Universidade Federal de Ouro Preto - UFOP - Escola de Minas - Colegiado do curso de Engenharia de Controle e Automação - CECAU

Raphael Iannini Dutra dos Santos

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO E MITIGAÇÃO DE INCÊNDIOS E VAZAMENTO DE GÁS VIA APLICATIVO

Monografia de Graduação em Engenharia de Controle e Automação

Ouro Preto, 2022

Raphael Iannini Dutra dos Santos

AUTOMAÇÃO RESIDENCIAL PARA MONITORAMENTO E MITIGAÇÃO DE INCÊNDIOS E VAZAMENTO DE GÁS VIA APLICATIVO

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: prof. Adrielle de Carvalho Santana

Ouro Preto, 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S237a Santos, Raphael Iannini Dutra dos.
Automação residencial para monitoramento e mitigação de incêndios e vazamento de gás via aplicativo. [manuscrito] / Raphael Iannini Dutra dos Santos. - 2022.
92 f.: il.: color..

Orientadora: Profa. Dra. Adrielle de Carvalho Santana.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. Incêndios. 2. Automação residencial. 3. Monitoramento. 4. Gás - Vazamentos. 5. Aplicativos móveis. I. Santana, Adrielle de Carvalho. II. Universidade Federal de Ouro Preto. III. Título.

CDU 681.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Raphael Iannini Dutra dos Santos

Automação Residencial Para Monitoramento e Mitigação de Incêndios e Vazamento de Gás Via Aplicativo

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia de Controle e Automação

Aprovada em 26 de Outubro de 2022

Membros da banca

Dra. Adrielle de Carvalho Santana - Orientadora (DECAT - Universidade Federal de Ouro Preto)
Dr. Agnaldo José da Rocha Reis - Convidado (DECAT - Universidade Federal de Ouro Preto)
Dr. Eduardo José da Silva Luz - Convidado (DECOM - Universidade Federal de Ouro Preto)

Adrielle de Carvalho Santana, orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 31/10/2022

Documento assinado eletronicamente por **Adrielle de Carvalho Santana**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 31/10/2022, às 22:01, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0416870** e o código CRC **649411FC**.

Referência: Caso responda este documento, indicar expressamente o Processo n° 23109.014570/2022-98 SEI n° 0416870

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Agradeço a toda a minha família, especialmente a minha mãe, Andrea Maria Iannini Dutra, por todo o apoio, amizade, e presença constante em todos os momentos.

Agradeço a Escola de Minas, e a Universidade Federal de Ouro Preto, por todo o aprendizado, amizades, e crescimento profissional e pessoal ao longo da minha graduação.

Agradeço a professora Adrielle de Carvalho Santana por todo o auxílio, disponibilidade, dedicação, e companheirismo na orientação deste trabalho. Expresso minha gratidão também aos professores Agnaldo José da Rocha Reis, e Eduardo José da Silva Luz, por todos os ensinamentos, amizade, e companheirismo ao longo do curso, e a todos os professores do DECAT e DECOM que fizeram parte da minha história.

Agradeço aos amigos e colegas do curso de Engenharia de Controle e Automação, e da Ciência da Computação, por tornarem os estudos e trabalhos mais prazerosos.

Agradeço a Deborah Puperi, por todo o apoio na confecção da arte desenvolvida para o aplicativo *mobile* deste trabalho.

Agradeço também aos meus amigos, por fazerem parte da minha história, por todo o aprendizado, alegrias, e apoio ao longo de todos esses anos.

“A ciência de hoje é a tecnologia de amanhã.” (Edward Teller)

Resumo

Este trabalho tem como objetivo o desenvolvimento de um sistema residencial de baixo custo, para monitoramento e mitigação de incêndios e vazamentos de gás via aplicativo. Para tal, foi realizado um estudo para selecionar os componentes, sensores, e a placa microcontroladora mais adequada para a proposta, bem como as ferramentas e plataformas ideais para a construção do aplicativo mobile. A partir daí, foi desenvolvido um sistema denominado *FireSafe*, composto por um protótipo físico que tem como microcontrolador o módulo ESP32, um banco de dados de tempo real criado no *Google Firebase*, e um aplicativo mobile desenvolvido na plataforma *Kodular*. O protótipo físico conta com um sensor de gás MQ-6, um sensor de temperatura LM35, um sensor de chamas, e um módulo de alerta e mitigação de incidentes composto por um LED, um *buzzer* ativo, e um módulo relé. Graças a programação desenvolvida para o protótipo, e a aplicação dos ajustes necessários para o correto funcionamento dos sensores, o *hardware* foi capaz de perceber as ocorrências de situações de risco no ambiente, relacionadas a vazamentos de gás GLP, princípios de incêndios, e detecção de temperaturas elevadas, acionando os atuadores de segurança quando necessário para mitigar os acidentes, e fornecendo valores de concentração de gás com erros entre 0,6% a 1% nas situações mais extremas, bem como valores de temperaturas com erros entre 0,5°C a 1°C. Os dados monitorados por meio do protótipo físico são enviados ao banco de dados, e exibidos diretamente no aplicativo mobile por meio de uma tela principal simples e intuitiva, o que permitiu não apenas o monitoramento do ambiente pelos usuários, como também a sinalização das situações de risco em tempo real, de forma remota. Além disso, o custo total do desenvolvimento do trabalho foi levado em consideração, e o orçamento final foi de R$352,97. Por fim, conclui-se que o sistema proposto cumpre os objetivos para os quais ele foi desenvolvido.

Palavras-chaves: Incêndios; Automação residencial; Monitoramento; Vazamentos de gás e Aplicativo mobile.

Abstract

This work aims to develop a low-cost residential system for monitoring and mitigating fires and gas leaks through a mobile application. For that, a study was carried out to select the components, sensors, and the most suitable microcontroller board for the proposal, as well as the ideal tools and platforms for the construction of the mobile application. From there, a system called *FireSafe* was developed, consisting of a physical prototype that has the ESP32 module as a microcontroller, a real-time database created in *Google Firebase*, and a mobile application developed in *Kodular* platform. The physical prototype has an MQ-6 gas sensor, an LM35 temperature sensor, a flame sensor, and an incident alert and mitigation module consisting of an LED, an active *buzzer*, and a relay module. Thanks to the programming developed for the prototype, and the application of the necessary adjustments for the correct functioning of the sensors, the *hardware* was able to perceive the occurrence of risk situations in the environment, related to LPG gas leaks, beginning of fires, and detection of high temperatures, activating safety actuators when necessary to mitigate accidents, and providing gas concentration values with errors between 0.6% to 1% in the most extreme situations, as well as temperature values with errors between 0.5°C to 1°C. The data monitored through the physical prototype is sent to the database, and displayed directly in the mobile application through a simple and intuitive main screen, which allowed not only the monitoring of the environment by users, but also the signaling of the risk situations in real time, remotely. In addition, the total cost of developing the work was taken into account, and the final budget was R$352,97. Therefore, it is concluded that the proposed system fulfills the objectives for which it was developed.

Key-words: Fires; Home automation; Monitoring; Gas leaks and Mobile application.

Lista de ilustrações

Figura 1 Distribuição das ocorrências de incêndios. Fonte: (SPRINKLER, 2019) . . . 16
Figura 2 Comparativo entre os números de registros de incêndios. Fonte: (JRS, 2021) 17
Figura 3 Orçamento do sistema de monitoramento de gás com alarme via SMS. Fonte: (PINTO, 2016) . . . 22
Figura 4 Orçamento do sistema residencial para monitoramento de gás. Fonte: (ALEXANDRE, 2021) . . . 22
Figura 5 Composição química do GLP. Fonte: (SILVA, 2016) . . . 23
Figura 6 Tipos de recipientes comercializados do GLP. Fonte: (PETROBRAS, 2022) 24
Figura 7 Diagrama de blocos funcional do ESP32. Fonte: (ESPRESSIF, 2022a) . . . 26
Figura 8 placa DOIT ESP32 Devkit V1. Fonte: (DOCS, 2022) . . . 27
Figura 9 Mapeamento dos pinos da placa DOIT ESP32 Devkit V1. Fonte: (TUTORIALS, 2022) . . . 27
Figura 10 Arduino UNO R3. Fonte: (ARDUINO, 2022) . . . 28
Figura 11 Quadro comparativo entre Arduino UNO R3, ESP32, e ESP8266. Fonte: (GHOSH, 2019) . . . 28
Figura 12 Quadro com os sensores da série MQ e os gases os quais apresentam sensibilidade. Fonte: (ALVES; MENDONÇA, 2018) . . . 29
Figura 13 Sensor MQ-6 Fonte: (PINTO, 2016) . . . 30
Figura 14 Modelo esquemático do sensor MQ-6 Fonte: Adaptado de (SENSORS, 2022) 31
Figura 15 Sensor LM35 Fonte: Adaptado de (INSTRUMENTS, 2000) . . . 31
Figura 16 Sensor de chamas. Fonte: (JOSEPH, 2022) . . . 33
Figura 17 Módulo relé de 5V e 1 canal. Fonte: (FLOP, 2022a) . . . 34
Figura 18 Composição de um *buzzer*. Fonte: (MAKERS, 2022) . . . 35
Figura 19 Tipos de *buzzer*. Fonte: (MAKERS, 2022) . . . 35
Figura 20 Modelo esquemático de um LED. Fonte: (PINTO, 2016) . . . 36
Figura 21 Estrutura do material semicondutor de um LED. Fonte: (TAKAHASHI, 2022) 36
Figura 22 Tela exibida na aba *Designer*. Fonte: Autoria própria. . . . 38
Figura 23 Tela exibida na aba *Blocks*. Fonte: (DJASSEMI, 2020) . . . 38
Figura 24 Fluxograma completo do *FireSafe*. Fonte: Autoria Própria . . . 41
Figura 25 Curvas da sensibilidade característica do MQ-6. Fonte: Adaptada de Today (2014) . . . 43
Figura 26 Circuito elétrico do divisor de tensão. Fonte: Autoria própria . . . 44

Figura 27 Variação da concentração de GLP. Fonte: Autoria própria 45
Figura 28 Circuito do *buzzer*. Fonte: Autoria própria 47
Figura 29 Circuito inicial do *FireSafe* (*protoboard*). Fonte: Autoria Própria 48
Figura 30 Circuito final do FireSafe (*protoboard*). Fonte: Autoria Própria 49
Figura 31 Fluxograma da programação do *hardware FireSafe* (1). Fonte: Autoria Própria 50
Figura 32 Fluxograma da programação do *hardware FireSafe* (2). Fonte: Autoria Própria 51
Figura 33 *Realtime Database* do *FireSafe*. Fonte: Autoria Própria 53
Figura 34 Tela de registro de aplicativo ao *Firebase*. Fonte: Autoria Própria 54
Figura 35 Tela de registro de aplicativo no *Firebase*. Fonte: Autoria Própria 54
Figura 36 Tela de inicialização do aplicativo *FireSafe*. Fonte: Autoria Própria 55
Figura 37 Tela de monitoramento do aplicativo *FireSafe*. Fonte: Autoria Própria 56
Figura 38 Configuração de acesso ao *Firebase* no aplicativo *FireSafe*. Fonte: Autoria Própria 56
Figura 39 Voltagem fornecida VS Valor ADC interpretado no ESP32. Fonte: Tutorials (2019) 59
Figura 40 Quadro com os 10 valores mensurados. Fonte: Autoria própria. 59
Figura 41 Curvas computadas para os valores de tensão. Fonte: Autoria própria. 60
Figura 42 Interferências no ADC1 que afetam as leituras do sensor LM35. Fonte: Autoria própria 61
Figura 43 Testes da solução adotada para contornar a interferência nos valores da temperatura. Fonte: Autoria própria. 62
Figura 44 Teste das interferências do ADC1 nos valores lidos por meio do sensor MQ-6. Fonte: Autoria própria. 63
Figura 45 *Hardware* do *FireSafe* em funcionamento após montagem em uma placa ilhada. Fonte: Autoria própria. 64
Figura 46 Fluxo de envio de dados do *hardware* ao *Firebase*. Fonte: Autoria própria. . 65
Figura 47 Fluxo de envio de dados do *Firebase* ao aplicativo. Fonte: Autoria própria. . 66
Figura 48 Monitoramento do cenário em que todas as situações de risco ocorrem simultâneamente no ambiente. Fonte: Autoria própria. 66
Figura 49 Quadro de orçamento do projeto. Fonte: Autoria própria. 67

Figura B.1.1 Bloco de inicialização da tela 88
Figura B.1.2 Bloco que aciona o contador (3 segundos) 88
Figura B.1.3 Bloco que aciona a próxima tela (*Screen*) 88
Figura B.1.4 Bloco de chamada da próxima tela 88
Figura B.2.1 Declaração da variável global de porcentagem de gás GLP 89
Figura B.2.2 Declaração da variável global de presença de chamas 89
Figura B.2.3 Declaração da variável global de temperatura 89

Figura B.2.4 Bloco de inicialização da tela, que invoca a leitura dos primeiros valores no *Firebase* 89
Figura B.2.5 Bloco que concretiza a solicitação de leitura dos primeiros valores no *Firebase* 89
Figura B.2.6 Bloco que aciona o processamento de dados após retorno de valores do *Firebase* 90
Figura B.2.7 Bloco que aciona o processamento de dados sempre que os valores salvos no *Firebase* forem atualizados 90
Figura B.2.8 Bloco que encerra o aplicativo por meio do botão 90
Figura B.2.9 Bloco de processamento dos dados 91

Lista de abreviaturas e siglas

UFOP	Universidade Federal de Ouro Preto
CECAU	Colegiado do curso de Engenharia de Controle e Automação
PIB	Produto Interno Bruto
EUA	*United States of America* (Estados Unidos da América)
GLP	Gás Liquefeito de Petróleo
COVID-19	*(Co)rona (Vi)rus (D)esease* (Doença do Corona Vírus)
IoT	*Internet of Things* (Internet das Coisas)
Wi-Fi	emphWireless Fidelity (Fideldiade sem Fio)
RAM	*Random Access Memory* (Memória de Acesso Randômico)
ROM	*Read-Only Memory* (Memória somente de Leitura)
GPIO	*General Purpose Input/Output* (Entradas e Saídas de Propósito Geral)
PWM	*Pulse Width Modulation* (Modulação de Largura de Pulso)
MQ	*Quality Monitoring* (Monitoramento de Qualidade)
PTC	*Positive Temperature Confficient* (Coeficiente Positivo de Temperatura)
NTC	*Negative Temperature Confficient* (Coeficiente Negativo de Temperatura)
LED	*Negative Temperature Confficient* (Coeficiente Negativo de Temperatura)
NA	Normalmente Aberto
NF	Normalmente Fechado
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
MIT	*Massachusetts Institute of Technology* (Instituto de Tecnologia de Massachusetts)
RTOS	*Real Time Operating System* (Sistema Operativo em Tempo Real)

JSON	*JavaScript Object Notation* (Notação de Objeto do JavaScript)
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)

Lista de símbolos

$\%$	Porcentagem
KB	Kilobytes
MHz	Mega Hertz
mV	Milivolts
V	Volts
ppm	Parte por milhão
$^{\circ}$C	Graus Celsius
nm	Nanômetro

Sumário

1 Introdução 16
1.1 Contextualização 16
1.2 Objetivos 18
1.2.1 Objetivos Específicos 19
1.3 Escopo do trabalho 19

2 Fundamentação Teórica 20
2.1 Sistemas similares 21
2.2 Gás Liquefeito de Petróleo 23
2.2.1 Definição e características 23
2.2.2 Principais aplicações 24
2.2.3 Perigos do GLP 25
2.3 *Hardware* 25
2.3.1 Microcontroladores 25
2.3.1.1 ESP32 26
2.3.1.2 Arduino UNO R3 27
2.3.2 Sensores 29
2.3.2.1 Sensores de gás MQ 29
2.3.2.2 Sensor de Gás MQ-6 30
2.3.2.3 Sensor LM35 31
2.3.2.4 Termistor 32
2.3.2.5 Sensor de Chamas 32
2.3.3 Módulo Relé 33
2.3.4 Buzzer 34
2.3.5 LED 35
2.3.6 *Softwares* 37
2.3.6.1 IDE do Arduino 37
2.3.6.2 *Kodular* 37
2.4 Banco de Dados 39
2.4.1 *Firebase* 39

3 Desenvolvimento 40
3.1 *Hardware* do *FireSafe* 41
3.1.1 Módulo de detecção de chamas 42
3.1.2 Módulo de monitoramento de gás 42

3.1.2.1 Tratamento de dados . . . 43
3.1.3 Módulo de monitoramento da temperatura . . . 45
3.1.3.1 Tratamento de dados . . . 46
3.1.4 Módulo de alerta e mitigação de incidentes . . . 46
3.1.5 Sinalização de erros e status de conexão . . . 47
3.1.6 Montagem do *Hardware* . . . 48
3.1.7 Programação do *Hardware* . . . 49
3.2 Banco de Dados . . . 52
3.3 Aplicativo *FireSafe* . . . 54

4 Resultados . . . 58
4.1 *Hardware* do *FireSafe* . . . 58
4.1.1 Testes no Módulo de Monitoramento da Temperatura . . . 58
4.1.2 Testes no Módulo de Monitoramento de Gás . . . 61
4.1.3 Testes no Módulo de Detecção de Chamas . . . 63
4.1.4 Testes no Módulo de Alerta e Mitigação de Incidentes . . . 63
4.1.5 Montagem do protótipo em uma placa ilhada . . . 64
4.2 Validações no Banco de Dados do *Firebase* . . . 65
4.3 Aplicativo *FireSafe* . . . 65
4.4 Orçamento do Sistema *FireSafe* . . . 67

5 Considerações Finais . . . 68
5.1 Conclusões . . . 68
5.2 Melhorias Futuras . . . 69

Referências . . . 71

Apêndices 75

APÊNDICE A Programação ESP32 . . . 76

APÊNDICE B Programação do Aplicativo *FireSafe* . . . 88
B.1 Tela de Inicialização . . . 88
B.2 Tela de Monitoramento . . . 89

1 Introdução

1.1 Contextualização

A ocorrência de incêndios em edificações é algo extremamente frequente em todo o mundo, e abrange não apenas os estabelecimentos comerciais e industriais, mas também os depósitos e as residências. Segundo Sprinkler (2019), somente em 2019 no Brasil foram contabilizadas 866 ocorrências de incêndios estruturais, onde o maior número de registros foi referente aos estabelecimentos comerciais (com 215 registros), e aos depósitos (com 187 registros). A Figura 1 ilustra a distribuição das ocorrências de incêndios entre os diferentes tipos de estabelecimentos no Brasil, em 2019.

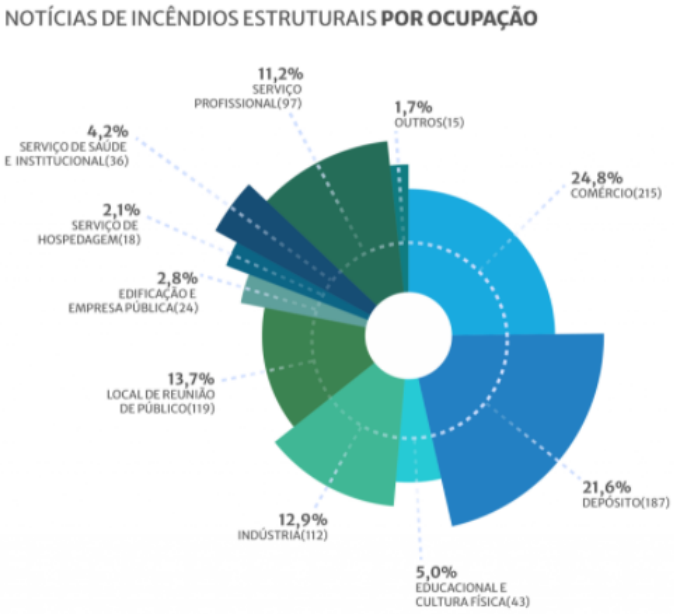

Figura 1 – Distribuição das ocorrências de incêndios. Fonte: (SPRINKLER, 2019)

O número de ocorrências de incêndios estruturais é ainda maior se compararmos com o ano de 2020, onde segundo JRS (2021), o monitoramento diário, que é realizado no país pelo Instituto Sprinkler Brasil, contabilizou 1244 ocorrências, resultando em um cenário que apresenta cerca de 43,7% de aumento nos registros. A Figura 2 ilustra um comparativo entre o número de notificações de incêndios registrados pelo instituto, entre os anos de 2018 e 2020.

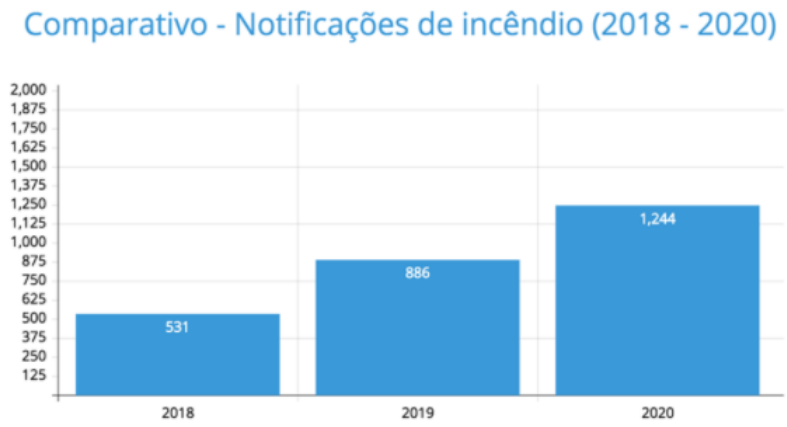

Figura 2 – Comparativo entre os números de registros de incêndios. Fonte: (JRS, 2021)

Segundo FBN (2021), foram divulgados dados estatísticos do Corpo de Bombeiros do Estado de São Paulo que mostram um total de 3.585 incêndios atendidos entre os meses de Janeiro e Julho de 2021, totalizando cerca de 18 atendimentos a incêndios em residências e empresas por dia no país. Infelizmente, boa parte das ocorrências de incêndios apresentam registros de pessoas feridas, ou até mesmo de mortes. Para se ter uma ideia, ainda no primeiro semestre de 2021 os incêndios ocorridos em residências na grande São Paulo resultaram na morte de 20 pessoas G1 (2021). Os incêndios também ocorrem com frequência em outros estados, como, por exemplo, em Minas Gerais, onde houveram 420 ocorrências em Uberlândia no ano de 2017, das quais cerca de 30% ocorreram em residências (G1, 2017).

