRc6NDhPv4tkNJq/?lang=pt#>>. Citado na página 35.
- EISENMAN, B. *Learning React Native*. [S.l.]: O'Reilly Media, Inc., 2015. ISBN 9781491929049. Citado na página 23.
- ELETRÔNICA, C. *MÓDULO SENSOR DE LUZ LDR*. 2023. Disponível em: <<https://copeletronica.com.br/p-modulo-sensor-de-luz-ldr-12536>>. Citado na página 18.
- ESPRESSIF. *ESP32-S3-WROOM-1U Datasheet*. 2023. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf>. Citado na página 16.
- FILHO, D. *UM PASSO A PASSO PARA A ELABORAÇÃO DO DIAGRAMA DE CASO DE USO DA UML*. Londrina, Paranaá: [s.n.], 2011. Disponível em: <<https://web.unifil.br/pergamum/vinculos/000003/00000320.pdf>>. Acesso em: 23 ago. 2023. Citado na página 33.
- FREITAS, L. G. O. de. *PROTOTIPAGEM DE SISTEMA DE ALARME DE INCÊNDIO BASEADO EM SENSOR NANOESTRUTURADO INTEGRADO A IOT*. Dissertação (Dissertação de Mestrado) — Universidade Federal do Pará, 2021. Disponível em: <https://repositorio.ufpa.br/bitstream/2011/13561/1/Dissertacao_PrototipagemSistemaAlarme.pdf>. Citado na página 35.
- FRITZING. *Regulador Tensão, Fonte, GSM*. 2023. Disponível em: <<https://fritzing.org/>>. Citado na página 42.
- GU, G.; PENG, G. The survey of gsm wireless communication system. In: *2010 International Conference on Computer and Information Application*. IEEE, 2010. p. 121–124. Disponível em: <<https://ieeexplore.ieee.org.ez20.periodicos.capes.gov.br/document/6141552>>. Acesso em: 11 jul. 2023. Citado na página 18.
- HIVEMQ. *Introducing the MQTT Protocol – MQTT Essentials: Part 1*. 2015. Disponível em: <<https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt>>. Citado na página 22.
- IFTIKHARALAM. *Maquete Png Transparente Para Celular Inteligente*. 2021. Disponível em: <https://pt.pngtree.com/freepng/smart-mobile-phone-transparent-png-mockup_5883046.html>. Citado 3 vezes nas páginas 39, 40 e 41.
- INAARA. *Imagen: Wi-fi, Internet, Sem fio*. 2023. Disponível em: <<https://pixabay.com/pt/illustrations/wi-fi-internet-sem-fio-1655553/>>. Citado na página 29.
- IO-IMAGENS. *Imagen: Cpu, Processador, Computador*. 2023. Disponível em: <<https://pixabay.com/pt/vectors/cpu-processador-computador-2103856/>>. Citado na página 29.

- KHAN, A. *Photocell (LDR) Sensor with Arduino*. 2022. Disponível em: <<https://www.circuits-diy.com/photocell-ldr-sensor-with-arduino/>>. Citado na página 17.
- KIKUCHI, A.; BETTERI, L.; ARAÚJO, L. Iot –internet das coisas: um levantamento sobre as tendências tecnológicas atuais. *Interface Tecnológica*, v. 18, n. 1, p. 118–130, 2021. Disponível em: <<https://revista.fatectq.edu.br/interfacetecnologica/article/view/1211/674>>. Citado na página 12.
- LIANG, L.; ZHU, L.; SHANG DONGYU FENG, Z. X. W. Express supervision system based on nodejs and mongodb. In: *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. IEEE, 2017. p. 607–612. Disponível em: <<https://ieeexplore-ieee-org.ez20.periodicos.capes.gov.br/document/7960064>>. Acesso em: 18 jul. 2023. Citado na página 24.
- LILYGO. *T-A7670E/G/SA R2*. 2024. Disponível em: <<https://www.lilygo.cc/products/t-sim-a7670e?srsltid=AfmBOooaNkUWwyw4pEpGZOcf2Gv2AsoxK74UTB2WYxluWnyLL3Ozi0Zck>>. Citado 2 vezes nas páginas 17 e 42.
- MACEDO, M. *Sensor Magnético Reed Switch*. 2022. Disponível em: <<https://portal.vidadesilicio.com.br/sensor-magnetico-reed-switch/>>. Citado na página 20.
- MAGRANI, E. *A Internet das Coisas*. [S.l.]: Rio de Janeiro: FGV Editora, 2018. v. 1. ISBN 978-85-225-2005-3. Citado na página 12.
- MARCHESAN, M. *SISTEMA DE MONITORAMENTO RESIDENCIAL UTILIZANDO A PLATAFORMA ARDUINO*. Santa Maria, Rio Grande do Sul: [s.n.], 2012. Disponível em: <https://www.ufsm.br/app/uploads/sites/495/2019/05/2012-Marcelo_Marchesan.pdf>. Acesso em: 11 jul. 2023. Citado na página 20.