O cenário também é crítico mesmo em países desenvolvidos, e o histórico de ocorrências não se restringe apenas aos tempos atuais com o aumento de *gadgets* e eletrônicos no dia a dia das pessoas. Nos Estados Unidos da América (EUA), por exemplo, houve um registro de cerca de 1,6 milhões de incêndios no ano de 2005, resultando no registro de 3677 mortes das quais 83% foram de ocorrências de incêndios em residências (SEITO et al., 2008).

Diante disso, fica evidente que esses incidentes resultam em um grande número de mortes a cada ano, além de deixar centenas de pessoas feridas e alavancar acentuados prejuízos, incluindo o desalojamento de pessoas que perderam suas casas para as chamas. Visando ilustrar este cenário, Corrêa et al. (2015) relatam em seu trabalho que bilhões de dólares americanos em perdas de propriedade ocorrem devido a incêndios em diversos países, e ilustram, com o trabalho de Ramachandran (2008), que os custos com incêndios chegam a representar cerca de 0,813% do Produto Interno Bruto (PIB) dos EUA, bem como 0,864% do PIB da Dinamarca.

Analisando-se as ocorrências de incêndios no âmbito residencial, nota-se que a origem de grande parte deles é devido a falhas e sobrecargas nas redes elétricas dessas edificações, além de vazamentos de gás de cozinha (GLP). Segundo FBN (2021), as falhas elétricas e os acidentes domésticos foram os principais responsáveis pelos incêndios ocorridos em residências no primeiro semestre de 2021. Um dos motivos para isso foi o aumento do cenário de *Home Office*

e da forma de trabalho Híbrida, devido a pandemia de COVID-19, que resultou em um aumento no número de equipamentos interligados nas redes elétricas domésticas, simultaneamente.

Esse fator provocou o aumento de excesso de cargas nas redes elétricas domésticas, provocando casos de curtos-circuitos e superaquecimentos de condutores, que resultaram em incêndios. Além disso, o maior fluxo de pessoas nas residências ao longo do dia ocasionou um aumento nos acidentes devido a manipulação de gás de cozinha e de produtos inflamáveis, bem como maiores incidentes com chamas e faíscas.

Sendo assim, conforme os dados apresentados até o momento, pode-se ter uma melhor visibilidade da gravidade dos danos econômicos e dos riscos à vida que os incêndios estruturais proporcionam, bem como da frequente ocorrência deles em áreas residenciais. Logo, fica evidente a importância do desenvolvimento de soluções que auxiliem na mitigação e na prevenção desses incidentes, de forma eficiente e segura.

Felizmente, é possível utilizar da tecnologia e da domótica (ou automação residencial) para propor soluções a esse cenário, uma vez que a automação residêncial nos permite aplicar técnicas de controle e automatização em equipamentos que estão interligados entre si, nos diferentes cômodos da casa. Nesse aspecto, é possível desenvolver diferentes soluções capazes de proporcionar conforto, bem-estar e segurança aos residentes de acordo com suas necessidades (MURATORI; BÓ, 2013).

Aliado à domótica tem-se também o conceito de Internet das Coisas (IoT), que segundo a Oracle (2019), diz respeito a rede de "objetos físicos" que são incorporados a sensores e *softwares*, com o objetivo de conectar e trocar dados com outros dispositovos por meio da internet. Neste aspecto, é possível conectar as soluções desenvolvidas na domótica com a internet, permitindo, por exemplo, o monitoramento remoto e em tempo real dos ambientes das residências.

Portanto, o foco deste trabalho é voltado para o desenvolvimento de um sistema de automação residencial de baixo custo que permite o monitoramento e a mitigação de incêndios e de vazamentos de gás, utilizando de conceitos de domótica e IoT para proporcionar ambientes mais seguros às pessoas.

1.2 Objetivos

Dada toda a contextualização referente aos perigos e aos danos que os incêndios podem proporcionar, o objetivo adotado para este trabalho é o desenvolvimento de um sistema residencial de baixo custo para monitoramento e mitigação de incêndios e vazamento de gás.

Diante disso, o sistema é capaz de perceber a ocorrência de situações de risco no ambiente e acionar atuadores para mitigar os possíveis incidentes, bem como alertar os residentes.

1.2.1 Objetivos Específicos

Os objetivos específicos para o desenvolvimento do trabalho são:

- Desenvolver o sistema propriamente dito, considerando o microcontrolador que será aplicado, seus respectivos atuadores, sensores, e técnicas de controle e monitoramento que serão adotadas.
- Construir um aplicativo mobile que possibilite o monitoramento remoto do ambiente no qual o sistema foi instalado, visando maior conforto e maior margem de segurança.
- Utilizar a menor quantidade de recursos financeiros possível, a fim de demonstrar a possibilidade de aplicar soluções que aumentem a segurança do ambiente residenciail de forma acessível.

1.3 Escopo do trabalho

Para facilitar a organização e o desenvolvimento, este presente trabalho foi dividido em cinco capítulos distintos. O primeiro capítulo é a introdução ao tema do trabalho, onde é apresentada a motivação que levou ao desenvolvimento do tema, além dos objetivos gerais e específicos que se esperam alcançar após a elaboração do trabalho. Além disso, também é apresentado o escopo do trabalho, ilustrando como o mesmo foi organizado e dividido. O segundo capítulo consiste na revisão da literatura realizada para a confecção do trabalho, onde são apresentados os principais componentes físicos, *hardwares*, *softwares*, plataformas de desenvolvimento *mobile*, e materiais que podem ser utilizados para a construção do sistema, considerando os objetivos especificados. O terceiro capítulo consiste na metodologia utilizada no desenvolvimento, onde os materiais e tecnologias escolhidos para a construção do sistema são apresentados junto as estratégias utilizadas. No quarto capítulo são apresentados os resultados obtidos após o desenvolvimento, onde o orçamento do sistema é realizado, seu funcionamento é demonstrado, e as soluções para os problemas encontrados são apresentadas. Por fim, o quinto capítulo consiste nas principais conclusões obtidas com a realização do trabalho, além de apresentar as propostas para possíveis trabalhos futuros.

2 Fundamentação Teórica

Para a construção do projeto proposto neste trabalho é necessário realizar um estudo prévio sobre o gás liquefeito de petróleo (GLP), visto que o monitoramento de suas concentrações no ambiente residencial faz parte dos requisitos do sistema.

Além disso, para construir o *hardware* do sistema propriamente dito será preciso utilizar diversos recursos eletrônicos, como os elementos sensores, os LEDs, e os módulos relés. De forma análoga, também será preciso utilizar algumas ferramentas em forma de *software* para desenvolver a programação do sistema, além de plataformas de desenvolvimento mobile para a construção do aplicativo.

Contudo, existe grande variedade de equipamentos e ferramentas disponíveis no mercado que podem ser aplicados no desenvolvimento, e portanto, a revisão bibliográfica dos mesmos se torna essencial para a escolha dos componentes e ferramentas mais adequados aos objetivos do trabalho, levando em consideração o baixo custo, conectividade, e a eficiência do sistema.

Pensando nesses aspectos se inicia toda a seção de estudos para a concretização do trabalho, onde é apresentado a revisão da literatura a respeito dos trabalhos similares, os conceitos relevantes do gás liquefeito de petróleo, e todos os componentes e ferramentas encontrados que são adequados para cumprir os objetivos.

- Sistemas similares
- Gás Liquefeito de Petróleo
  - Definição e características
  - Principais aplicações
  - Perigos do GLP
- *Hardware*
  - Microcontroladores
    - ESP 32
    - Arduino UNO R3
  - Sensores
    - Sensores de gás MQ
    - Sensores de gás MQ-6

    - Sensor LM35
    - Termistor
    - Sensor de Chamas
    - Módulo Relé
    - Buzzer
    - LED

- *Softwares*
  - IDE do Arduino
  - *Kodular*
- Banco de Dados
  - *Firebase*

2.1 Sistemas similares

Esta etapa consistiu na pesquisa por trabalhos com propostas similares ao deste projeto, onde as estratégias adotadas para a construção dos protótipos voltados ao âmbito residencial, e os respectivos resultados que foram obtidos, auxiliaram na escolha dos componentes e ferramentas adotados na etapa de desenvolvimento deste trabalho.

Neste contexto, cita-se o trabalho desenvolvido por Pinto (2016), que consistiu na criação de um sistema de monitoramento de gás GLP com alarme via SMS. Em sua metodologia, Pinto (2016) utilizou o sensor MQ-6 para aferir as concentrações de gás GLP no ambiente, e a placa de desenvolvimento Arduino UNO como central para processamento dos dados. Como resultados, o sistema foi capaz de mensurar os níveis de gás GLP no ambiente, acionar os atuadores para prevenção de acidentes nas situações de risco, e enviar os sinais de alerta aos usuários. Além disso, o orçamento para a concretização do protótipo também foi levado em consideração, e é ilustrado por meio do quadro apresentado na Figura 3, onde o valor final da aquisição dos componentes foi de R$304,36 (ptax de compra do Dólar EUA igual a 3,2795, na data 19/07/2016).

De forma similar, Alexandre (2021) utilizou o sensor MQ-5 para desenvolver um sistema residencial de monitoramento de gás GLP, por meio da plataforma Arduino UNO e IOT (*Internet of Things*). Como resultados, também foi possível mensurar as concentrações de gás GLP no ambiente, bem como emitir alertas sonoros e visuais nas situações de vazamentos, e apresentar os dados das concentrações aos usuários por meio de um aplicativo *Android* desenvolvido com a linguagem de programação *Java*. O orçamento final para a concretização do protótipo foi de R$ 175,96 (ptax de compra do Dólar EUA igual a 5,1972, na data 19/07/2021), entretanto, a

Orçamento do projeto			
Item	**Valor unitário**	**Quantidade**	**Valor total**
Arduino UNO	R$ 68,30	1	R$ 68,30
Sensor de gás MQ-6	R$ 24,90	1	R$ 24,90
Shield GSM/GPRS SIM900+ Antena	R$ 108,53	1	R$ 108,53
Buzzer 5V	R$ 1,90	1	R$ 1,90
LED Vermelho 5mm	R$ 0,79	1	R$ 0,79
LED verde 5mm	R$ 0,79	1	R$ 0,79
Resistor 330 ohms	R$ 0,60	2	R$ 1,20
Potenciômetro	R$ 1,25	1	R$ 1,25
Shield relé 5V 2 canais	R$ 19,90	1	R$ 19,90
Protoboard 170 pinos	R$ 5,90	1	R$ 5,90
Fonte 9V/1A	R$ 25,00	1	R$ 25,00
Recarga de celular	R$ 10,00	2	R$ 20,00
Display LCD	R$ 25,90	1	R$ 25,90
TOTAL	**-**	**-**	**R$ 304,36**

Figura 3 – Orçamento do sistema de monitoramento de gás com alarme via SMS. Fonte: (PINTO, 2016)

montagem física limitou-se apenas a *protobard*, isto é, o protótipo não foi desenvolvido em uma placa integrada dedicada. Além disso, conforme representado no quadro da Figura 4, foram necessários menos componentes para a concretização do protótipo se comparado ao trabalho desenvolvido por Pinto (2016), o que justificou a diferença nos valores finais dos orçamentos.

Produto	Quantidade	Valor
Arduino UNO	1	R$ 69,89
Protoboard	1	R$ 14,90
Jumpers fêmea	12	R$ 6,00
Jumpers macho	12	R$ 6,00
Módulo bluetooth HC-05	1	R$ 41,45
Módulo buzzer grove	1	R$ 13,90
Sensor MQ-5	1	R$ 23,82
Total		R$ 175,96

Figura 4 – Orçamento do sistema residencial para monitoramento de gás. Fonte: (ALEXANDRE, 2021)

Por fim, vale ressaltar que ambos os sistemas pesquisados não contam com módulos específicos para o monitoramento dos princípios de incêndio, e da temperatura no ambiente, que fazem parte dos objetivos deste trabalho.

2.2 Gás Liquefeito de Petróleo

2.2.1 Definição e características

O gás liquefeito de petróleo (GLP), conhecido popularmente como "gás de cozinha" ou "gás de botijão", é definido por Dantas (2010) como a forma líquida de uma substância derivada do petróleo que, ao ser exposta a condições normais de temperatura e pressão, assume a forma de um gás. Essa substância é obtida por meio da mistura de moléculas de propano e de butano, em conjunto com outros compostos que são utilizados em menores quantidades, tais como o isobutano, o propeno, o n-butano, e o buteno.

A mistura desses elementos é realizada de forma a se obter um composto químico (substância) bem estruturado, formado por moléculas de hidrocarbonetos que contêm de três a quatro átomos de carbono, dando origem ao GLP. São justamente essas particularidades na sua composição que permitem ao GLP assumir a forma liquefeita caso seja submetido a compressão ou ao resfriamento (PETROBRAS, 2022). A Figura 5 ilustra a estrutura da composição química do GLP, onde círculos em preto representam os átomos de carbono e os círculos na cor cinza representam átomos de hidrogênio.

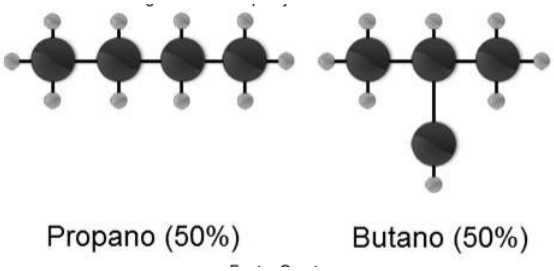

Figura 5 – Composição química do GLP. Fonte: (SILVA, 2016)

Em seu estado natural, o gás GLP tem como característica ser incolor e inodoro, o que dificulta sua detecção em casos de vazamentos. Logo, por se tratar de um produto altamente inflamável e de fácil combustão, é necessário adicionar uma pequena quantidade de enxofre em sua composição, como uma medida de segurança. Essa adição faz com que o produto adquira um odor característico que pode ser facilmente identificado pelas pessoas, facilitando sua detecção na ocorrência de vazamentos (OLIVEIRA; SHINOHARA, 2014).

Também vale destacar que quando o GLP está sob efeito de combustão, as chamas que são produzidas possuem uma coloração específica graças aos elementos butano e propano. Isso acontece porque o propano é um gás mais leve que o butano, e é o responsável por gerar a chama de cor azul. O butano, por ser mais pesado, entrará em combustão por último, atribuindo a chama de cor amarelada (ou alaranjada) (PETROBRAS, 2022).

Dentre suas demais características, a mais interessante é que o GLP pode ser facilmente levado do estado líquido para o estado gasoso, e vice-versa, o que facilita o seu transporte e

o seu armazenamento. Além disso, o produto também não é tóxico ou poluente, e não causa nenhum tipo de corrosão. No entanto, em casos de vazamentos ele pode causar asfixia por meio da redução da concentração de oxigênio no ar (PETROBRAS, 2022).

2.2.2 Principais aplicações

Devido às características detalhadas anteriormente que tornam o GLP único, ele acabou se popularizando mundialmente. Sua comercialização é realizada por meio de recipientes (botijões) que, segundo a Petrobras (2022), são produzidos com capacidades que variam de 2 Kg a 90 Kg do produto em sua forma liquefeita. Por se tratar de uma fonte de energia confiável e de alto poder calorífico, o GLP é utilizado principalmente para fins culinários, permitindo o cozimento de alimentos. Isso fez com que o produto fosse amplamente difundido nas residências, onde também é utilizado para outros fins domésticos, como a calefação da água, e o aquecimento de ambientes. A Figura 6 abaixo ilustra um quadro com os tipos mais comuns de recipientes do GLP comercializados.

Embalagem	Capacidade, kg	Aplicação
P-2	2	Camping e ambulantes
P-5	5	Camping e ambulantes
P-7	7	Uso Residencial
P-8	8	Uso Residencial
P-13	13	Uso Residencial
P-20	20	Empilhadeiras
P-45	45	Condomínios
P-90	90	Restaurantes

Figura 6 – Tipos de recipientes comercializados do GLP. Fonte: (PETROBRAS, 2022)

Contudo, sua aplicação não se restringe apenas às residências, visto que suas propriedades também são amplamente aproveitadas em outros setores. Na indústria, por exemplo, o GLP pode ser aplicado na área da siderurgia como meio para realizar cortes e soldas industriais, bem como pode ser utilizado na secagem de grãos e na esterilização de ambientes de criação de animais em instalações agropecuárias. Embora as possibilidades de se utilizar o GLP sejam amplas, vale ressaltar que existem determinadas exceções. No Brasil, por exemplo, não é permitida sua aplicação em saunas e piscinas, ou em motores de qualquer espécies.

Diante de sua vasta aplicabilidade, e do fato de que o GLP pode ser facilmente transportado em sua forma liquefeita, o produto também foi amplamente empregado nas regiões mais remotas, suprindo várias necessidades das papoluções mais afastadas dos centros urbanos.

2.2.3 Perigos do GLP

Conforme apresentado anteriormente, o GLP é um produto extremamente utilizado por possuir um alto potencial calorífico, que favorece principalmente sua aplicação no preparo de alimentos e faz com que ele esteja presente na maioria das residências. Por se tratar de um produto bastante difundido, é comum que aconteçam cenários em que as pessoas fiquem expostas ao vazamento desse gás no ambiente.

O problema principal é que, em uma situação de vazamento, pequenos "gatilhos" podem dar início a ignição do gás e causar grandes explosões. Isso acontece pois pequenas concentrações de volume de GLP no ambiente são suficientes para sustentar sua combustão, e portanto, uma centelha gerada pelo acionamento de um interruptor, ou ainda, uma pequena faísca produzida ao se conectar um equipamento elétrico em uma tomada, são suficientes para causar um acidente.

Diante disso, para os gases inflamáveis como o GLP, existe a chamada "faixa de explosividade", que determina o risco de combustão e de explosão desses gases de acordo com suas concentrações. No caso do GLP, basta antigir um teor de 2% do gás na atmosfera do ambiente para que sua combustão possa ser iniciada por meio de uma fonte de ignição. Esse valor de 2% se justifica devido a composição química do próprio gás, visto que ele é formado majoritariamente por propano e butano, que são inflamáveis a teores mínimos de 2,1% e 1,8%, respectivamente (PETROBRAS, 2022).

Por fim, além dos riscos de explosões, o GLP também pode provovocar "queimaduras por frio"(*frostbite*) caso seja colocado em contato direto com a pele FISPQ (2020). Também vale ressaltar que, segundo Leak (2018), a exposição das pessoas à grandes concentrações desse gás pode causar asfixia, visto que ele expulsa aos poucos o oxigênio do ambiente e vai se concentrando ao nível do solo.

2.3 *Hardware*

2.3.1 Microcontroladores

Os módulos microcontroladores são essenciais para a construção da parte física do sistema proposto, já que a lógica de programação para controlar e integrar os sensores e atuadores necessários é desenvolvida com esses módulos, permitindo o monitoramento do ambiente. Como existem diversos modelos disponíveis no mercado, foram selecionados e estudados dois módulos possíveis de serem utilizados nesse trabalho, considerando o baixo custo, a eficiência para cumprir os objetivos, e a facilidade de utilização. Os controladores são o ESP32 e o Arduino Uno R3.

2.3.1.1 ESP32

O ESP32 é um microcontrolador que foi desenvolvido pela empresa Espressif, no ano de 2016, que oferece diversos recursos e funcionalidades incluídas em seu *hardware*. Uma de suas principais vantagens é que ele possui suporte para conexão Wi-Fi e Bluetooth de forma integrada, e, portanto, não é necessário utilizar módulos adicionais para ter acesso a essas funcionalidades, o que o torna uma excelente escolha quando o desenvolvimento utiliza aplicações móveis ou de IoT (*Internet of Things*).

Conforme especificado por ESPRESSIF (2022b), o ESP32 possui um microprocessador modelo Xtensa 32-bit LX6, que possui dois núcleos que podem ser controlados separadamente e operam com *clocks* de até 240 MHz. Além disso, o módulo conta com memória RAM de 520KB, memória ROM de 448KB, e 30 portas GPIOS que podem integrar um rico conjunto de periféricos, como portas PWM, interfaces Seriais, e sensoriamento por toque capacitivo. Também vale ressaltar que o modelo conta com dois conversores do tipo analógico-digital que, juntos, somam 18 canais (ESPRESSIF, 2022b). A Figura 7 ilustra o diagrama de blocos do ESP32.

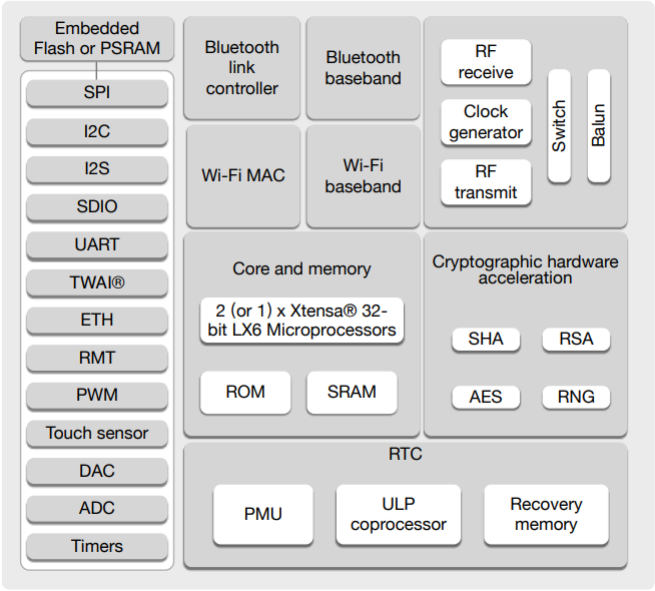

Figura 7 – Diagrama de blocos funcional do ESP32. Fonte: (ESPRESSIF, 2022a)

Contudo, a utilização direta do ESP32 considerando apenas o chip microcontrolador é complexa e não muito indicada para pessoas com pouco conhecimento técnico. Pensando nisso, diversas empresas, como a DOIT (*Doctors of Intelligence & Technology*), lançam placas de desenvolvimento para o ESP32 que oferecem todo um kit de funcionalidades que facilitam a sua utilização. Um exemplo é a placa ESP32 Devkit V1, ilustrada na Figura 8. Esse modelo de placa é facilmente encontrado no mercado e oferece recursos atrativos, como um regulador de tensão de 3,3V para o microcontrolador, um chip de interface Serial-USB junto a uma porta micro USB, que permitem a fácil conexão do módulo ao computador, além de botões de LOAD e RESET e outros módulos integrados (DOCS, 2022).

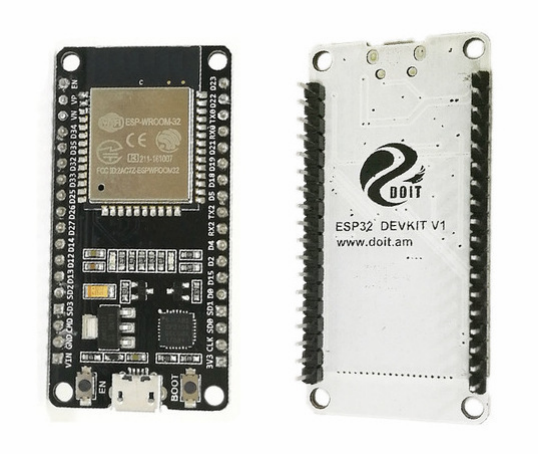

Figura 8 – placa DOIT ESP32 Devkit V1. Fonte: (DOCS, 2022)

Por fim, a Figura 9 apresenta o mapeamento dos pinos GPIO da placa ESP32 Devkit V1, que é uma placa de desenvolvimento para o módulo ESP-WROOM-32, exibindo as funcionalidades disponíveis para cada um deles.

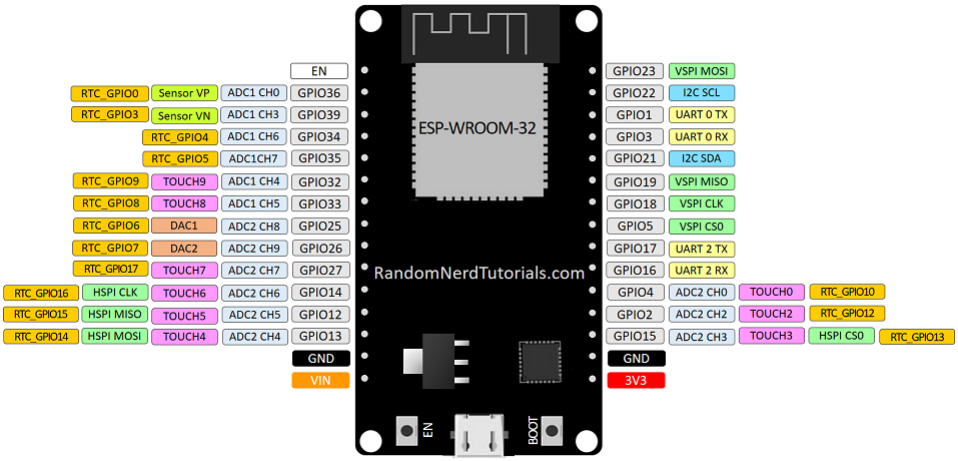

Figura 9 – Mapeamento dos pinos da placa DOIT ESP32 Devkit V1. Fonte: (TUTORIALS, 2022)

2.3.1.2 Arduino UNO R3

O Arduino é uma plataforma de prototipagem eletrônica *open-source* (código aberto), desenvolvido na Itália por Massimo Banzi e David Cuartielles no ano de 2005 (PORTUGAL, 2022). Além de apresentar um baixo custo, a plataforma pode ser facilmente programada e aplicada no controle de diversos sistemas iterativos, possuindo boa integração com uma vasta gama de módulos, sensores, e atuadores, o que a torna atrativa para projetos de pequeno e médio porte.

Existem diversos modelos de placas Arduino disponíveis no mercado, que oferecem diferentes recursos, funcionalidades, e capacidade do *hardware*. Dentre esses modelos temos

o Arduino UNO R3, que é composto por um microcontrolador ATmega328 de 8bits, 2KB de memória RAM, 1KB de memória EEPROM, e 32 KB de memória Flash. Além disso, o modelo também possui 6 pinos I/O analógicos, e 14 pinos I/O digitais dos quais 6 podem ser utilizados como entradas PWM (ARDUINO, 2022). A Figura 10 ilustra a placa Arduino UNO R3.

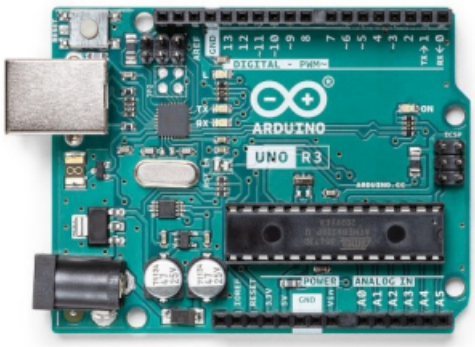

Figura 10 – Arduino UNO R3. Fonte: (ARDUINO, 2022)

Entretanto, o Arduino UNO R3 não oferece suporte para conexão Wi-Fi de forma integrada, como no ESP32. Logo, caso este modelo seja selecionado para desenvolver projetos com aplicações móveis, será necessário utilizar um módulo a parte junto ao Arduino capaz de forneçar o serviço de conexão Wi-Fi. Também existem outras diferenças de *hardware* entre o Arduino UNO R3 e o ESP32, que podem ser observadas por meio do quadro apresentado na Figura 11.

SPECS/BOARD	ESP32	ESP8266	ARDUINO UNO
Number of Cores	2	1	1
Architecture	32 Bit	32 Bit	8 Bit
CPU Frequency	160 MHz	80 MHz	16 MHz
WiFi	YES	YES	NO
BLUETOOTH	YES	NO	NO
RAM	512 KB	160 KB	2 KB
FLASH	16 MB	16 MB	32 KB
GPIO PINS	36	17	14
Busses	SPI, I2C, UART, I2S, CAN	SPI, I2C, UART, I2S	SPI, I2C, UART
ADC Pins	18	1	6
DAC Pins	2	0	0

Figura 11 – Quadro comparativo entre Arduino UNO R3, ESP32, e ESP8266. Fonte: (GHOSH, 2019)

2.3.2 Sensores

De acordo com Wendling (2010), os sensores são dispositivos sensíveis a alguma forma de energia presente no ambiente, podendo ser térmica, cinética, ou luminosa. Assim, o objetivo de sua utilização é conseguir mensurar e relacionar informações de alguma grandeza específica que queremos monitorar, como pressão, temperatura, aceleração, entre outras.

Sendo assim, sensores que são sensíveis a estimúlos específicos, como a concentração de gás no ambiente, a variação de temperatura, e a presença de chamas, foram essenciais para a construção do trabalho. Portanto, são apresentados a seguir os modelos de sensores disponíveis no mercado que foram estudados para o trabalho, considerando os estímulos específicos que são monitorados.

2.3.2.1 Sensores de gás MQ

Conforme apresentado por Alves e Mendonça (2018), os sensores da série MQ (*Quality Monitoring*) são dispositivos sensíveis a presença de gases no ambiente, tornando possível a detecção dos mesmos.

Existem diferentes tipos de sensores da linha MQ disponíveis no mercado, onde cada modelo é desenvolvido especialmente para a detecção de determinados tipos de gases. Logo, a escolha do modelo mais adequado para cada caso está diretamene relacionada ao tipo de gás que se deseja detectar. A Figura 12 ilustra, por meio de um quadro, diferentes modelos de sensores da linha MQ e os respectivos gases aos quais eles são sensíveis.

TIPO DO SENSOR	SENSÍVEL PARA
MQ-02	Metano, butano, GLP e fumaça
MQ-03	Álcool, Etanol e Fumaça
MQ-04	Metano e Gás CNG
MQ-05	Gás natural e GLP
MQ-06	GLP e gás butano
MQ-07	Monóxido de carbono
MQ-08	Gás hidrogênio
MQ-09	Monóxido de carbono e gases inflamáveis

Figura 12 – Quadro com os sensores da série MQ e os gases os quais apresentam sensibilidade. Fonte: (ALVES; MENDONÇA, 2018)

Dentre os modelos apresentados, os sensores MQ-6 e MQ-2 são os mais adequados para utilização neste trabalho, visto que o gás a ser monitorado é o GLP. Os dois modelos são estruturalmente similares, mas se diferem quanto a composição do elemento sensível que permite

a detecção de gases no ambiente. Dentre eles, o MQ-6 é o modelo que possui maior sensitividade ao gás GLP e ao gás butano, motivo pelo qual ele foi selecionado para estudos.

2.3.2.2 Sensor de Gás MQ-6

O sensor MQ-6 é um dispositivo que apresenta sensibilidade aos gases GLP, Isobutano, Propano, e Gás Natural liquefeito. Logo, ele pode ser aplicado para detectar a presença desses gases no ambiente, bem como para mensurar suas concentrações.

De acordo com Alves e Mendonça (2018), o sensor é composto por um elemento eletroquímico sensível ao monóxido de carbono em ampla temperatura, além de possuir um resistor de proteção e um resistor variável (potenciômetro). O módulo é de fácil instalação, possui uma vida útil longa, e fornece sinais que podem ser lidos por um microcontrolador por meio de duas saídas, onde uma é analógica, e a outra é digital. O sensor é apresentado na Figura 13.

Figura 13 – Sensor MQ-6 Fonte: (PINTO, 2016)

O MQ-6 precisa ser alimentado com uma tensão de 5V para operar corretamente, e é sensível a níveis de concentrações de gases dentro da faixa de 200ppm a 10.000ppm (ALVES; MENDONÇA, 2018). É possível ajustar a sensibilidade do sensor por meio do seu potenciômetro, e as concentrações dos gases são obtidas por meio de um tratamento nos dados lidos de sua saída analógica. Vale ressaltar que, na medida em que a concentração de gás no ambiente varia, a queda de tensão na resistência interna do sensor também varia, sendo esse o valor fornecido à sua saída analógica.

Conforme relatado por Pinto (2016), o elemento eletroquímico do sensor precisa atingir uma determinada temperatura de operação para funcionar corretamente, podendo atingir valores por volta de 60 graus Celsius. Não é preciso utilizar nenhum componente extra junto ao módulo para que a temperatura de operação seja atingida, visto que o pino de alimentação de 5V do sensor também atua como fonte de aquecimento para o mesmo. Contudo, é preciso aguardar

cerca de 5 minutos após ligar o sensor junto ao microcontrolador, para que a temperatura de trabalho seja atingida e as medições se tornem estáveis.

O modelo esquemático do sensor MQ-6 está ilustrado pela Figura 14, onde é apresentado o elemento sensível e seus canais de conexão, bem como o circuito elétrico equivalente do módulo como um todo.

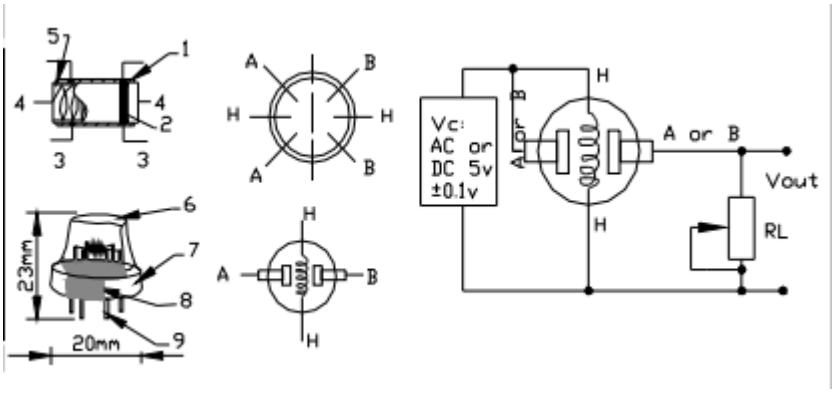

Figura 14 – Modelo esquemático do sensor MQ-6 Fonte: Adaptado de (SENSORS, 2022)

2.3.2.3 Sensor LM35

O LM35 é um sensor de temperatura capaz de fornecer uma resposta de tensão linearmente relativa a variação de temperatura em graus Celsius no ambiente, de forma precisa. Esse sensor apresenta a vantagem de não necessitar de nenhum tipo de calibração para se obter exatidão em suas medições. Como ele não é nativamente calibrado na escala Kelvin, também não é necessário realizar transformações para se obter a temperatura em graus célsius. Além da precisão, da linearidade, e da calibração direta na escala Celsius, o sensor também possui um baixo custo.

O sensor é apresentado na Figura 14, onde é ilustrado o seu pino de alimentação, de aterramento, e de saída digital.

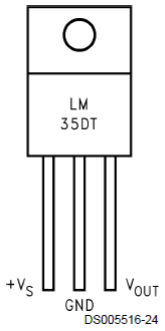

Figura 15 – Sensor LM35 Fonte: Adaptado de (INSTRUMENTS, 2000)

Em relação a suas características, o LM35 pode ser alimentado com uma tensão de entrada de 4V à 40V, apresenta uma variação de 10mV por grau Celsius, e possui uma precisão de $\pm$ 0,25°C para variações de temperatura entre -55°C à 150°C (INSTRUMENTS, 2000).

2.3.2.4 Termistor

Os termistores são resistores termicamente sensíveis, que sofrem alteração em sua resistência elétrica a partir da variação da temperatura no meio em que estão inseridos. Eles são construídos a partir de materiais semicondutores cerâmicos, e possuem como matérias-primas misturas sintetizadas de óxidos, sulfetos e silicatos (ASSIS, 2018). Além disso, os termistores possuem duas classificações distintas, definidas conforme o tipo de material semicondutor utilizado na fabricação.

Os termistores fabricados com um material semicondutor positivo são classificados como PTC (*Positive Temperature Confficient*), e possuem a relação resistência-temperatura aproximada por um polinômio de grau 1. Já os termistores produzidos a partir de um material semicondutor negativo são classificados como NTC (*Negative Temperature Confficient*), e apresentam uma relação aproximadamente linear entre a variação da temperatura e da resistência elétrica para certas aplicações, como na medição de temperatura (ASSIS, 2018).

Contudo, os termistores apresentam a desvantagem de necessitarem de calibrações e tratamentos de dados para a obtenção de aferições de temperatura na escala Celsius.

2.3.2.5 Sensor de Chamas

Os sensores de chamas são componentes sensíveis a uma determinada faixa de comprimentos de onda do spectro da luz visível, compreendida entre 760nm a 1100nm. Sendo assim, como essa faixa engloba os comprimentos de onda característicos do spectro do fogo, esses sensores são capazes de detectar a presença de chamas no ambiente em que estão inseridos.

Esses sensores podem operar com tensões entre 3,3V a 5V, e possuem um angulo de detecção compreendido entre 0° a 60° (FLOP, 2022b). Além disso, conforme relatado por Joy-it (2017), esse tipo de sensor é composto basicamente por 3 componentes principais em sua placa integrada. O primeiro deles é um foto transistor responsável por detectar a incidência das ondas do spectro da luz, enviando um sinal analógico ao segundo componente. Este, por sua vez, consiste em um amplificador de sinal que amplifica os valores enviados de acordo com o valor da resistência ajustada no potenciômetro, e os envia para a saída analógica do módulo. Por fim, o terceiro módulo consiste em um comparador LM393 que envia sinais digitais para a saída digital do módulo, e para o LED indicativo desta respectiva saída.

O sensor de chamas é ilustrado por meio da Figura 16, onde é apresentado os principais componentes do módulo e os pinos de saída analógica e digital, representados com as siglas *A*0

e $D0$, respectivamente.

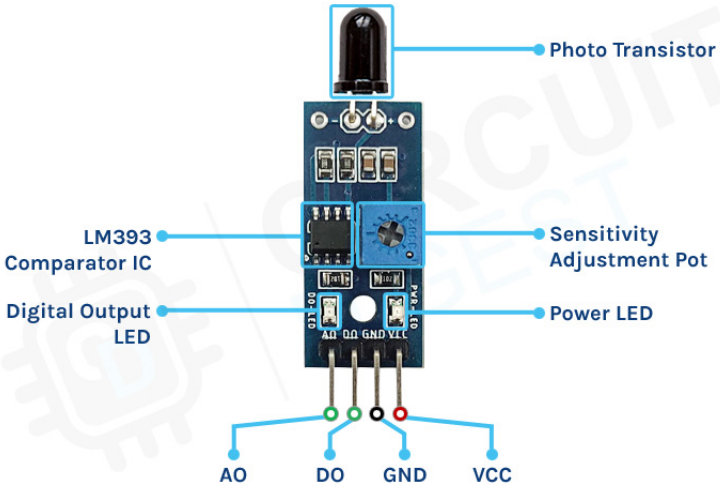

Figura 16 – Sensor de chamas. Fonte: (JOSEPH, 2022)

Por fim, vale ressaltar que também existem variações deste tipo de sensor que não dispõem do pino de saída analógica, isto é, possuem apenas a saída digital e, portanto, são utilizados apenas para a detecção da presença de chamas.

2.3.3 Módulo Relé

Conforme relatado por Santos (2018), os *shields* (escudos) são pequenas placas de circuito que contém outros dispositivos integrados, como os displays de LCD, ou os receptores GPS. Esses dispositivos são projetados para serem facilmente acoplados aos microcontroladores, fornecendo funcionalidades extras para as aplicações.

Nesse aspecto, um *shields* relé (ou módulo relé) trata-se de um dispositivo eletromecânico projetado em uma placa integrada, que atua como um dispositivo interruptor, capaz de alternar entre o estado "ligado" e "desligado", de acordo com a presença ou não de energia elétrica em seu terminal de ativação. Logo, esses módulos podem ser utilizados junto aos microcontroladores para realizar o acionamento de cargas, como lâmpadas, motores, ou outros dispositivos de corrente alternada, sem que o circuito principal seja alterado (SANTOS, 2018).

O componente relé presente no módulo é composto por: um eletroímã de fio de cobre, uma armadura de ferro móvel, um conjunto de contatos, uma mola de rearme, e os terminais de conexão (SANTOS, 2018). Sendo assim, os módulos relés são construídos a partir da utilização de um ou mais relés dispostos na mesma placa integrada, com componentes extras que auxiliam na utilização do módulo, como os LEDs indicativos de acionamento dos relés. A Figura 17 ilustra um módulo relé de um canal.

Por fim, vale ressaltar que os relés são classificados quanto ao seu tipo de contato elétrico, que pode ser NA (normalmente abertos) ou NF (normalmente fechados). Os relés do tipo NA

Figura 17 – Módulo relé de 5V e 1 canal. Fonte: (FLOP, 2022a)

mantêm o seu contato "aberto" enquanto estiverem desenergizados, fechando-o no momento em que forem energizados. Os relés do tipo NF, por sua vez, operam de forma contrária aos do tipo NA, abrindo seu contato elétrico no momento em que são energizados.

2.3.4 Buzzer

O *buzzer* é um pequeno componente eletrônico de baixo custo capaz de gerar sinais sonoros em diferentes frequências, que é comumente utilizado junto ao *hardware* das aplicações que necessitam de um meio para fornecer sinais sonoros aos usuários.

Conforme relatado por ROBÓTICA (2022), o *buzzer* dispõe de um circuito de acionamento com dois pinos externos para conectá-lo à alimentação e ao aterramento, e possui um pequeno transdutor piezoelétrico em seu interior, composto por um disco central de cerâmica envolvido por outro disco vibratório de metal. Esses componentes são acoplados a um invólucro plástico, dando forma ao *buzzer*, e quando o transdutor interno recebe uma determinada onda de tensão, ele entra em vibração e produz os sinais sonoros audíveis. A Figura 18 ilustra a composição de um *buzzer*.

Além disso, existem dois tipos de *buzzer* disponíveis no mercado: o *buzzer* ativo, e o *buzzer* passivo. Ambos se diferem quanto ao modo de operação, e encontram-se ilustrados por meio da Figura 19.

O *buzzer* ativo possui um oscilador interno próprio, responsável por fornecer as ondas de tensão que resultam nas vibrações do transdutor, produzindo o sinal sonoro (ROBÓTICA, 2022). Sendo assim, é preciso apenas energizar esse componente para que o mesmo produza um sinal sonoro audível, o que o torna ideal para aplicações onde o sinal sonoro é necessário apenas para realizar alguma sinalização.

O *buzzer* passivo, por sua vez, não possui um oscilador interno responsável por produzir

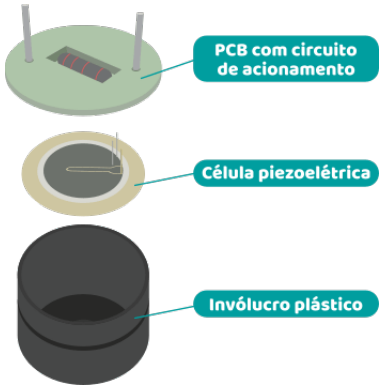

Figura 18 – Composição de um *buzzer*. Fonte: (MAKERS, 2022)

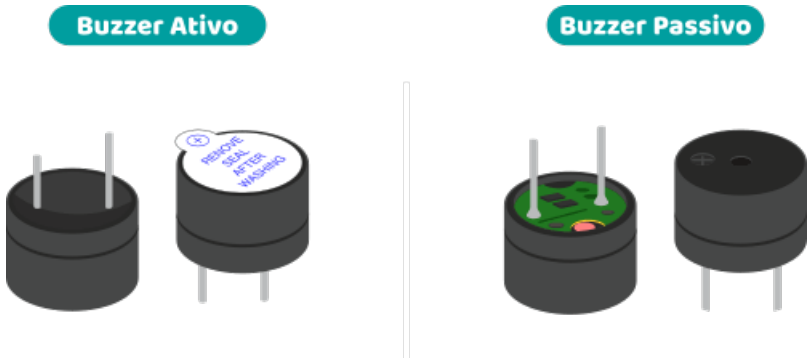

Figura 19 – Tipos de *buzzer*. Fonte: (MAKERS, 2022)

as ondas de tensão, e portanto, é necessário fornecer um sinal de onda quadrada em diferentes frequências para que ele seja acionado (ROBÓTICA, 2022). Sendo assim, sua utilização se torna mais complexa se comparada ao *buzzer* ativo, mas é possível produzir diferentes sinais sonoros com esse tipo de componente, tornando-o ideal para aplicações onde se deseja reproduzir pequenas melodias.

2.3.5 LED

O diodo emissor de luz, também conhecido como LED (*light-emitting diode*), é um componente eletrônico capaz de converter a energia elétrica em energia luminosa. Conforme relatado por Takahashi (2022), a grande diferença entre esses componentes e as fontes de luz convencionais, como as lâmpadas, é que eles não convertem a energia elétrica em calor e depois em luz. Isto é, a conversão entre a energia elétrica e a luminosa ocorre de forma direta, por meio de um fenômeno denominado eletroluminescência.

Conforme ilustrado pela Figura 20, cada LED é construído a partir de um terminal denominado anodo, um outro terminal denominado catodo, e um material semicondutor. Além disso, por se tratar de um diodo, a passagem ou não de corrente elétrica entre seus terminais depende diretamente da polarização do LED.