- MONICA, P.; KUMAR, M. A.; VEMULAPALLI, S. Esp32 cam-based car security system via telegram integration. In: *2023 3rd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*. [s.n.], 2023. p. 43–48. Disponível em: <<https://ieeexplore-ieee-org.ez20.periodicos.capes.gov.br/document/10425967>>. Citado na página 36.
- MQTT. *MQTT*. 2022. Disponível em: <<https://mqtt.org/>>. Citado na página 22.
- NERI, R.; LOMBA, M.; BULHÕES, G. *MQTT*. 2019. Disponível em: <[https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/#:~:text=MQTT\(Message%20Queuing%20Telemetry%20Transport,cima%20do%20protocolo%20TCP%2FIP,>](https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/#:~:text=MQTT(Message%20Queuing%20Telemetry%20Transport,cima%20do%20protocolo%20TCP%2FIP,>)>. Citado na página 21.
- NI, S.; HAGGMAN, S.-G. Tgprs performance estimation in gsm circuit switched services and gprs shared resource systems. In: *WCNC. 1999 IEEE Wireless Communications and Networking Conference (Cat. No.99TH8466)*. IEEE, 1999. p. 1417–1421 vol.3. Disponível em: <<https://ieeexplore-ieee-org.ez20.periodicos.capes.gov.br/document/796971>>. Acesso em: 11 jul. 2023. Citado na página 18.
- OASIS. *MQTT Version 3.1.1 Plus Errata 01*. 2015. Disponível em: <<https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html>>. Citado na página 21.

- OLIVEIRA, E. *Como usar com Arduino – Sensor Magnético com fio para Alarme MC-38*. 2021. Disponível em: <<https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-magnetico-com-fio-para-alarme-mc-38>>. Citado na página 20.
- PENDER, T. *UML, A Bíblia*. Rio de Janeiro: Elsevier, 2004. Citado na página 33.
- PIXABAY. *Imagen: Servidor, Nuvem, Design*. 2023. Disponível em: <<https://pixabay.com/pt/illustrations/servidor-nuvem-design-apartamento-3297974/>>. Citado na página 29.
- PROJECT, G. B. R. *GUIDE Business Rules Project: Final Report, revision 1.2*. [S.l.], 1997. Citado na página 32.
- SARAVATI. *Módulo GSM SIM800L Quad Band Micro SIM V2.0 + Antena*. 2023. Disponível em: <https://www.saravati.com.br/modulo-gsm-sim800l-quad-band-micro-sim-v2-0-antena.html?gclid=Cj0KCQjwtamlBhD3ARIAsAARoaEzJnYeufKNhVI36lEQGDpY8_YMxKLd9nzsCBcNkRsEGya3UGwTlbkwaAr6AEALw_wcB>. Citado na página 20.
- SILVA, E.; ESPEJO, M. *1 INTERNET OF THINGS (IOT) NO AGRONEGÓCIO: Uma revisão bibliométrica sobre o campo de pesquisa*. 2020. Disponível em: <<https://periodicos.ufms.br/index.php/EIGEDIN/article/view/11475/8292>>. Citado na página 13.
- SILVA, G. Z. da et al. Atenas: Um sistema gerenciador de regras de negócio. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. *XV Simpósio Brasileiro de Engenharia de Software*. 2001. p. 338–343. Disponível em: <<https://sol.sbc.org.br/index.php/sbes/article/download/23999/23827>>. Citado na página 32.
- SOMMERVILLE, I. *Engenharia de software*. São Paulo, SP: Pearson, 2007. Translation of the original work in English. Citado 2 vezes nas páginas 30 e 31.
- SUEN, C. Y. B.; CHAN, K. T.; HUNG, C. C. L. T. K. Remote monitoring on capacity of portable power bank in testing laboratories. In: *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2017. p. 4734–4739. Disponível em: <<https://ieeexplore-ieee-org.ez20.periodicos.capes.gov.br/document/8216816>>. Acesso em: 11 jul. 2023. Citado na página 21.
- TAVARES, C. *Imagen: Smartphone, Celular, Android*. 2023. Disponível em: <<https://pixabay.com/pt/vectors/smartphone-celular-android-4076145/>>. Citado na página 29.
- USINAINFO. *Módulo Sensor de Luminosidade LDR para Arduino e ESP32*. 2023. Disponível em: <<https://www.usinainfo.com.br/sensor-de-luminosidade/modulo-sensor-de-luminosidade-ldr-para-arduino-e-esp32-2539.html>>. Citado na página 42.
- USINAINFO. *ESP32 NodeMCU IoT com WiFi e Bluetooth - 30 Pinos*. 2024. Disponível em: <<https://www.usinainfo.com.br/nodemcu/esp32-nodemcu-iot-com-wifi-e-bluetooth-30-pinios-5147.html>>. Citado na página 16.