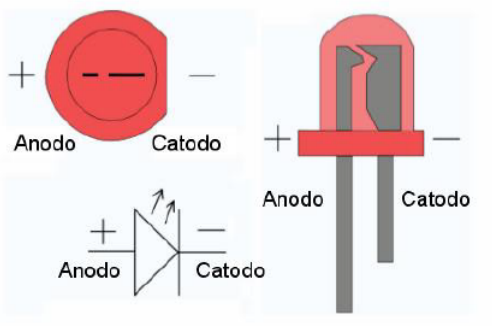

Figura 20 – Modelo esquemático de um LED. Fonte: (PINTO, 2016)

Para que a emissão de luz seja possível, o material semicondutor do LED é construído a partir da combinação de um semicondutor do tipo-P com um semicondutor do tipo-N, além de uma região de junção entre eles. Portanto, como os semicondutores do tipo-P possuem alta concentração de lacunas, e os do tipo-N possuem alta concentração de elétrons, ao aplicarmos uma tensão direta no LED alguns elétrons ultrapassam a barreira de junção entre esses materiais, e são recombinados pelas lacunas que se encontram próximas à região de junção, o que resulta na geração dos fótons de luz (TAKAHASHI, 2022).

A Figura 21(a) ilustra a injeção de um elétron da região do tipo-N para a região do tipo-P, por meio da junção entre esses materiais. A Figura 21(b), por sua vez, ilustra a recombinação desse elétron que resulta na emissão do fóton de luz:

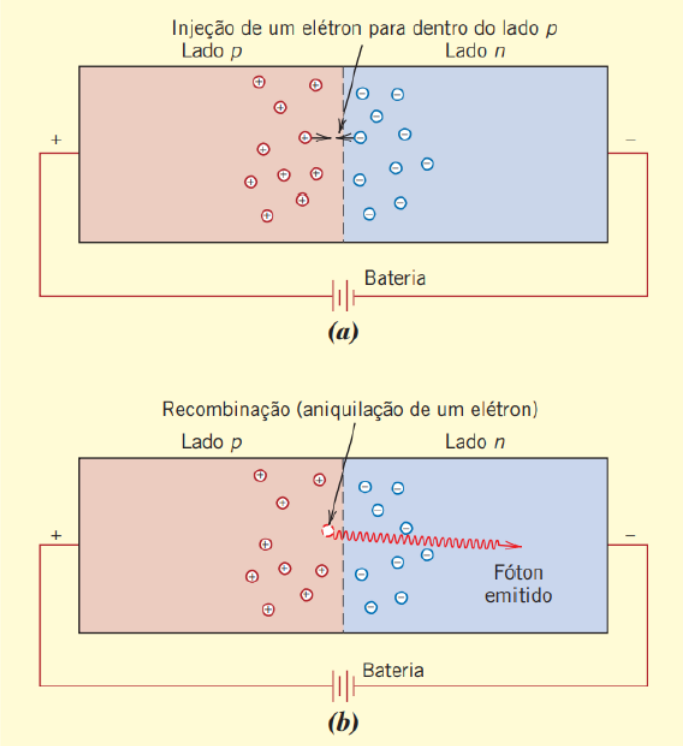

Figura 21 – Estrutura do material semicondutor de um LED. Fonte: (TAKAHASHI, 2022)

Por fim, vale ressaltar que por se tratar de um componente de baixo custo, e de simples instalação, os LEDs são comumente utilizados em aplicações eletrônicas como meio para emitir sinalizações visuais.

2.3.6 *Softwares*

2.3.6.1 IDE do Arduino

A IDE (*integrated development environment*) do Arduino é um ambiente de desenvolvimento integrado que permite a criação de toda a programação do *hardware* das aplicações que utilizam o Arduino. Portanto, é por meio da IDE que o usuário conseguirá configurar as entradas e saídas utilizadas, realizar todo o tratamento dos dados que serão lidos dos sensores, e programar as ações que o Arduino realizará diante das necessidades de cada projeto.

Conforme relatado por Souza (2013), a IDE do Arduino conta com uma linguagem própria, baseada na linguagem C e C ++, e possibilita a detecção de erros na sintaxe dos códigos desenvolvidos. A IDE também apresenta um alto grau de abstração, o que permite a utilização dos microcontroladores mesmo quando o usuário não possui conhecimentos avançados sobre os mesmos, ou sobre o funcionamento dos seus registradores internos.

Além disso, a IDE também disponibiliza recursos e ferramentas que auxiliam na construção dos códigos, permite a inclusão de diferentes bibliotecas de funcionalidades, e realiza a tradução do código criado para uma linguagem que pode ser interpretada pelos microcontroladores de forma automática.

Por fim, vale ressaltar que é possível utilizar a IDE do Arduino como meio para programar microcontroladores que não sejam da família Arduino, como por exemplo o ESP32 da ESPRESSIF, o que torna esta IDE uma opção completa e ideal para programar diferentes tipos de microcontroladores.

2.3.6.2 *Kodular*

O *Kodular* é uma plataforma online e gratuita, que permite a criação de aplicativos *mobile* por meio da linguagem de programação *Scratch* (popularmente conhecida como "linguagem de blocos"). De acordo com Merçon (2022), o *Kodular* foi criado como uma evolução da plataforma Makeroid, que é baseada na plataforma App Inventor desenvolvida pelo MIT (*Massachusetts Institute of Technology*). Embora ambas as plataformas utilizem da "programação de blocos", o *Kodular* dispõe de mais elementos para incluir nos aplicativos, e por essa razão, foi selecionado para uso neste trabalho.

Por utilizar a linguagem *Scratch* , o *Kodular* não demanda conhecimentos específicos em linguagem de programação para que possa ser utilizado. Nesse contexto, toda a lógica de

programação, bem como o *layout* das telas, são construídos baseados no sistema *drag and drop*, onde o usuário seleciona e "arrasta" os elementos funcionais disponíveis para moldar a aplicação, como os ícones, os botões, e os blocos de funções.

Diante disso, o *Kodular* dispõe de duas abas de desenvolvimento distintas. A primeira é a aba *Designer*, ilustrada por meio da Figura 22, onde estão disponíveis os componentes visuais necessários para a criação do *layout* das telas.

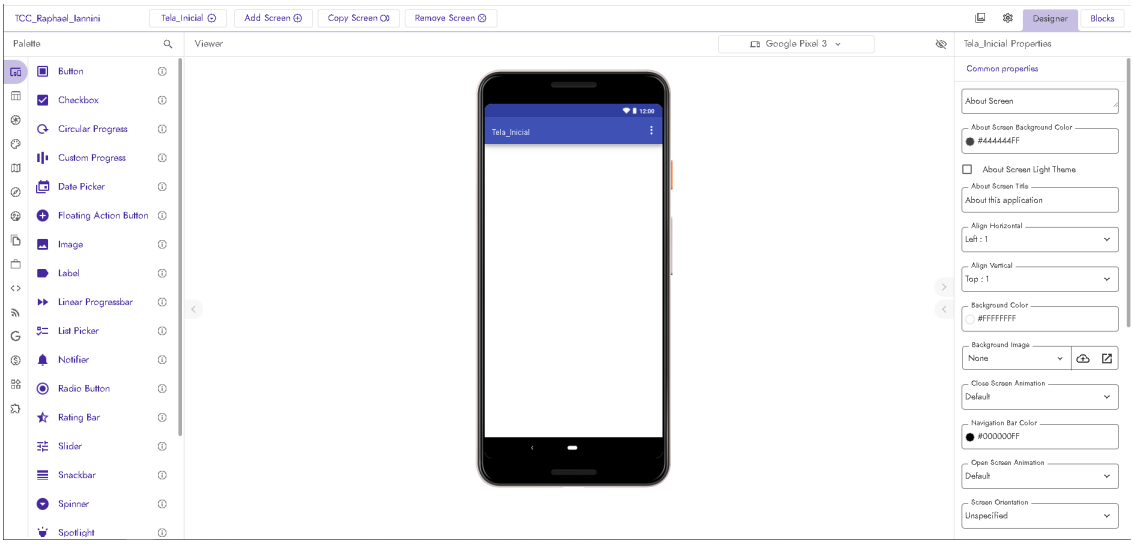

Figura 22 – Tela exibida na aba *Designer*. Fonte: Autoria própria.

A segunda aba é denominada *Blocks*, onde são criadas as funções responsáveis por realizar as operações das telas do aplicativo, isto é, todo o *Backend* é construído a partir desta aba, por meio da utilização de blocos funcionais, conforme ilustrado por meio da Figura 23.

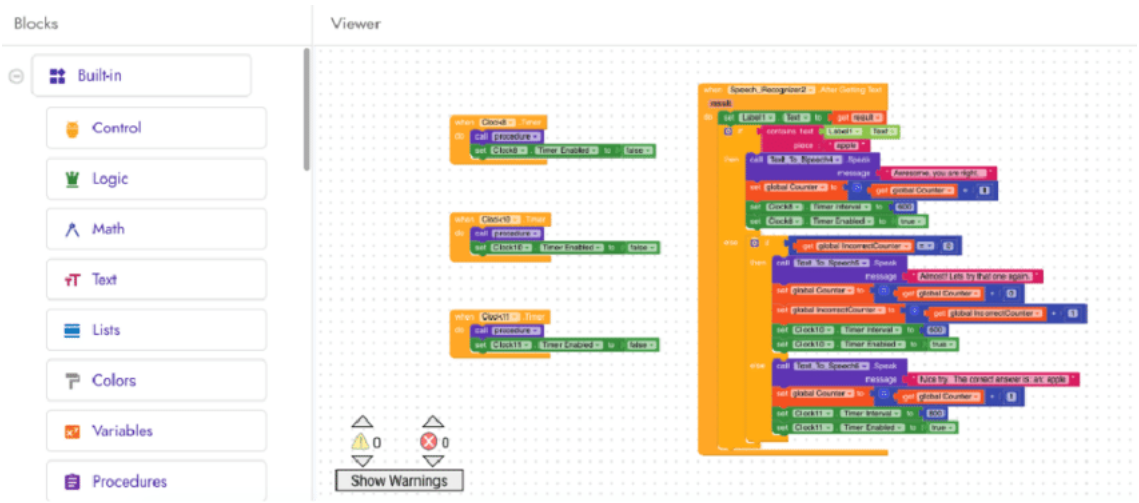

Figura 23 – Tela exibida na aba *Blocks*. Fonte: (DJASSEMI, 2020)

Por fim, o *Kodular* também dispõe de recursos que permitem a comunicação das aplicações com dispositivos externos, como os microcontroladores, além de permitir a troca de

informações com bancos de dados, o que amplifica ainda mais as possibilidades de utilização da plataforma.

2.4 Banco de Dados

Neste trabalho o banco de dados é utilizado para possibilitar a comunicação entre o *hardware*, desenvolvido para o monitoramento do ambiente, e o aplicativo mobile, que permite acompanhar as informações de forma remota.

Para tal, o *hardware* envia as informações mensuradas para o banco de dados, de forma periódica. A partir daí, o banco de dados será alterado e o aplicativo mobile realizará a leitura das informações registradas, atualizando as informações na tela em tempo real, e emitindo mensagens de alerta caso alguma situação de risco ocorra no ambiente.

Vale ressaltar que a utilização do banco de dados como um intermédio entre a troca de informações do *hardware* com o aplicativo elimina a necessidade de configurar ambos na mesma rede de comunicação. Sendo assim, o usuário poderá monitorar o ambiente por meio do aplicativo mesmo quando não estiver próximo ao local da instalação do *hardware*.

2.4.1 *Firebase*

O *Firebase* é uma plataforma digital desenvolvida pela Google, que oferece diversos recursos para o desenvolvimento de aplicações *mobile* e *web*. Nesse contexto, o *Firebase* é considerado um *Backend as a Service* (BaaS), ou seja, uma ferramenta capaz de oferecer todo um conjunto de serviços essenciais para o funcionamento interno dos *softwares*, tais como o banco de dados, o envio e o recebimento de informações, e os serviços de hospedagem (ONLINE, 2021).

A grande vantagem do *Firebase* é que ele pode ser utilizado de forma gratuita, além de permitir a configuração do banco de dados de forma simples, sem exigir conhecimentos técnicos aprofundados dos usuários. Contudo, os serviços gratuitos possuem algumas limitações, como, por exemplo, o número máximo de acessos simultâneos permitido ao banco de dados.

Mesmo diante das limitações, o banco de dados disponibilizado pela plataforma é atualizado em tempo real, o que garante uma boa sincronia na troca de informações entre o *hardware* e o aplicativo *mobile*. Além disso, as funcionalidades gratuitas oferecidas pelo *Firebase* suprem as necessidades deste trabalho, o que o torna ideal para o desenvolvimento do sistema proposto.

3 Desenvolvimento

Neste capítulo são apresentadas as etapas que foram necessárias para desenvolver o sistema de mitigação de incêndios e de vazamentos de gás, para o ambiente residencial, considerando a facilidade de instalação e a possibilidade de monitorar o ambiente de forma remota por meio de um aplicativo. O custo total do projeto foi levado em consideração nesta etapa, onde os componentes com preços mais acessíveis que atendiam aos requisitos do trabalho, bem como as plataformas de desenvolvimento gratuitas, foram priorizados.

Diante disso, foi desenvolvido o sistema denominado *FireSafe*, composto por um *hardware* físico controlado por meio da placa microcontroladora ESP32-Devkit V1, um aplicativo mobile desenvolvido na plataforma *Kodular*, e um banco de dados criado no *Firebase* para intermediar a troca de informações entre o módulo físico e o aplicativo.

A plataforma *Kodular* foi escolhida por permitir o desenvolvimento do aplicativo do sistema por meio da programação de blocos, que é mais intuitiva e de fácil entendimento. Além disso, a plataforma também permitiu desenvolver a comunicação entre o aplicativo e o banco de dados no *Firebase* de forma descomplicada, por meio das ferramentas disponíveis.

A placa microcontroladora ESP32-Devkit V1 foi selecionada para o desenvolvimento do *hardware* do trabalho por ser mais robusta se comparada ao Arduino UNO, o que permitiu desenvolver um código, por meio da IDE do *Arduino*, que utiliza os dois núcleos disponíveis no microprocessador da placa, bem como conceitos de programação em tempo real (RTOS) que foram essenciais para o funcionamento do sistema. Além disso, o microcontrolador ESP32-WROOM-32 possui tecnologia Wi-Fi integrada, o que descartou a necessidade de adquirir um módulo Wi-Fi à parte para realizar a comunicação com o banco de dados via rede.

Por fim, o fluxograma completo do sistema desenvolvido é ilustrado por meio da Figura 24, e todos os passos necessários para a construção de cada uma das partes do mesmo serão detalhados a seguir.

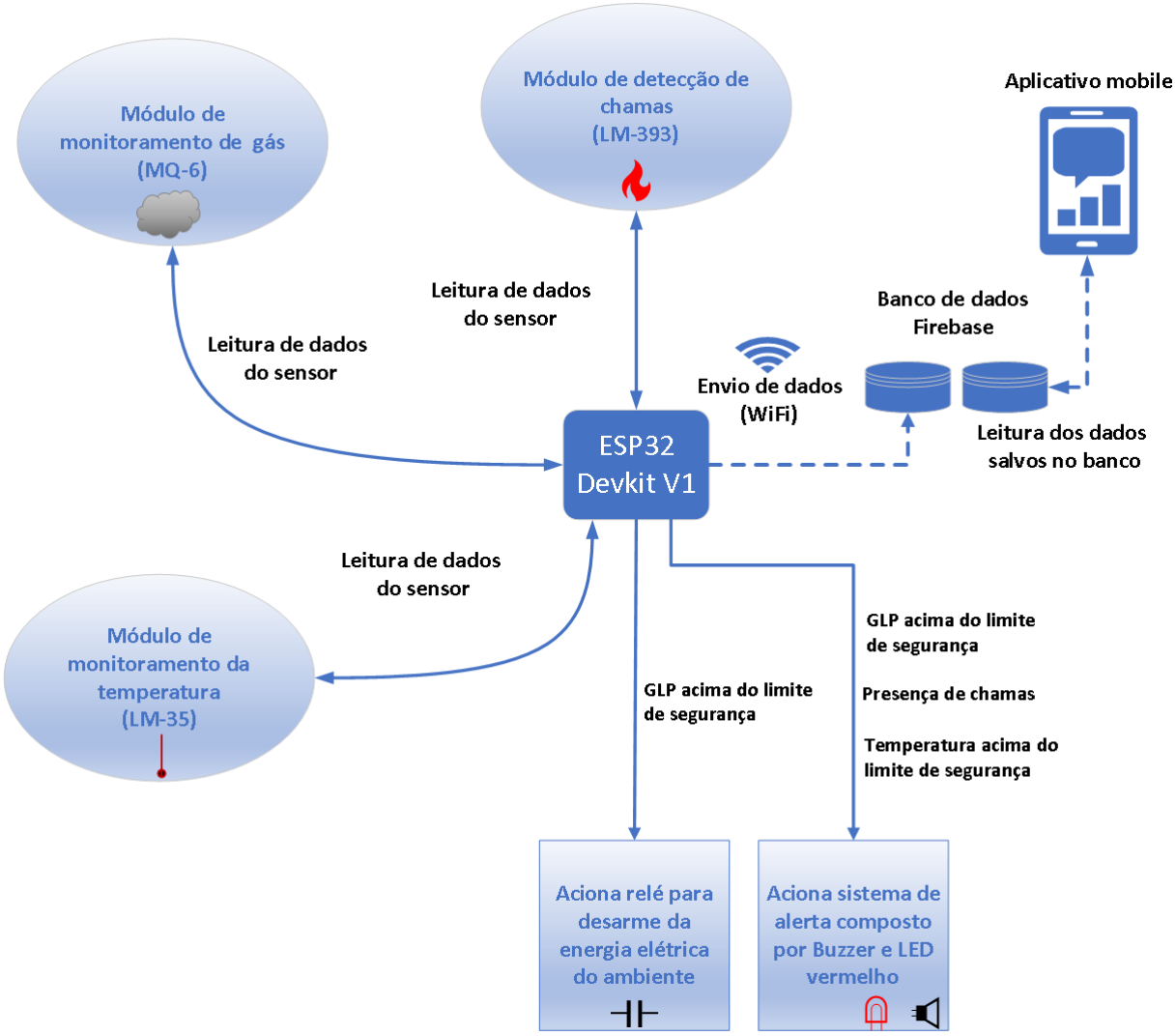

Figura 24 – Fluxograma completo do *FireSafe*. Fonte: Autoria Própria

3.1 *Hardware* do *FireSafe*

Para a construção do *hardware* do sistema *FireSafe* foi necessário primeiramente listar e adquirir os componentes e sensores essenciais para a concretização de todos os requisitos do sistema. A partir daí, para facilitar o entendimento e a organização do desenvolvimento, toda a parte física foi dividida nos seguintes tópicos:

- Módulo de detecção de chamas.
- Módulo de monitoramento de gás.
- Módulo de monitoramento da temperatura.
- Módulo de alerta e mitigação de incidentes.

- Sinalização de erros e *status* de conexão.

Para cada um dos tópicos foram selecionados os sensores e componentes adequados, realizada a montagem física, e desenvolvida a lógica de programação necessária para o funcionamento e a integração com a placa microcontroladora. Também foi necessário realizar tratamentos de dados em algumas das etapas, para permitir o funcionamento correto dos sensores, como no caso do módulo de monitoramento gás.

Vale ressaltar que uma fonte de *protoboard* foi adquirida para realizar a alimentação elétrica do sistema. Essa escolha foi necessária para facilitar a montagem física, uma vez que alguns dos sensores operam com uma alimentação de 5 V, enquanto que outros operam com uma alimentação de 3,3 V (similar ao ESP32). Além disso, a fonte também é projetada para ser facilmente acoplada em placas ilhadas, facilitando a montagem final do *hardware*.

Por fim, os tópicos desenvolvidos serão apresentados a seguir, bem como a montagem física do *hardware* na *protoboard*, e o detalhamento das estratégias de programação utilizadas no sistema.

3.1.1 Módulo de detecção de chamas

Para permitir a detecção de chamas no ambiente, foi utilizado um módulo sensor infravermelho genérico de três canais, que é sensível à faixa de comprimento de onda emitida pelo fogo. Em relação aos seus canais (pinos), dois deles são destinados a alimentação elétrica do módulo sensor (V_{CC} e GND), e o terceiro (D0) fornece uma saída digital que varia de acordo com a incidência de luz infravermelha percebida pelo sensor.

O sensor utilizado é baseado no módulo sensor KY-037, e portanto, utiliza a "lógica invertida" em sua saída digital. Quando o sensor não está exposto ao fogo, o pino D0 apresenta uma saída de nível lógico alto, e quando o sensor é exposto ao fogo, a detecção da luz infravermelha faz com que o sinal da saída digital vá para nível lógico baixo.

O sensor foi alimentado diretamente com uma tensão de 3,3V, e o pino de saída D0 foi conectado à porta GPIO 13 do ESP32, o que permitiu monitorar a presença de fogo no ambiente a partir dos valores lidos.

3.1.2 Módulo de monitoramento de gás

Para a utilização do sensor MQ-6 foi necessário primeiramente realizar um pré-aquecimento (*pre-heat*) do mesmo, por um período de 24 horas. Esta etapa consistiu em alimentar o sensor com a tensão de 5 V, e mantê-lo energizado e exposto ao ambiente, o que permitiu alcançar a sua estabilização e garantiu uma maior fidelidade em seu desempenho se comparado aos dados descritos no *datasheet*. Como o objetivo deste módulo é monitorar as concentrações de gás GLP

no ambiente, detectando situações de vazamento que ofereçam risco de explosões ou combustões, apenas a saída analógica A0 do sensor foi utilizada em conjunto com a porta GPIO 35 do ESP32.

Após a etapa de *pre-heat*, foi realizada uma análise na curva característica da sensibilidade do sensor, para definir como interpretar e tratar os dados lidos a partir da porta analógica A0. Neste contexto, quando o sensor é submetido à concentrações de gás que variam em parte por milhão (ppm), a sua resistência de carga interna também varia, fazendo com que a queda de tensão na porta analógica A0 também sofra variações.

A curva que ilustra a sensibilidade característica do sensor MQ-6 ao gás GLP é representada por meio da Figura 25. A partir de uma análise da curva foi possível observar uma relação aproximadamente linear entre a variação da resistência de carga interna e a concentração de gás em ppm, e portanto, a queda de tensão fornecida no pino A0 também possui o mesmo comportamento.

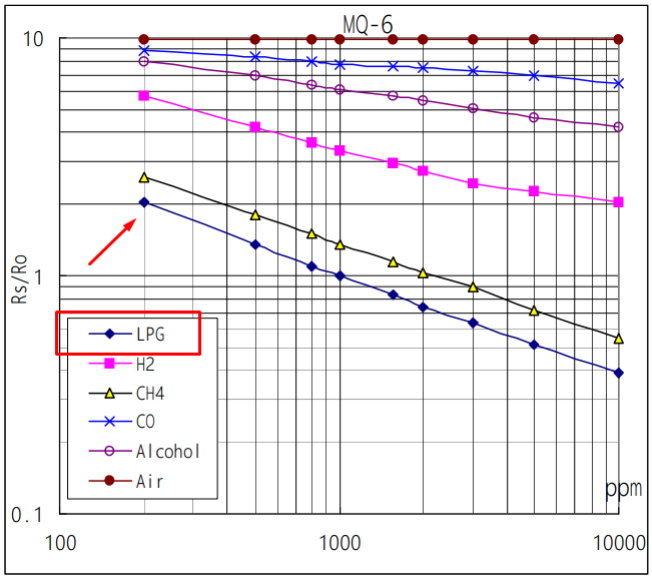

Figura 25 – Curvas da sensibilidade característica do MQ-6. Fonte: Adaptada de Today (2014)

Sendo assim, foi possível utilizar diretamente os valores da queda de tensão do pino A0 para estimar a porcentagem de concentração do gás GLP no ambiente, uma vez que, para concretrizar os objetivos do trabalho, não é necessário transformar os dados de tensão lidos em valores de concentração em ppm do gás.

3.1.2.1 Tratamento de dados

Para interpretar os dados fornecidos pelo sensor por meio do pino A0, foi necessário primeiramente definir um circuito divisor de tensão para interligar o pino com a porta GPIO do ESP32. Conforme apresentado por Today (2014) , quando o sensor está completamente imerso ao gás GLP, ele apresenta uma queda de tensão de cerca de 4,62 V em sua resistência interna.

Portanto, os valores que podem ser obtidos no pino A0 variam de 0 V à 4,62 V, extrapolando a faixa de operação das portas GPIO do ESP32, que trabalham com tensões de 0 V à 3,3 V.

Para ajustar as faixas de tensão, foi definido um divisor de tensão para transformar os valores de 0 V à 5 V para uma faixa de 0 V à 3,3 V. Portanto, para o cenário onde o sensor apresenta uma queda de tensão de cerca de 4,62 V na escala de 5 V, esse valor será reduzido para 3,05 V por meio do divisor de tensão. O modelo esquemático do circuito é representado por meio da Figura 26

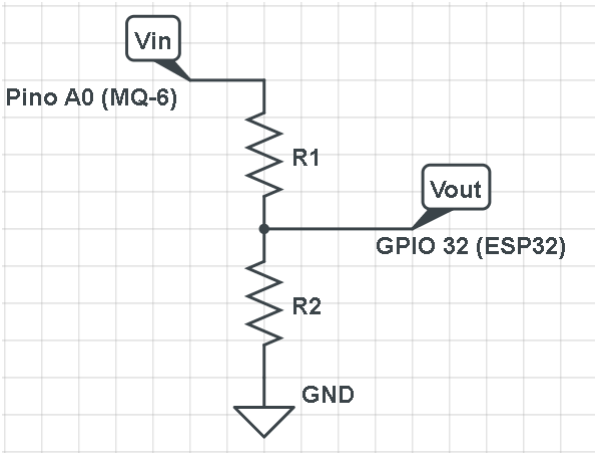

Figura 26 – Circuito elétrico do divisor de tensão. Fonte: Autoria própria

A equação 3.1 relaciona a tensão de entrada do divisor de tensão (V_{in}), com a tensão de saída (V_{out}) que é lida na porta GPIO.

$$V_{out} = V_{in} * \frac{R2}{R1 + R2} \tag{3.1}$$

Para determinar as resistências do circuito, as faixas de valores comerciais disponíveis foram consideradas. Nesse contexto, a resistência R1 foi definida como 1KΩ, e então, a resistência R2 foi calculada por meio da equação 3.2

$$\begin{aligned} R2 &= V_{out} * \frac{R1}{V_{in} - V_{out}} \\ R2 &= 3,3 * \frac{1K}{5 - 3,3} \\ R2 &= 1941,18\Omega \end{aligned} \tag{3.2}$$

Portanto, o valor de 2KΩ foi adotado para a resistência R2, por ser a faixa comercial mais próxima do valor calculado.

Após concretizar a ligação física por meio do divisor de tensão, foi realizado o tratamento de dados para interpretar os valores lidos com o conversor analógico-digital do ESP32. Este conversor possui 12 bits, e portanto, ele interpreta os valores em uma faixa numérica de 0 a 4095.

Considerando que a queda de tensão máxima lida pela GPIO é de 3,05V, o valor máximo que será interpretado no conversor é de 3784,77. Como apenas números inteiros são interpretados, o valor foi definido como 3785. Sendo assim, foi possível estabelecer uma relação entre os valores do conversor e a porcentagem do gás GLP no ambiente, conforme representado na Figura 27.

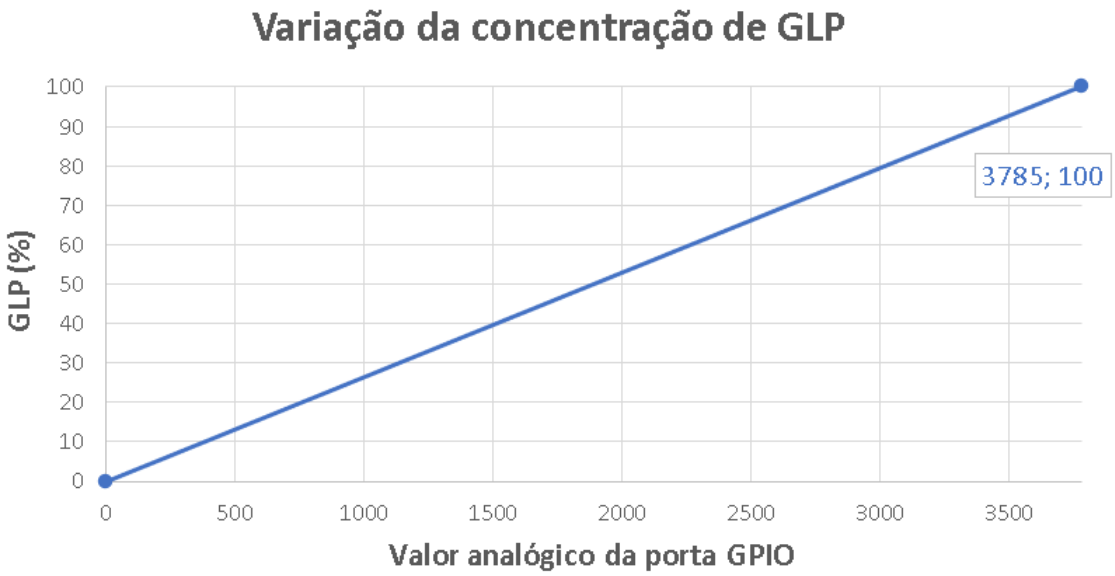

Figura 27 – Variação da concentração de GLP. Fonte: Autoria própria

Os valores analógicos lidos são convertidos em porcentagem de gás GLP por meio da equação 3.3.

$$GLP_{\%} = \frac{100 * ValorAnalgico_{GLP}}{3785} \tag{3.3}$$

Por fim, o valor de 10% de concentração de GLP foi definido como limite de segurança para o sistema. Como o código foi construído trabalhando-se com os valores analógicos, o valor equivalente à faixa de segurança definida é de 379. Portanto, caso as concentrações de gás GLP excedam esse valor, as devidas ações de segurança são realizadas por meio da programação. Essa faixa de valor foi definida com base nas especificações de inflamabilidade do gás GLP descritas por Petrobras (2022), onde os valores de concentrações em torno de 1000ppm do gás são suficientes para resultar em explosões. Sendo assim, a faixa de 10% da porcentagem de GLP representa as concentrações próximas aos 1000ppm, conforme ilustrado por meio da relação de sensibilidade do sensor que foi apresentada na Figura 25.

3.1.3 Módulo de monitoramento da temperatura

Para monitorar a temperatura no ambiente, foi utilizado o sensor LM35. A escolha deste sensor foi motivada devido a sua facilidade de implantação, visto que ele é nativamente calibrado

na escala Celcius. O Sensor foi alimentado como uma tensão de 5 V, e o seu pino de saída analógica (V_{out}) foi conectado diretamente na GPIO 34 do ESP32.

Embora a tensão de operação deste sensor seja diferente se comparada ao ESP32, não foi necessário implantar um divisor de tensão para realizar as leituras da saída analógica do sensor. Isso acontece porque a queda de tensão na saída do sensor varia em 10 mV a cada 1 °C, e portanto, para que a GPIO seja danificada, seria necessário que a queda de tensão atingisse valores superiores ao valor máximo de 3300 mV suportado. Neste contexto, seria necessário a ocorrência de uma temperatura de 330 °C, o que foge da realidade das medições e da faixa máxima suportada pelo LM35, que é de 100 °C.

3.1.3.1 Tratamento de dados

Para transformar os valores de tensão fornecidos por meio do pino de saída do sensor em valores de temperatura (°C), foi necessário primeiramente converter os valores interpretados pelo conversor analógico-digital para milivolts, por meio da equação 3.4.

$$TEMP_{mv} = \frac{3300 * ValorAnalgico_{temp}}{4096} \tag{3.4}$$

Após determinar o valor da temperatura em millivolts, foi realizada uma segunda conversão para se obter a temperatura em graus Celsius, considerando a relação representada por meio da equação 3.5

$$TEMP_{(\circ C)} = TEMP_{mv} * 0,1 \tag{3.5}$$

Por fim, a faixa de segurança determinada para a temperatura foi de 55 °C. Esse valor foi escolhido porque temperaturas acima desta faixa não são comuns nos ambientes residências, e portanto, valores superiores a essa faixa podem indicar uma situação de risco no ambiente, como um possível incêndio que não foi detectado por meio do sistema.

3.1.4 Módulo de alerta e mitigação de incidentes

O módulo de alerta e mitigação de incidentes é composto por um LED vermelho, um Buzzer ativo de 5 V, e um módulo relé de 5 V. O Buzzer e o LED são utilizados para emitir os sinais de alerta caso alguma situação de risco seja detectada no ambiente por meio dos três módulos de monitoramento desenvolvidos, enquanto que o relé é acionado especificamente por meio do módulo de gás GLP.

Neste contexto, quando concentrações de gás GLP superiores a 10% são detectadas por meio do sensor, o relé é acionado em conjunto com o LED e com o *buzzer*. O objetivo do relé é

permitir o desarme da energia elétrica no ambiente, uma vez que uma pequena centelha elétrica que ocorra por meio do acionamento de um interruptor é o suficiente para causar uma explosão.

O *buzzer* de 5 V é acionado por meio da porta GPIO 23 do ESP32, e esse modelo foi escolhido por proporcionar um sinal sonoro mais elevado se comparado ao *buzzer* de 3,0 V. Entretanto, como os sinais digitais de nível alto fornecidos nos pinos do ESP32 são de 3,3 V, o sinal sonoro ficou comprometido quando o acionamento do buzzer foi realizado de forma direta. Sendo assim, foi necessário alimentar o buzzer com uma tensão de 5 V, e utilizar um transistor NPN atuando como chave para efetuar o seu acionamento por meio da porta GPIO. A Figura 28 representa o circuito elétrico da montagem do *buzzer*, onde foi utilizado o transistor 2N2222.

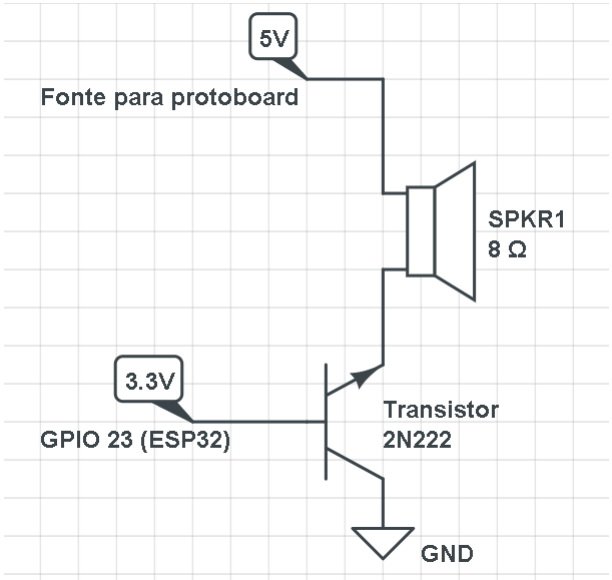

Figura 28 – Circuito do *buzzer*. Fonte: Autoria própria

Em relação ao módulo relé, foi utilizado um conector Borne KRE de três canais junto ao protoboard, para concretizar a sua montagem. Sendo assim, os dois pinos de alimentação, e o sinal de saída da porta GPIO 21 (escolhida para o acionamento do relé), foram conectados aos terminais do conector KRE. Em sequência, foram utilizados fios do tipo Macho/Fêmea para conectar o módulo relé aos terminais do KRE. Essa abordagem permitiu que a utilização do módulo relé para desarme da energia elétrica fosse opicional ao usuário.

Por fim, o LED vermelho é acionado por meio da porta GPIO 5, e possui uma resistência de 220Ω conectada em série entre a porta de alimentação e o pino anodo do LED, proporcionando uma proteção ao componente.

3.1.5 Sinalização de erros e status de conexão

Para sinalizar a conexão do ESP32 com a rede WiFi foi utilizado um LED na cor azul, acionado por meio da porta GPIO 4. De forma similar ao LED vermelho, também foi utilizado uma resistência de 220Ω conectada em série na montagem do LED azul. A partir daí, no

momento em que o ESP32 tenta realizar a conexão com a rede WiFi configurada, o LED pisca em intervalos de 250 ms, emitindo uma sinalização visual característica para esta situação, e quando a conexão é estabelecida, o LED é desligado. Caso ocorra a perda de conexão durante o funcionamento do *hardware*, o LED azul passará a piscar em intervalos mais lentos, configurados para 1 s, o que tornou possível a idetinficação visual deste caso.

Para sinalizar a ocorrência de erro interno no sistema, ambos os LEDs foram programados para serem ligados simultâneamente, e permanecem neste estado até que o sistema seja reiniciado. Este cenário ocorre apenas quando o código desenvolvido não consegue criar um recurso interno denominado semáforo MUTEX, que é explicado na seção que aborda a programação do *hardware* do sistema.

3.1.6 Montagem do *Hardware*

O modelo esquemático do circuito do trabalho foi desenvolvido inicialmente para ser montado em uma única protoboard, onde a fonte de alimentação, os componentes, os sensores, e o ESP32, seriam conectados.A Figura 29 ilustra o projeto inicial desenvolvido.

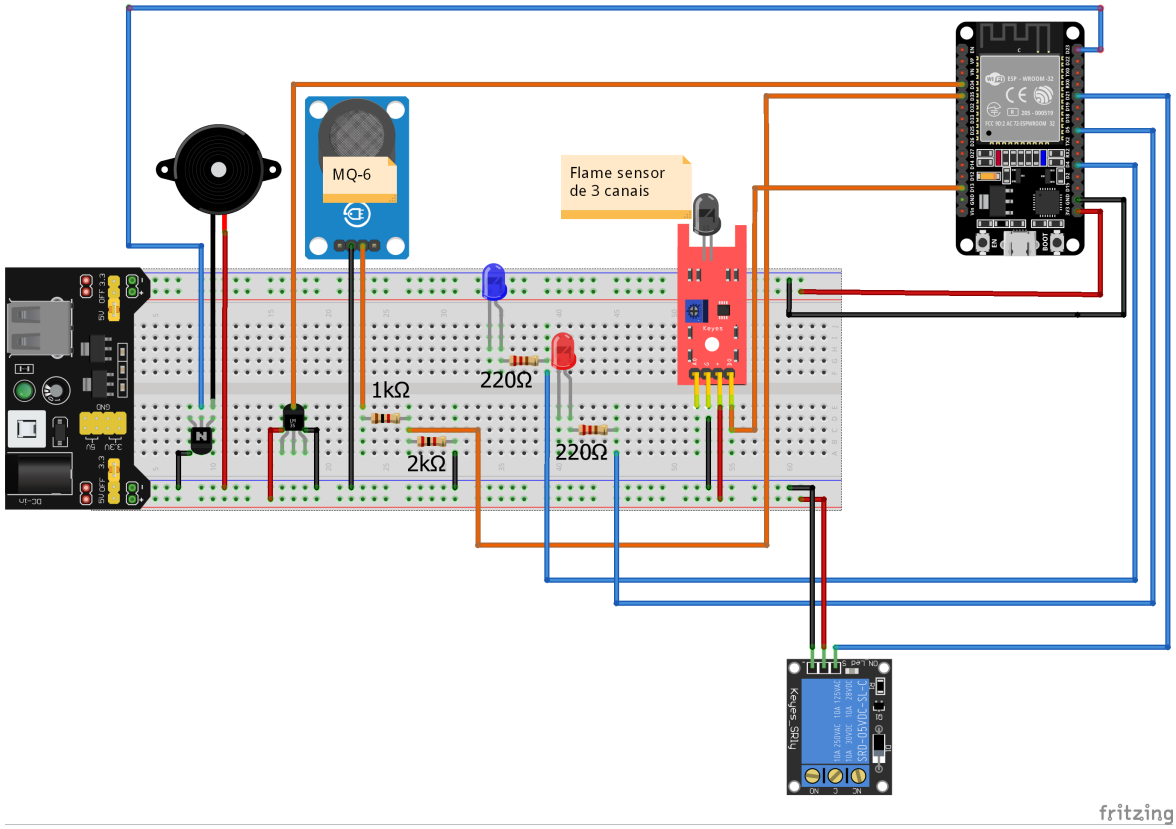

Figura 29 – Circuito inicial do *FireSafe* (*protoboard*). Fonte: Autoria Própria

Contudo, no ato da montagem constatou-se a necessidade de se utilizar duas protoboards unidas, para que fosse possível trabalhar com os dois lados do ESP32 que possuem as portas

GPIOs, bem como ter um ganho em espaço para organizar os componentes e sensores. Sendo assim, a concretização da montagem física é representada por meio da Figura 30.

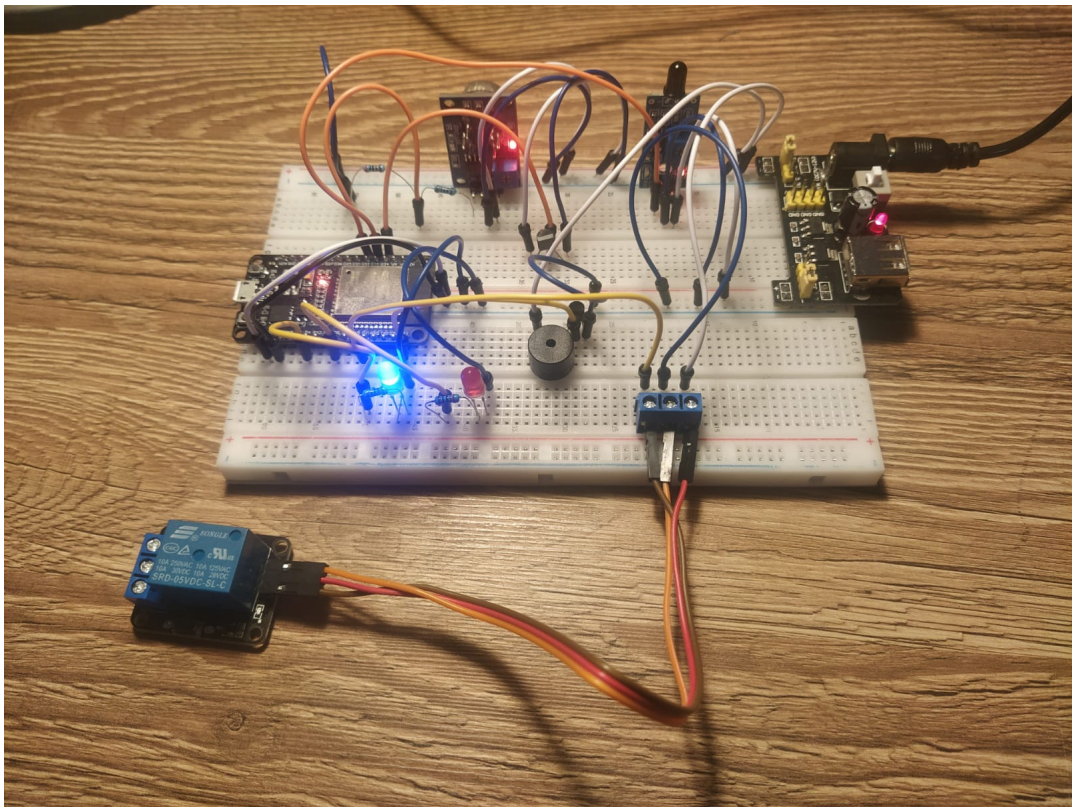

Figura 30 – Circuito final do FireSafe (*protoboard*). Fonte: Autoria Própria

3.1.7 Programação do *Hardware*

Para definir a estrutura da programação do sistema, primeiramente foi levado em consideração que, tanto a temperatura do ambiente, quanto o aumento da concentração de gás em caso de vazamentos, variam de forma mais lenta ao longo do tempo se comparado a um princípio de incêndio, uma vez que as chamas são geradas de forma instantânea. Sendo assim, o monitoramento do princípio de incêndio deve ser realizado com uma frequência maior, para reduzir ao máximo o tempo em que o sistema realiza as ações de segurança.

Portanto, se o código fosse construído de forma sequencial, a eficiência do monitoramento de chamas seria comprometida, uma vez que seria preciso ler e tratar os dados de todos os sensores, para posteriormente realizar as ações de segurança e reiniciar o processo. Então, caso uma chama se inicie logo após a leitura do sensor, seria preciso esperar que todos os outros sensores fossem tratados, e apenas na próxima iteração do processo é que a chama seria percebida pelo sistema.

Além disso, o envio das informações monitoradas no ambiente ao banco de dados do *Firebase* também demanda tempo de processamento do ESP32, e portanto, não deve competir

processamento com a seção de código que realiza o monitoramento do ambiente.

Pensando nesses aspectos, o código desenvolvido foi pensado como um sistema de tempo real do tipo *soft-realtime*, e dividido em três tarefas independentes, criadas por meio das funcionalidades do sistema operacional *free RTOS*, disponibilizadas na IDE do Arduino UNO. As tarefas foram distribuídas nos dois núcleos distintos do microcontrolador (núcleo 0, e núcleo 1), para permitir o paralelismo na execução do código. As tarefas 1 e 2 são responsáveis por realizar o monitoramento do ambiente, e portanto, são executadas de forma paralela no núcleo 1 do ESP32. Entretanto, a tarefa 3 é responsável apenas por estabelecer a conexão com a rede e realizar o envio das informações ao banco de dados, não realizando ações de segurança em casos de incidentes. Sendo assim, esta tarefa é executada de forma isolada no núcleo 0, para que não interfira na eficiência do monitoramento do ambiente. O fluxograma da estrutura do código desenvolvido é ilustrado por meio das Figuras 31 e 32, respectivamente.

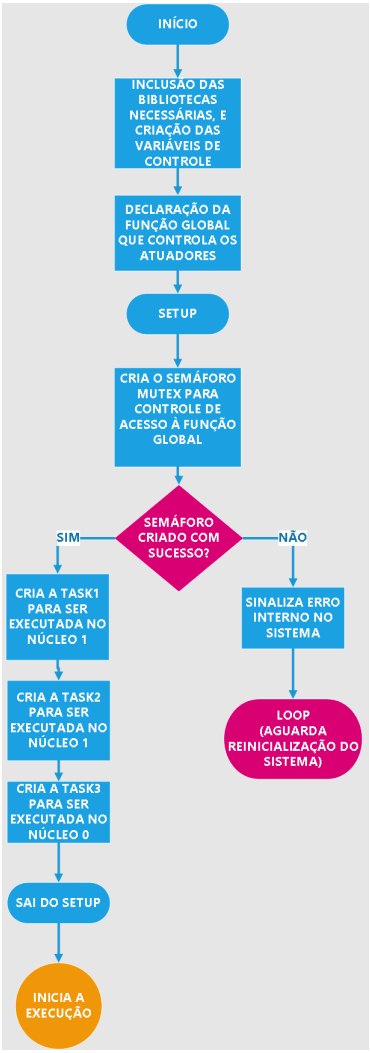

Figura 31 – Fluxograma da programação do *hardware FireSafe* (1). Fonte: Autoria Própria

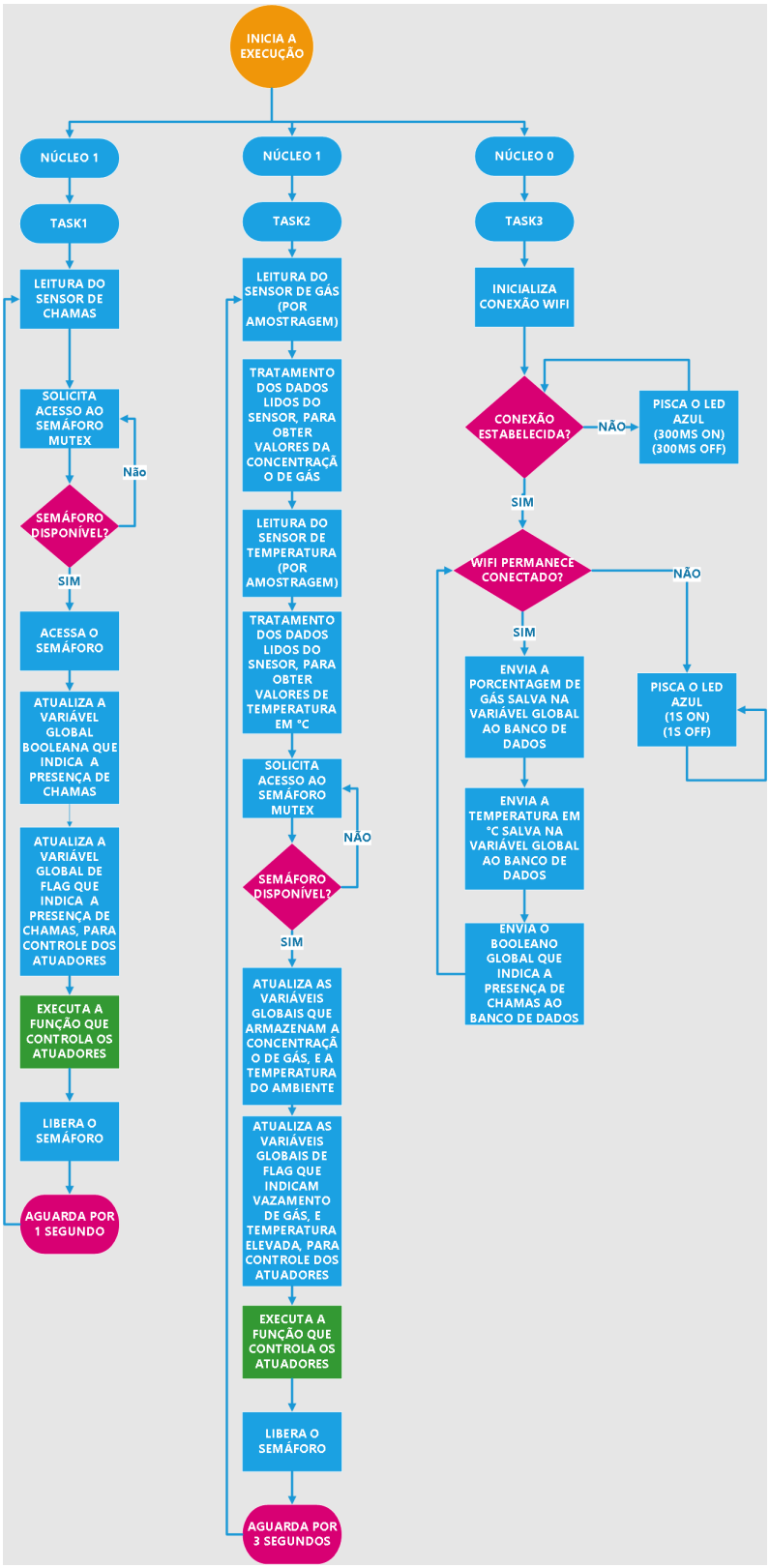

Figura 32 – Fluxograma da programação do *hardware FireSafe* (2). Fonte: Autoria Própria

Como existem duas tarefas distintas realizando o monitoramento do ambiente, ambas precisam acessar a função global que controla os atuadores do módulo de segurança do sistema. Portanto, para evitar que ambas acessem a função simultaneamente, comprometendo o funcio-

namento correto dos atuadores, foi necessário utilizar um semáforo do tipo MUTEX (*Mutual Exclusion*). Este recurso permitiu controlar o fluxo de acesso das tarefas à função, de forma que apenas uma tarefa por vez realizasse as ações necessárias (exclusão mútua), como por exemplo, acionar o LED e o *buzzer* em caso de detecção de chamas. Como este recurso é de extrema importância para o funcionamento do código, caso ocorra algum erro interno na sua criação o sistema emite um erro visual por meio dos LEDs, e encerra o seu funcionamento.

Na tarefa 1 foi desenvolvido o código referente ao módulo de detecção de chamas, onde são realizadas leituras diretas na saída do sensor. Após a leitura do sensor, é solicitado acesso ao semáforo MUTEX, e caso o mesmo esteja ocupado, a tarefa aguarda a sua liberação. Assim que o acesso ao semáforo é concedido, as variáveis de controle do módulo são atualizadas, e a função que controla os atuadores é executada pela tarefa. Após realizar as ações, a tarefa aguarda por um período de 1 s para reiniciar o seu funcionamento.

Na tarefa 2 foram desenvolvidos os códigos dos módulos de monitoramento da temperatura e vazamentos de gás. Ambos foram desenvolvidos sequencialmente na mesma tarefa por apresentarem variações mais lentas ao longo do tempo. Como os dados de saída dos sensores de temperatura e de gás são analógicos, ambos foram lidos por meio de amostras, para obter uma melhor estabilização. Sendo assim, amostras de 50 medições são coletadas de cada sensor, e as respectivas médias dos valores são computadas. Em seguida, são realizados os tratamentos de dados para determinar a concentração de gás GLP, e os valores da temperatura em graus Celsius. Após o tratamento de dados, é realizada a solicitação de acesso ao semáforo MUTEX, de forma similar a tarefa 1, onde as variáveis de controle dos módulos são atualizadas, e a função que controla os atuadores é executada.

Para a construção da tarefa 3, referente ao envio de dados, foi utilizada a biblioteca *WiFi.h* para configurar a conexão do ESP32 com a rede, bem como a biblioteca *IOXhop_FirebaseESP32.h* para enviar os dados monitorados ao banco de dados *Firebase*. A lógica da sinalização visual do *status* de conexão do sistema também foi desenvolvida nesta tarefa.

Por fim, a programação completa do *hardware* do sistema *FireSafe* pode ser acessada no Apêndice A.

3.2 Banco de Dados

O banco de dados foi criado a partir de uma estrutura de dados do tipo árvore, criada por meio do *Realtime Database* do *Firebase*. A estrutura projetada é ilustrada por meio da Figura 33, onde o nó principal denominado *monitoredEnvironment* representa o ambiente que o sistema está monitorando. A partir do nó principal, foram criados três nós secundários (filhos) que representam cada um dos monitoramentos realizados no ambiente.

Figura 33 – *Realtime Database* do *FireSafe*. Fonte: Autoria Própria

Portanto, os nós denominados *gassPercent* e *temperature* foram criados para receber valores numéricos, e armazenam, respectivamente, a porcentagem de gás GLP e os valores da temperatura em graus Celsius. O nó denominado *hasFire*, por sua vez, foi configurado para receber um valor do tipo *booleano*, que sinaliza a presença, ou ausência, de chamas no ambiente.

Para vincular o *Realtime Database* ao *hardware* do *FireSafe*, foi preciso coletar a chave secreta do banco de dados, disponibilizada nas configurações do *Firebase*, e a URL (*Uniform Resource Locator*) de acesso criada para o *Realtime Database*. Essas informações foram inseridas no código desenvolvido por meio da IDE do Arduino.

Além disso, também foi preciso configurar as regras de acesso ao banco, para permitir a leitura e a escrita de dados por meio de acessos externos. Para este trabalho, as permissões foram configuradas para sempre permitirem a leitura e a escrita ao *Realtime Database*.

Por fim, também foi necessário realizar o registro do aplicativo *FireSafe* no *Firebase*, por meio da seção de configurações do projeto do próprio *Firebase*. Esse registro gerou um arquivo do tipo *JSON* (JavaScript Object Notation), onde foi extraída a chave de segurança (api_key) responsável por autenticar a comunicação entre o aplicativo *FireSafe* e o banco de dados em tempo real. A tela de registro de um aplicativo no *Firebase* é representada por meio da Figura 34.

Figura 34 – Tela de registro de aplicativo ao *Firebase*. Fonte: Autoria Própria

3.3 Aplicativo *FireSafe*

Antes de iniciar o desenvolvimento do aplicativo, primeiramente foi definido o padrão de cores a ser utilizado nas telas, que foi pensado diante dos objetivos do sistema. Além disso, também foi criada a arte da logomarca do aplicativo, que foi projetada pensando-se no nome (*FireSafe*). Vale ressaltar que a fonte utilizada no título da logomarca é uma fonte *open source* (código aberto), e foi acessada gratuitamente por meio do *Google Fonts*. A arte da logomarca, bem como a fonte, e as cores adotadas, são representados por meio da Figura 35.

Figura 35 – Tela de registro de aplicativo no *Firebase*. Fonte: Autoria Própria

Após as definições dos padrões, foram projetadas duas telas para compor toda a construção do aplicativo *FireSafe*. Cada uma das telas foi construída a partir dos componentes disponibilizados na plataforma *Kodular*, por meio da aba *Designer*. Após montados os *layouts*, foi realizada a construção da lógica de funcionamento de cada uma das telas, por meio da programação de blocos disponibilziada na aba *Blocks*.

A primeira tela é a de abertura do aplicativo *FireSafe*, onde é apresentado a logomarca em forma de inicialização, e uma animação de carregamento para o usuário. A tela foi programada para manter sua exibição por um período de três segundos, e após esse intervalo, sua execução é encerrada e a segunda tela é inicializada. A Figura 36 representa a tela de inicialização em execução.

Figura 36 – Tela de inicialização do aplicativo *FireSafe*. Fonte: Autoria Própria

A segunda tela desenvolvida é a de monitoramento do ambiente, e é a tela principal deste trabalho. Por meio dela é possível monitorar em tempo real o *status* do ambiente, a temperatura, a presença de chamas, e a porcentagem de gás GLP mensurada. A Figura 37 representa a construção da tela.

Para exibir os dados monitorados na tela, foi realizada uma configuração para permitir a comunicação do *Kodular* com o *Firebase*, onde foi preciso fornecer a URL de acesso do *realtime database* criado, o *token* (API_key) disponibilizado no arquivo JSON (após vincular o aplicativo ao *Firebase*), e o identificador do nó raiz da árvore de dados (*Project Bucket*). Após realizada

Figura 37 – Tela de monitoramento do aplicativo *FireSafe*. Fonte: Autoria Própria

as configurações, foi possível ler os dados preenchidos no banco e disponibilizá-los em tela, e a partir daí, sempre que *hardware* realiza uma modificação nos valores do banco, essa ação é percebida pelo aplicativo e os valores são atualizados na tela. A configuração de acesso ao *Firebase* no *Kodular* é representada por meio da Figura 37

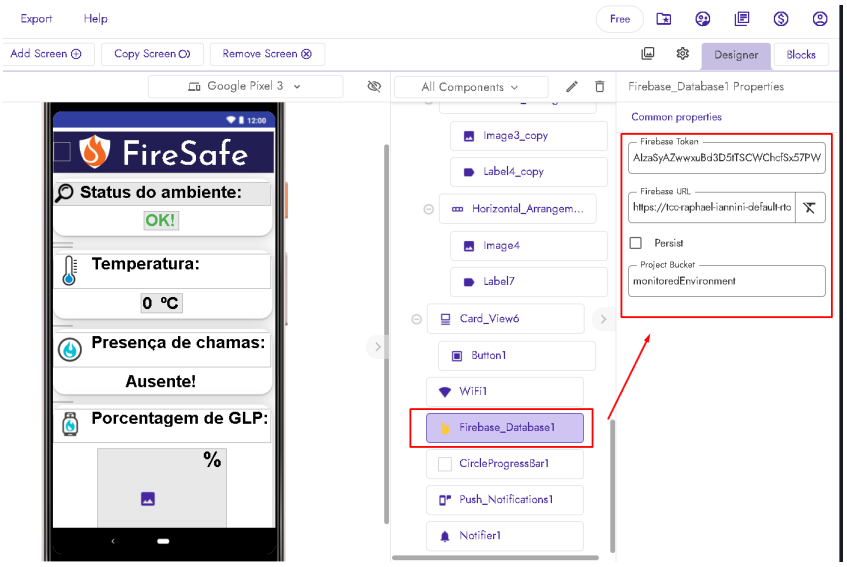

Figura 38 – Configuração de acesso ao *Firebase* no aplicativo *FireSafe*. Fonte: Autoria Própria

Os mesmos parâmetros de segurança que foram definidos para o *hardware* são utilizados na programação do aplicativo. Neste contexto, sempre que o valor lido da temperatura exceder 55 °C, a presença de chamas for detectada, ou a porcentagem de gás GLP registrada for superior a 10%, o campo da tela que informa o *status* do ambiente é alterado para "PERIGO!", com texto em cor vermelha. De forma similar, em situações de risco as cores dos textos também são alteradas para vermelho nos campos que monitoram a temperatura e a presença de chamas, o que auxiliou na percepção visual destes cenários.

Em relação ao campo que exibe os valores da concentração de gás GLP, vale ressaltar que uma barra circular dinâmica foi adicionada junto ao valor numérico da porcentagem, representando a variação dos valores por meio do preenchimento da barra.

Por fim, a programação completa do aplicativo *FireSafe* que foi desenvolvida pode ser acessada no Apêndice B.

4 Resultados

4.1 *Hardware* do *FireSafe*

Após a concretização da montagem física do *hardware* do *FireSafe* nas *protoboards*, foram realizados alguns testes para validar o funcionamento de cada um dos módulos que compõem o sistema, com a utilização de alguns componentes e ferramentas auxiliares para simular os cenários de testes. Os resultados obtidos, bem como as tratativas para os problemas encontrados, serão detalhados nos tópicos seguintes. Além disso, também serão apresentados os custos totais do projeto, e o resultado da montagem física do sistema em uma placa ilhada própria, após a realização dos testes e ajustes.

4.1.1 Testes no Módulo de Monitoramento da Temperatura

Durante os testes com o sensor LM35, foram detectados dois problemas que afetaram diretamente os valores de temperatura mensurados. O primeiro deles é relacionado a não linearidade dos conversores ADCs do ESP32, que causaram erros nas faixas de valores mensurados. Conforme representado por meio da Figura 39, os ADCs do ESP32 apresentam uma região crítica de conversão para valores de tensão baixos, entre 0 V e 0,30 V, bem como para valores acima dos 3,0 V. Como o sensor LM35 fornece uma tensão de 10 mV (0,01V) a cada 1 °C, as faixas de temperaturas mais comuns aos ambientes, que se encontram entre 0 °C e 40 °C, fornecem valores de tensões dentro da faixa crítica dos ADCs do ESP32, o que resultou em erros na interpretação da temperatura.

Neste contexto, com o auxílio de um multímetro para mensurar os valores na saída do sensor, e compará-los aos valores interpretados no ESP32, foi possível notar que os valores de temperatura interpretados eram sempre menores que os valores reais fornecidos. Sendo assim, para realizar testes mais específicos, o sensor LM35 foi substituído por um potenciômetro, onde 10 valores de tensão diferentes foram regulados e fornecidos ao ESP32. Os ajustes no potenciômetro a cada aferição foram realizados de forma que os valores de tensão na saída ficassem entre 0 mV e 1000 mV, simulando temperaturas na faixa de 0 °c a 100 °c, e abordando valores da região crítica de operação dos ADCs do ESP32.

Os resultados são representados por meio da Figura 40, onde o erro médio entre os valores de temperatura interpretados no ESP32, e os valores simulados por meio do potenciômetro, foi de 10,4 °. Além disso, foi possível observar que, para valores acima dos 1000 mV, o erro diminui consideravelmente.

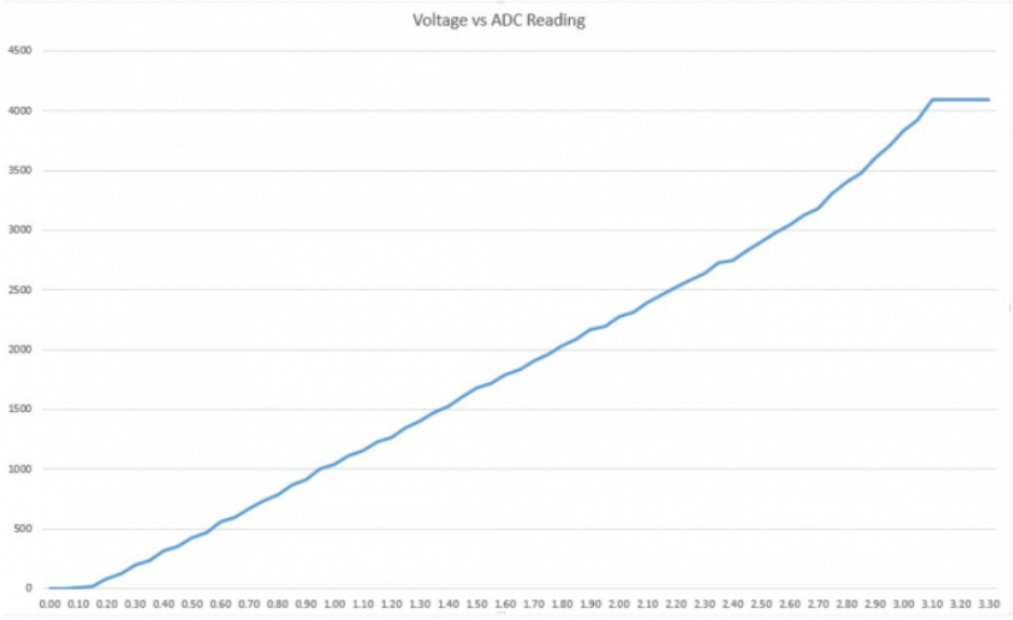

Figura 39 – Voltagem fornecida VS Valor ADC interpretado no ESP32. Fonte: Tutorials (2019)

Medições	Potenciômetro (mv)	Potenciômetro (°C)	ESP32 (mv)	ESP32 (°C)	Erro (°C)
1	53,8	5,38	0	0	5,38
2	184	18,4	34	3,4	15
3	210	21	62	6,2	14,8
4	320	32	169	16,9	15,1
5	395	39,5	240	24	15,5
6	463	46,3	306	30,6	15,7
7	625	62,5	468	46,8	15,7
8	685	68,5	526	52,6	15,9
9	783	78,3	746	74,6	3,7
10	985	98,5	955	95,5	3
					Erro médio = 10.4 °C

Figura 40 – Quadro com os 10 valores mensurados. Fonte: Autoria própria.

Diante disso, para realizar a correção na faixa dos valores, foi construída uma curva entre os valores simulados por meio do potenciômetro e os valores interpretados no ESP32, e a partir da curva encontrada, foi traçada uma curva de tendência por meio da aproximação polinomial. Ambas as curvas são representadas por meio da Figura 41, onde foi possível concluir que a curva de tendência apresenta valores satisfatórios para a faixa de 0 °C a 60 °C. Sendo assim, como o valor limite de segurança para a temperatura adotado neste trabalho é de 55 °C, a equação da curva de tendência pôde ser aplicada no código desenvolvido para o ESP32, onde foi criada a função *adjustTemperatureRange* no código do *hardware* para ajustar os valores de temperatura

interpretados pelo ESP32. Como resultado, o erro entre os valores reais da temperatura, e os valores interpretados, caíram para uma faixa de 0,5 °C à 1 °C, valores esperados para a proposta deste trabalho.

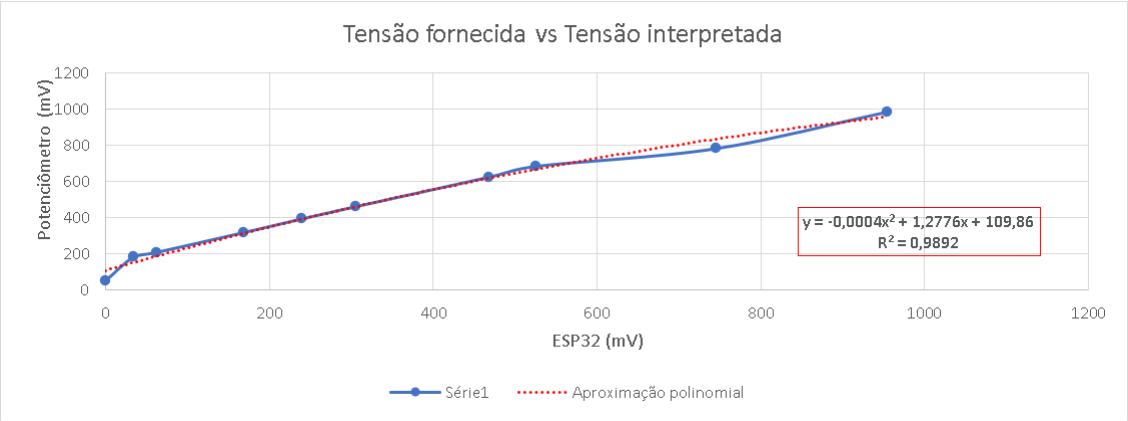

Figura 41 – Curvas computadas para os valores de tensão. Fonte: Autoria própria.

O segundo problema encontrado é a ocorrência de uma interferência interna no ADC1 do ESP32 quando o sensor de gás detecta concentrações de GLP acima da faixa de 10%. Esta interferência causa um aumento significativo nos valores da temperatura, e ocorre devido a ambos os sensores estarem conectados no mesmo conversor (ADC1). Neste contexto, mesmo com cada sensor utilizando um canal específico do ADC1, quando a tensão fornecida por meio do sensor de gás aumenta devido às altas concentrações de GLP, um ruído interno ocorre e afeta os valores analógicos do segundo canal, onde são realizadas as leituras da temperatura, causando um aumento nos valores exibidos. Este cenário é ilustrado por meio da Figura 42, onde foi utilizado um isqueiro para simular o vazamento de gás, e as altas concentrações causaram interferência nos valores da temperatura. O teste foi feito utilizando o potenciômetro como entrada, com um valor arbitrário abaixo dos 55 °.

Como o ESP32 possui dois conversores analógicos, foi realizado um teste em que os sensores de gás e de temperatura foram conectados em conversores distintos, e neste cenário, a interferência deixou de existir. Contudo, o segundo conversor (ADC2) apresenta a particularidade de ser desativado quando a função Wi-Fi do ESP32 é utilizada. Neste contexto, como o Wi-Fi é necessário para realizar o envio de dados, os sensores foram mantidos em um único conversor (ADC1).

Portanto, para contornar o problema da interferência, uma variável foi criada no código para armazenar o último valor mensurado da temperatura, sempre que as concentrações de GLP estiverem abaixo de 10%. Neste contexto, quando uma situação de vazamento de gás ocorrer, e as concentrações estiverem acima dos 10%, o valor da temperatura utilizado no código será o último valor lido antes das concentrações ultrapassarem a marca de 10%, e portanto, os ruídos não serão computados. Assim que as concentrações de gás baixarem, os valores da temperatura

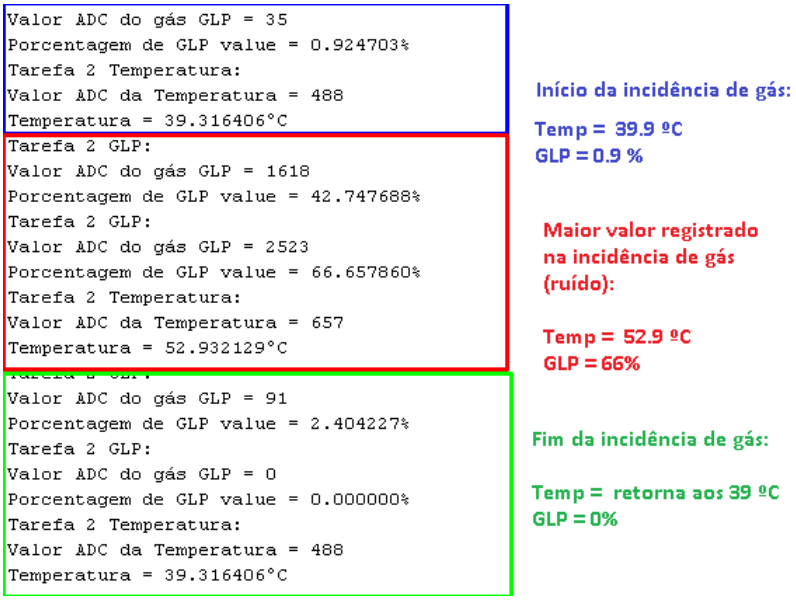

Figura 42 – Interferências no ADC1 que afetam as leituras do sensor LM35. Fonte: Autoria própria

mensurados voltam a ser computados, e a variável que armazena o último valor volta a ser atualizada.

Essa abordagem impede a ocorrência da eventual situação em que o ruído computado cause um aumento indevido na temperatura suficiente para alcançar a marca dos 55 °C, e indique uma falsa situação de risco no ambiente quanto à valores elevados de temperatura. Além disso, a temperatura varia de forma lenta ao longo do tempo, e portanto, esta abordagem não causa discrepâncias significativas nos valores.

Após aplicar as modificações no código, o cenário de teste que simula vazamentos de gás GLP foi repetido, e desta vez, a interferência não foi computada. A Figura 43 ilustra os resultados do teste, onde a temperatura registrada antes da incidência de gás foi de 27.7 °C. A partir daí, durante a detecção de altas concentrações de gás, o valor computado da temperatura se manteve em 27.7 °C, e quando finalmente as concentrações baixaram, a nova temperatura registrada foi de 27.9 °C, ilustrando que a abordagem contorna o problema sem causar discrepâncias nos valores da temperatura.

Por fim, vale ressaltar que os dois problemas encontrados podem ser evitados adotando-se um conversor ADC externo para computar os valores da temperatura, contanto que ele possua uma escala de conversão mais adequada para o sensor LM35 se comparado aos conversores do ESP32. Contudo, essa abordagem ficou como sugestão para trabalhos futuros.

4.1.2 Testes no Módulo de Monitoramento de Gás

Os testes do módulo de monitoramento de gás foram realizados por meio de incidências diretas de gás GLP no sesor MQ-6, com o auxílio de um isqueiro como fonte de gás. Diferente

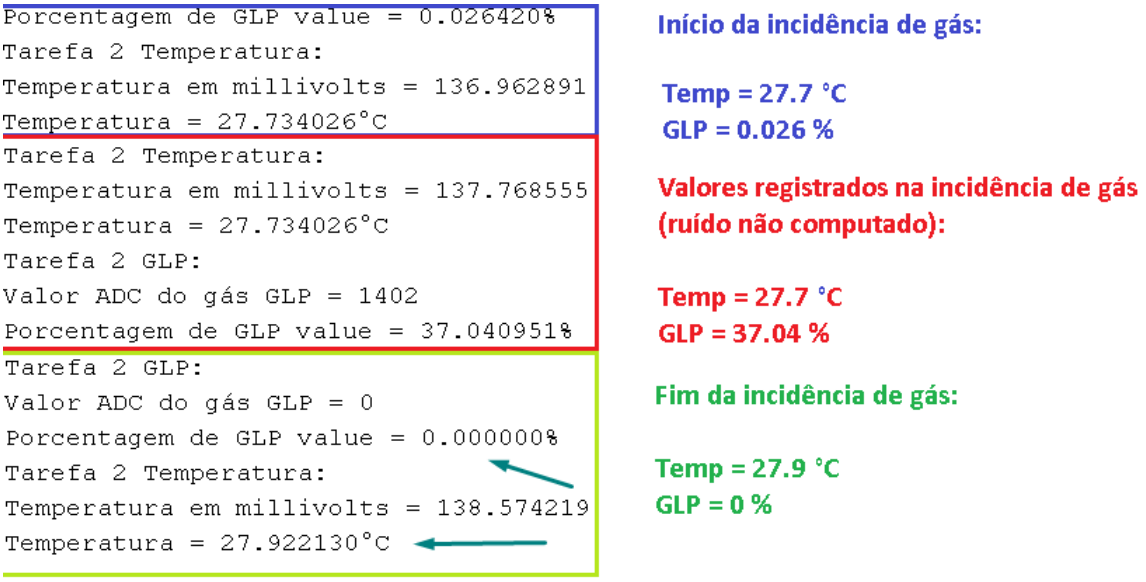

Figura 43 – Testes da solução adotada para contornar a interferência nos valores da temperatura. Fonte: Autoria própria.

do sensor LM35, o sensor MQ-6 não apresentou erros consideráveis entre os valores de tensão interpretados pelo ESP32 e os valores reais fornecidos no pino analógico do MQ-6. Isso aconteceu pois o sensor MQ-6 trabalha com variações de tensão mais altas quando exposto ao gás GLP (valores acima de 1V), e portanto, nas situações de vazamento de gás os valores fornecidos de tensão estão fora da zona crítica mínina dos conversores do ESP32.

Diante disso, não foi necessário realizar nenhuma correção de faixa nos valores lidos no sensor MQ-6. Entretanto, o sensor também apresentou ruídos em suas medições provenientes das leituras do sensor LM35, devido a ambos utilizarem o mesmo conversor (ADC1). A diferença é que, no caso do sensor MQ-6, os ruídos causados por meio do LM35 geraram uma interferência muito baixa nas concentrações de GLP, uma vez que os valores de tensão fornecidos na saída analógica do LM35 são baixos, onde por exemplo, para temperaturas na faixa dos 20 °C foram fornecidas tensões de saída na faixa dos 0,2 V. Neste contexto, os ruídos são percebidos nas medições de gás apenas quando a temperatura assume valores mais elevados, acima dos 60 °C, onde foram percebidas variações máximas em torno de 1% nos valores de concentração de GLP.

Este cenário é representado por meio da Figura 44, onde, com o auxílio de um potenciômetro, foram simuladas variações na temperatura e registrados os efeitos nas medições de GLP. Os testes foram realizados em um ambiente livre de incidência de gás, onde para baixas temperaturas (0 °C) foi possível notar que não ocorreram interferências nas medições. E então, quando a temperatura foi elevada até os 76 °C, foi possível registrar uma interferência de cerca de 0,6% nos valores medidos, o que não interfere nos objetivos de segurança definidos para este módulo.

Por fim, como os erros nas medições do sensor MQ-6 devido as interferências do sensor LM35 são baixos, e ocorrem apenas quando a temperatura assume valores acima da faixa de

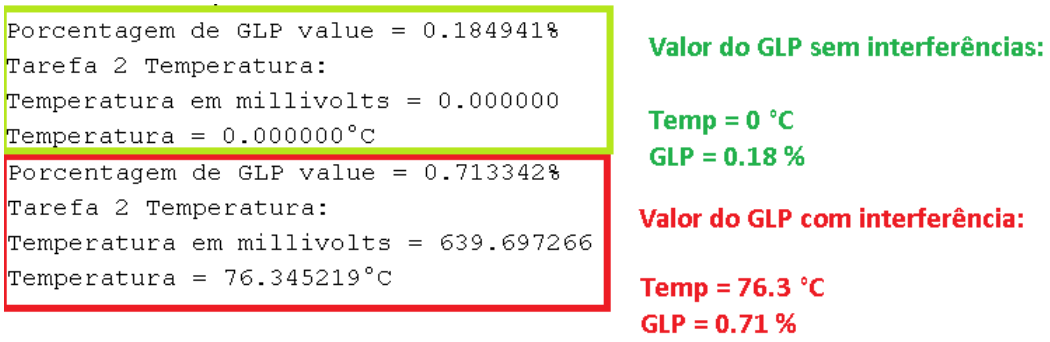

Figura 44 – Teste das interferências do ADC1 nos valores lidos por meio do sensor MQ-6. Fonte: Autoria própria.

segurança de 55 °C, nenhuma ação de correção precisou ser adotada para este cenário. Além disso, vale ressaltar que entre os primeiros cinco minutos de operação do sensor MQ-6, onde ele ainda está alcançando a sua temperatura de operação, ocorreram registros de erros de até 3% de concentração de GLP no ambiente, e portanto, é preciso aguardar um tempo após energizar o sistema para que os valores mensurados se tornem mais precisos.

4.1.3 Testes no Módulo de Detecção de Chamas

Os testes do módulo de detecção de chamas também foram realizados com o auxílio de um isqueiro, onde a chama produzida foi posicionada próxima ao sensor para sua respectiva detecção. Após um pequeno ajuste na sensibilidade do sensor, por meio do seu potenciômetro integrado, a chama produzida no isqueiro foi devidamente reconhecida por meio do sensor.

4.1.4 Testes no Módulo de Alerta e Mitigação de Incidentes

Os testes do módulo de alerta e mitigação de incidentes foram realizados por meio das seguintes simulações:

- Simulações de vazamento de gás por meio de incidências diretas de GLP no sensor MQ-6, com o auxílio de um isqueiro.
- Simulação de incêndio no ambiente por meio das chamas produzidas no isqueiro, próximas ao sensor de chamas.
- Simulações de temperaturas elevadas com a substituição do sensor LM35 por um potenciômetro, onde foram ajustadas diferentes faixas de tensões para os testes.
- Simulações de diferntes situações riscos simultâneamente, para validar o fluxo de controle dos atuadores por meio das tarefas distintas.

Em todos os casos testados o sistema de alerta funcionou conforme o esperado, e foi possível constatar a importância da utilização do semáforo MUTEX no código do *hardware* para o correto funcionamento dos atuadores nos cenários onde várias situações de risco ocorrem simultâneamente.

4.1.5 Montagem do protótipo em uma placa ilhada

Após a realização de todos os testes e ajustes nos módulos do *hardware*, o sistema físico do *FireSafe* foi transferido para uma placa ilhada perfurada, com dimensões de 10 cm x 10 cm. A Figura 45 representa o sistema em funcionamento após a montagem na placa ilhada.

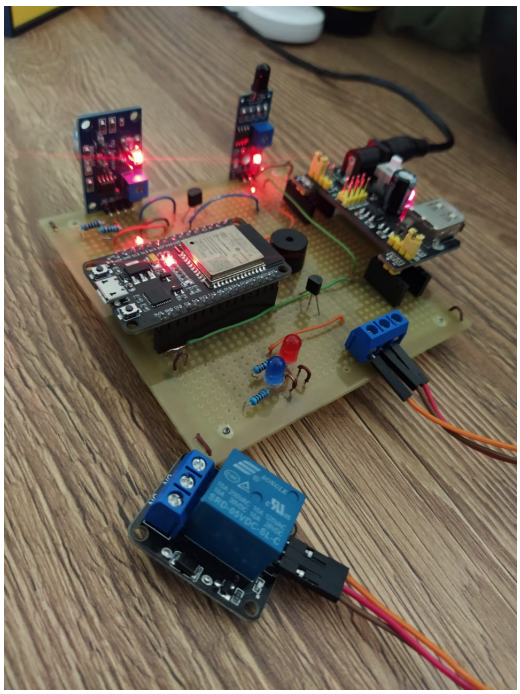

Figura 45 – *Hardware* do *FireSafe* em funcionamento após montagem em uma placa ilhada. Fonte: Autoria própria.

Durante a montagem foram utilizados barramentos do tipo fêmea para anexar a placa microcontroladora ESP32 Dvkit V1, e a fonte de alimentação para protoboards. Essa abordagem permite que o ESP32 seja desacoplado do módulo em caso de ajustes no código fonte quanto a configuração da rede Wi-Fi, bem como permite a fácil substituição da fonte em caso de possíveis defeitos.

Além disso, foram utilizados fios rígidos coloridos de cobre, extraídos de um cabo do tipo RJ-45, para auxiliar na conexão físicas dos componentes. Parafusos também foram utilizados nas extremidades da placa, para formar uma espécie de base e garantir a estabilização do módulo.

Após a montagem, novos testes foram realizados para validar o funcionamento do sistema, e o próprio aplicativo *mobile FireSafe* foi utilizado para acompanhar os cenários monitorados.

Os resultados obtidos foram coerentes com os testes realizados no *protoboard*, validando a montagem física da placa.

4.2 Validações no Banco de Dados do *Firebase*

Após a estruturação do banco de dados no *Firebase*, duas validações foram realizadas para testar a comunicação entre o *hardware* e o aplicativo *mobile*.

A primeira validação foi realizada no fluxo de envio de dados do *hardware* ao banco de dados, por meio das funções da biblioteca *IOXhop_FirebaseESP32.h*. Neste contexto, foi constatado que o tempo de processamento para a atualização dos dados no *Firebase* resultou em um pequeno atraso entre os valores mensurados, e os valores registrados. Sendo assim, os valores salvos no banco sempre estavam defasados entre uma ou duas medições, conforme representado por meio da Figura 46. Contudo, este pequeno atraso era esperado, e não causou impedimentos no monitoramento do ambiente.

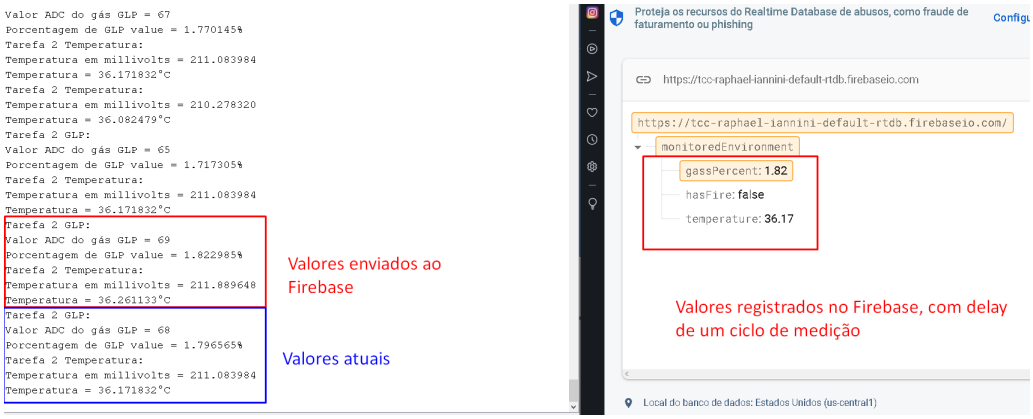

Figura 46 – Fluxo de envio de dados do *hardware* ao *Firebase*. Fonte: Autoria própria.

Por fim, a segunda validação foi realizada no fluxo entre os valores registrados no *Firebase*, e os valores exibidos no aplicativo *mobile*. Para este fluxo foi constatado que a atualização dos valores no aplicativo é quase instantânea, conforme ilustrado por meio da Figura 47, e portanto, os únicos atrasos nos valores monitorados são provenientes do envio de dados do ESP32 ao *Firebase*.

4.3 Aplicativo *FireSafe*

Após a estruturação do aplicativo *FireSafe*, dois fluxos de teste foram realizados para validar suas funcionalidades. O primeiro fluxo consistiu em utilizar o *hardware* em conjunto com o aplicativo *FireSafe*, durante um período de tempo, onde foi verificada a troca de informações entre eles. O segundo fluxo consistiu em simular uma situação em que todos os cenários de

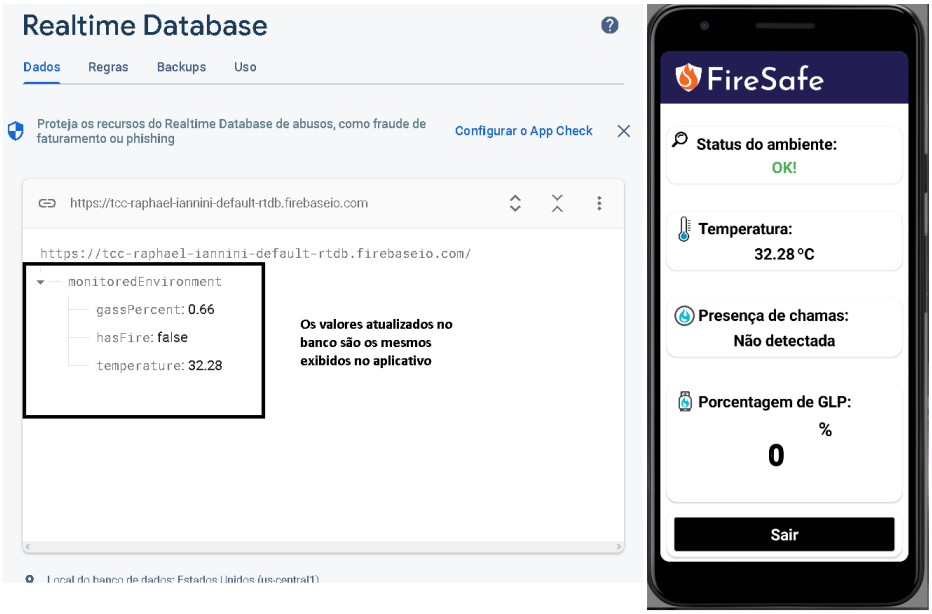

Figura 47 – Fluxo de envio de dados do *Firebase* ao aplicativo. Fonte: Autoria própria.

risco possíveis ocorreram simultânemanete no ambiente, onde foi validado o monitoramento dos mesmos por meio do aplicativo.

Em ambos os casos foi possível monitorar com sucesso as condições do ambiente, por meio do aplicativo, com um pequeno atraso (*delay*) na troca de informações. O cenário em que todas as situações de risco ocorreram simultânemante é representado por meio da Figura 48.

Figura 48 – Monitoramento do cenário em que todas as situações de risco ocorrem simultâneamente no ambiente. Fonte: Autoria própria.

Por fim, após as validações, o projeto foi exportado da plataforma *Kodular* como uma aplicação *mobile*, e instalado com sucesso em um smartphone.

4.4 Orçamento do Sistema *FireSafe*

Como um dos objetivos do trabalho foi desenvolver um sistema de baixo custo, todos os componentes utilizados na construção do *hardware* foram selecionados pensando-se na melhor relação entre o orçamento e os requisitos de segurança a serem cumpridos no projeto. Diante disso, a Figura 49 representa um quadro com o orçamento total da construção física do protótipo do *hardware*, em que os valores listados foram obtidos no período da aquisição dos componentes, isto é, no segundo semestre de 2022 (ptax do Dólar EUA igual a 5,3242, na data 26/10/2022).

Orçamento do Projeto				
Item	Valor unitário	Quantidade	Observações	Valor total
ESP32 Doit Devkit V1	R$ 43,99	1	-	R$ 43,99
Sensor Detector de Chama (Chip LM393)	R$ 10,90	1	-	R$ 10,90
Sensor de Temperatura LM35	R$ 22,90	1	-	R$ 22,90
Sensor de gás MQ-6 Propano Isobutano (GLP)	R$ 21,99	1	-	R$ 21,99
Kit 10x Barra de 6, 8, 10 Pinos Fêmea Empilhá	R$ 39,32	1	Quatro unidades de 6 pinos utilizadas	R$ 39,32
Placa de Fenolite Ilhada 10 X 10 cm	R$ 18,00	1	-	R$ 18,00
Kit 10 Peças Conector Borne Kre kf3013p	R$ 16,90	1	Uma unidade utilzada	R$ 16,90
Kit 5 Barra dde Pinos Fêmea Mci 1x15 (2.54m	R$ 27,28	1	Duas unidades utilizadas	R$ 27,28
Kit 3x Buzzer Ativo 5V	R$ 15,99	1	Uma unidade utilzada	R$ 15,99
Transistor NPN 2N2222	R$ 14,50	1	-	R$ 14,50
Kit 20X LED colorido Difuso 5mm	R$ 31,10	1	Duas unidades utilizadas	R$ 31,10
Resistor 220 Ω 5%	R$ 0,38	10	Duas unidades utilizadas	R$ 3,80
Resistor 1K Ω 5% (Pacote com 10)	R$ 0,38	1	Uma unidade utilzada	R$ 0,38
Resistor 2K Ω 5%	R$ 0,14	10	Uma unidade utilzada	R$ 1,40
Fonte ajustável para protoboard	R$ 14,16	1	-	R$ 14,16
Protoboard	R$ 17,90	2	-	R$ 35,80
Cabo de rede RJ-45 (1m)	R$ 12,00	1	lizado para extração de fios rígidos colori	R$ 12,00
Módulo Relé 5V - 1 canal	R$ 8,09	1	-	R$ 8,09
Jumper Premium Macho / Fêmea (40 p x 10 cr	R$ 7,47	1	Utilizado para acoplar o relé	R$ 7,47
Fonte 12 V Estabilizada Bivolt	R$ 7,00	1	-	R$ 7,00
TOTAL	-	-	-	R$ 352,97

Figura 49 – Quadro de orçamento do projeto. Fonte: Autoria própria.

Vale ressaltar que toda a programação do *hardware*, bem como a concretização do aplicativo mobile, foram desenvolvidos por meio de plataformas e sistemas gratuitos. Sendo assim, o orçamento total do projeto ficou limitado apenas à construção do física do protótipo.

5 Considerações Finais

5.1 Conclusões

Este trabalho teve como objetivo a criação de um sistema residencial de baixo custo de alerta e mitigação de incêndios e vazamentos de gás, que permite o monitoramento do ambiente via aplicativo mobile. Para tal, foi desenvolvido o sistema denominado *FireSafe*, que é composto por um protótipo físico construído com a utilização da placa microcontroladora ESP32 Devkit V1, um aplicativo mobile que foi criado por meio da plataforma *Kodular*, e um banco de dados de tempo real construído na plataforma *Google Firebase*.

O protótipo físico desenvolvido foi capaz de mensurar os níveis das concentrações de gás GLP no ambiente por meio do sensor MQ-6, bem como foi capaz de detectar princípios de incêndio com o sensor de chamas, e enviar todos os dados mensurados ao banco de dados criado no *Firebase*. Contudo, foi necessário aplicar uma curva de correção nos valores de temperatura mensurados por meio do sensor LM35, para fornecer dados corretos e precisos. Essa ação foi necessária devido a não linearidade do conversor analógico-digital do ESP32, onde a faixa crítica de operação do conversor coincidiu com a faixa de valores fornecidas por meio do sensor de temperatura, ocasionando erros nos valores interpretados pelo microcontrolador.

Além disso, foi detectada uma interferência interna nos valores analógicos interpretados no conversor ADC1 quando concentrações de gás GLP superiores à 10% são mensuradas. Essa interferência também causou erros nos valores da temperatura, visto que tanto o sensor de gás (MQ-6), quanto o sensor de temperatura (LM35), utilizam o mesmo conversor (ADC1). Felizmente, esse problema foi contornado por meio da programação do *hardware*, e deixou de causar impactos nos resultados.

Em ambas as situações de risco testadas com o protótipo físico, descritas no capítulo anterior, houve o acionamento correto do módulo de alerta composto pelo LED vermelho e pelo *buzzer* passivo. Além disso, para os casos específicos em que houveram vazamentos de gás, o módulo relé foi devidamente acionado para permitir o desligamento da energia elétrica do local, reduzindo os riscos de explosões no ambiente.

Em relação ao aplicativo *FireSafe*, seu desenvolvimento teve como foco a facilidade que o usuário teria em monitorar as condições do ambiente por meio da interação com as telas desenvolvidas. Para isso, o aplicativo foi criado com apenas 2 telas, sendo uma de inicialização, e outra principal responsável por exibir os dados do monitoramento do ambiente, bem como o

status do mesmo. Os dados são processados e exibidos com sucesso na tela principal, de forma automática ao usuário, sempre que ocorre o registro das informações enviadas por meio do protótipo físico no banco de dados. Diante disso, foi possível monitorar o ambiente com sucesso, de forma remota, onde as ocorrências de situações de risco são facilmente identificadas por meio das informações exibidas na tela principal.

Por fim, como o aplicativo e o banco de dados não geraram custos ao sistema desenvolvido, conclui-se que o valor total de R$352,97 gasto na concretização do protótipo físico foi o custo total para a construção deste trabalho. Este valor, considerando a ptax de compra do Dólar EUA apresentada na Seção 4.4, foi de $66,30. Portanto, em relação ao baixo custo, o valor total gasto foi razoável se comparado ao trabalho similar proposto por Pinto (2016), onde o orçamento final foi de R$304,36 (que foi equivalente a $92,81, considerando a ptax do Dólar EUA apresentada na Seção 2.1). Contudo, este presente trabalho utilizou uma quantidade maior de componentes, uma vez que ele também monitora a temperatura e a presença de chamas no ambiente. Além disso, o custo deste trabalho pode ser reduzido caso se trabalhe com a produção em maiores quantidades do sistema, e este seja montado diretamente na placa de circuito impresso.

5.2 Melhorias Futuras

A primeira proposta para melhorias futuras é a integração de um conversor analógico-digital externo no *hardware* do sistema, que possua uma faixa de operação ajustada para pequenos valores de tensão. Neste contexto, o conversor externo seria utilizado para realizar as leituras analógicas fornecidas por meio do sensor de temperatura LM35, e em seguida, os valores analógicos seriam enviados para processamento no ESP32. Com esta abordagem não seria necessário aplicar uma curva de correção sobre os valores analógicos lidos do sensor de temperatura, e além disso, como o sensor de gás e o sensor de temperatura estariam conectados em conversores distintos, as interferências internas no conversor ADC1 do ESP32 também deixariam de existir.

Além disso, como o ESP32 conta com a tecnologia *Bluetooth* integrada, uma segunda melhoria seria modificar o código fonte do *hardware* para permitir a troca de dados com o aplicativo por meio do *Bluetooth*. Neste contexto, uma tela de configurações seria adicionada ao aplicativo mobile, com o objetivo de permitir ao usuário alterar a rede Wi-Fi do protótipo físico por meio de uma interface mais amigável. Sendo assim, o usuário não precisaria modificar o código fonte e regravá-lo no ESP32 sempre que houver a necessidade de alterar a rede Wi-Fi à qual o *hardware* irá se conectar.

Por fim, também é interessante realizar um estudo sobre o consumo energético do protótipo físico, para validar a viabilidade de alimentar o sistema por meio do uso de uma bateria externa, uma vez que a fonte para *protoboard* permite essa integração, e o sistema atua no

desarme da rede elétrica do local em caso de vazamentos de gás.

Referências

ALEXANDRE, D. L. Automação residencial de monitoramento de gás por meio da plataforma arduino e iot. 2021. 2021. Citado 3 vezes nas páginas 8, 21 e 22.

ALVES, S. R. P.; MENDONÇA, T. D. Desenvolvimento de protótipo robótico móvel para monitoramento gasoso em ocorrências do tipo "vazamento de glp". 2018. 2018. Citado 3 vezes nas páginas 8, 29 e 30.

ARDUINO, R. U. *Arduino UNO R3 Datasheet*. 2022. Disponível em: <https://docs.arduino.cc-/resources/datasheets/A000066-datasheet.pdf>. Acesso em: 08 ago. 2022. Citado 2 vezes nas páginas 8 e 28.

ASSIS, R. F. A. d. *Projeto de um anemômetro térmico baseado em Termistor NTC com modelo linearizado por realimentação*. Dissertação (Mestrado) — Brasil, 2018. Citado na página 32.

CORRÊA, C. et al. Mapeamento de incêndios em edificações: Um estudo de caso na cidade do recife. *Revista de Engenharia Civil IMED*, 2015. v. 2, n. 3, p. 15–34, 2015. Citado na página 17.

DANTAS, M. A. *Análise do desempenho de um queimador infravermelho funcionando com gás liquefeito de petróleo e glicerina*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2010. Citado na página 23.

DJASSEMI, O. *Kodular block code*. 2020. Disponível em: <https://www.researchgate.net-/figure/Kodular-block-code_fig4_352479055>. Acesso em: 23 ago. 2022. Citado 2 vezes nas páginas 8 e 38.

DOCS, Z. *Site Zerynth Docs - DOIT ESP32 Devkit V1*. 2022. Disponível em: <https://olddocs-.zerynth.com/r2.6.2/official/board.zerynth.doit_esp32/docs/index.html>. Acesso em: 07 ago. 2022. Citado 3 vezes nas páginas 8, 26 e 27.

ESPRESSIF. *ESP32 Series Datasheet*. 2022. Disponível em: <https://www.espressif.com/sites-/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 07 ago. 2022. Citado 2 vezes nas páginas 8 e 26.

ESPRESSIF. *ESP32-WROOM-32 Datasheet*. 2022. Disponível em: <https://www.espressif-.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 07 ago. 2022. Citado na página 26.

FBN, G. *Grupo FBN, Incêndios residenciais no 1 semestre de 2021*. 2021. Disponível em: <https://grupofbn.com.br/blog/2021/09/10/incendios-residenciais-ocorridos-no-1o-semestre-de-2021-passam-a-marca-de-3-000>. Acesso em: 04 dez. 2021. Citado na página 17.

FISPQ, G. *Ficha de Informação de Segurança de Produto Químico*. 2020. Disponível em: <https://glpgas.com.br/wp-content/uploads/2020/08/FICHA_FISPQ_2020.pdf>. Acesso em: 06 ago. 2022. Citado na página 25.

FLOP, F. *Site Filipe Flop*. 2022. Disponível em: <https://www.filipeflop.com/produto-/modulo-rele-5v-1-canal/>. Acesso em: 15 ago. 2022. Citado 2 vezes nas páginas 8 e 34.

FLOP, F. *Site Filipe Flop*. 2022. Disponível em: <https://www.filipeflop.com/produto/sensor-de-chama-fogo/>. Acesso em: 13 ago. 2022. Citado na página 32.

G1, M. G. *G1 Minas Gerais, Incêndios em residências Uberlândia 2017*. 2017. Disponível em: <https://g1.globo.com/sp/sao-paulo/noticia/2021/08/23/incendios-na-capital-e-grande-sp-ja-mataram-20-pessoas-neste-ano-numero-e-50percent-maior-do-que-em-todo-ano-passado-.ghtml>. Acesso em: 05 dez. 2021. Citado na página 17.

G1, S. P. *G1 São Paulo, Incêndios na Capital e Grande SP 2021*. 2021. Disponível em: <https:/-/g1.globo.com/sp/sao-paulo/noticia/2021/08/23/incendios-na-capital-e-grande-sp-ja-mataram-20-pessoas-neste-ano-numero-e-50percent-maior-do-que-em-todo-ano-passado.ghtml>. Acesso em: 05 dez. 2021. Citado na página 17.

GHOSH, A. *What Are the Advantages of EPP32 Over Arduino UNO?* 2019. Disponível em: <https://thecustomizewindows.com/2019/05/what-are-the-advantages-of-epp32-over-arduino-uno/>. Acesso em: 09 ago. 2022. Citado 2 vezes nas páginas 8 e 28.

INSTRUMENTS, T. *Datasheet LM35*. 2000. Disponível em: <https://pdf1.alldatasheet.com-/datasheet-pdf/download/517588/TI1/LM35.html>. Acesso em: 11 ago. 2022. Citado 3 vezes nas páginas 8, 31 e 32.

JOSEPH, J. *Site Circuit Digest*. 2022. Disponível em: <https://circuitdigest.com/microcontroller-projects/interfacing-flame-sensor-with-arduino>. Acesso em: 13 ago. 2022. Citado 2 vezes nas páginas 8 e 33.

JOY-IT. *Datasheet KY-026*. 2017. Disponível em: <https://datasheetspdf.com/datasheet/KY-026-.html>. Acesso em: 13 ago. 2022. Citado na página 32.

JRS, D. *Digital JRS, Incêndios estruturais em 2020*. 2021. Disponível em: <https://jrs.digital-/2021/02/08/noticias-de-incendios-estruturais-aumentam-437-em-2020>. Acesso em: 05 dez. 2021. Citado 3 vezes nas páginas 8, 16 e 17.

LEAK, I. *Site LEAK Inspection*. 2018. Disponível em: <https://www.leak.com.br/2018/11/16-/entenda-os-riscos-que-oferece-um-vazamento-de-gas/>. Acesso em: 06 ago. 2022. Citado na página 25.

MAKERS, K. *Site Kits Arduino*. 2022. Disponível em: <https://www.kitsarduino-.com.br/cmp/buzzer.html>. Acesso em: 17 ago. 2022. Citado 2 vezes nas páginas 8 e 35.

MERÇON, V. d. A. Sistema de controle de iluminação e monitoramento de consumo elétrico residencial via aplicativo. 2022. 2022. Citado na página 37.

MURATORI, J. R.; BÓ, P. H. D. Automação residencial: conceitos e aplicações. *Belo Horizonte: Educere*, 2013. 2013. Citado na página 18.

OLIVEIRA, M.; SHINOHARA, A. Experience with natural gas/lpg in the plasterer polo araripe, pe, brazil. *Cerâmica*, 2014. SciELO Brasil, v. 60, p. 243–253, 2014. Citado na página 23.

ONLINE, R. *Firebase: descubra para que serve, como funciona e como usar*. 2021. Disponível em: <https://www.remessaonline.com.br/blog/firebase-descubra-para-que-serve-como-funciona-e-como-usar/>. Acesso em: 26 ago. 2022. Citado na página 39.

ORACLE. *Oracle Brasil, O que é IoT*. 2019. Disponível em: <https://www.oracle-.com/br/internet-of-things/what-is-iot>. Acesso em: 07 dez. 2021. Citado na página 18.

PETROBRAS. *Gás Liquefeito de Petróleo - Informações Técnicas*. 2022. Disponível em: <https://petrobras.com.br/data/files/47/63/18/74/EB62F7105FC7BCD7E9E99EA8-/Manual\%20de\%20GLP_\%20fevereiro\%202022.pdf>. Acesso em: 04 jun. 2022. Citado 5 vezes nas páginas 8, 23, 24, 25 e 45.

PINTO, A. C. V. Desenvolvimento do sistema de monitoramento do gás lp com alarme por sms. 2016. 2016. Citado 6 vezes nas páginas 8, 21, 22, 30, 36 e 69.

PORTUGAL, A. *Site Arduino Portugal*. 2022. Disponível em: <https://www.arduinoportugal.pt-/o-que-e-arduino/>. Acesso em: 08 ago. 2022. Citado na página 27.

RAMACHANDRAN, G. *The economics of fire protection*. [S.l.]: Routledge, 2008. Citado na página 17.

ROBÓTICA, U. E. . *Site Usinainfo*. 2022. Disponível em: <https://www.usinainfo.com-.br/buzzer-arduino-535>. Acesso em: 17 ago. 2022. Citado 2 vezes nas páginas 34 e 35.

SANTOS, R. V. d. Uso do arduíno e shield ethernet para monitoramento de luminosidade, controle de temperatura e dispositivos. 2018. Universidade Tecnológica Federal do Paraná, 2018. Citado na página 33.

SEITO, A. I. et al. A segurança contra incêndio no brasil. *São Paulo: Projeto Editora*, 2008. p. 496, 2008. Citado na página 17.

SENSORS, H. *Datasheet MQ-6*. 2022. Disponível em: <https://www.sparkfun.com/datasheets-/Sensors/Biometric/MQ-6.pdf>. Acesso em: 11 ago. 2022. Citado 2 vezes nas páginas 8 e 31.

SILVA, G. C. D. *Dispositivo eletrônico paraconsistente de alerta de vazamento de gás liquefeito de petróleo para instalações residenciais e pequenos comércios*. Tese (Doutorado) — Universidade Paulista, 2016. Citado 2 vezes nas páginas 8 e 23.

SOUZA, F. *Introdução ao Arduino – Primeiros passos na plataforma*. 2013. Disponível em: <https://embarcados.com.br/arduino-primeiros-passos/>. Acesso em: 21 ago. 2022. Citado na página 37.

SPRINKLER. *Instituto Sprinkler Brasil, Estatísticas 2019*. 2019. Disponível em: <https:/-/sprinklerbrasil.org.br/estatisticas-2019>. Acesso em: 04 dez. 2021. Citado 2 vezes nas páginas 8 e 16.

TAKAHASHI, G. B. Melhorias na automação de smart lâmpadas com uso inteligente da temperatura de cor e sua dimerização. 2022. Universidade Federal de São Paulo, 2022. Citado 3 vezes nas páginas 8, 35 e 36.

TODAY, C. *LPG sensor using arduino*. 2014. Disponível em: <http://www.circuitstoday-.com/lpg-sensor-using-arduino>. Acesso em: 17 set. 2022. Citado 2 vezes nas páginas 8 e 43.

TUTORIALS, R. N. *ESP32 ADC – Read Analog Values with Arduino IDE*. 2019. Disponível em: <https://randomnerdtutorials.com/esp32-adc-analog-read-arduino-ide/>. Acesso em: 01 out. 2022. Citado 2 vezes nas páginas 9 e 59.

TUTORIALS, R. N. *Site Randon Nerd Tutorials*. 2022. Disponível em: <https:/-/randomnerdtutorials.com/esp32-doit-devkit-v1-board-pinout-30-gpios-copy/>. Acesso em: 07 ago. 2022. Citado 2 vezes nas páginas 8 e 27.

WENDLING, M. Sensores. *Universidade Estadual Paulista. São Paulo*, 2010. v. 2010, p. 20, 2010. Citado na página 29.

Apêndices

APÊNDICE A – Programação ESP32

```
////////////////////////////////////////////////////////////////////////////////////////////

// Autor: Raphael Iannini Dutra
// UFOP − Eng. de Controle e Automação Industrial.

////////////////////////////////////////////////////////////////////////////////////////////

//−−−−−−−−−−−−−−−−−−−−−−−−−−−− HARDWARE FireSafe

//Bibliotecas:

#include <WiFi.h> //biblioteca para conectar o ESP32 com a rede via WiFi
#include <IOXhop_FirebaseESP32.h> //biblioteca para o ESP32 se comunicar com Firebase
#include <ArduinoJson.h> //biblioteca para colocar informação no formato json, utilizado no firebase
#include <Arduino.h>
#include "driver/adc.h"
#include "esp_adc_cal.h"

////////////////////////////////////////////////////////////////////////////////////////////

//Pinagem e ADC do ESP32:

#define GLP_SENSOR 35
#define TEMP_SENSOR 34
#define FIRE_SENSOR 13
#define BUZZER_ALARM 23
#define WIFI_LED 4
#define DANGER_LED 5
#define RELAY_ACTUATOR 21



////////////////////////////////////////////////////////////////////////////////////////////

//Constantes calculadas para controle da programação:
```

```
#define GLP_SAFE_LIMIT_VALUE 379 // Valor calculado referente a 10% de presença de gás GLP,
    ↪ na escala analógica.
#define GLP_MAX_VALUE 3785.0 // Valor calculado referente a 100% de presença de gás GLP, na
    ↪ escala analógica do ESP32
#define ADC_Vref_mV 3300.0 // V de referencia para o ADC do ESP32 (3.3V)
#define ADC_resolution 4096.0 // Resolução do ADC do ESP32 (12 bits)

////////////////////////////////////////////////////////////////////////////////////////

// Variáveis globais para acesso ao banco de Dados (Firebase):

#define WIFI_SSID "Rede_Raphael" //Nome da rede WiFi
#define WIFI_PASSWORD "senha@123!@#" //Senha da rede WiFi
#define FIREBASE_HOST "https://tcc-raphael-iannini-default-rtdb.firebaseio.com/" //Link do banco
    ↪ de dados Firebase
#define FIREBASE_AUTH "3gwTu9KPx2nqLgqIm9RZnaxBi0HLNe6sU4XruyRl" //Senha do Banco
    ↪ de dados

////////////////////////////////////////////////////////////////////////////////////////

//variáveis globais para o monitoramento do gás GLP;

int analogGLPValue;
float milliVoltsGLPValue;
float percentGLPValue = 0;

////////////////////////////////////////////////////////////////////////////////////////

//variáveis globais para o monitoramento da temperatura

int analogTempValue;
float milliVoltsTempValue;
double tempCelsius = 0;
double adjustedMilliVoltsTempValue; // Variável para correção de faixa (Adicionada após as análises
    ↪ realizadas no capítulo 4)
float lastValidMilliVoltsTempValue = 0; // Variável que armazena o último valor analógico válido lido
    ↪ (para contornar interferências - capítulo 4)

////////////////////////////////////////////////////////////////////////////////////////

//variáveis globais para o monitoramento da presença de chamas
```

```
bool fireSensorValue = false; // O sensor de chamas atua de forma invertida: (LOW = chamas
    ↪ detectadas, HIGH = ausência de chamas)

////////////////////////////////////////////////////////////////////////////////////////

//variáveis globais do tipo flag para controle dos atuadores;

bool hassFire = false;
bool hassGLPLeak = false;
bool hassHighTemp = false;

bool alarmActive = false; // variável para validar se o alarme já esta ligado (LED vermelho +
    ↪ BUZZER)
bool relayActive = false; // variável para validar se o relé já está ligado

////////////////////////////////////////////////////////////////////////////////////////

// Função que controla os atuadores:
void actuatorsControl (bool hassFire, bool hassGLPLeak, bool hassHighTemp){
  //Condições para ligar/desligar o sistema de alarme (DANGER_LED e o BUZZER)
  if (hassFire || hassGLPLeak || hassHighTemp){ // Qualquer incidência de perigo, aciona o alarme do
      ↪ sistema:
    if(!alarmActive){
      digitalWrite(DANGER_LED, HIGH);
      digitalWrite(BUZZER_ALARM, HIGH);
      alarmActive = true;
    }
  } else{ //Se não houver nenhuma situação de perigo, desarma o sistema:
    digitalWrite(DANGER_LED, LOW);
    digitalWrite(BUZZER_ALARM, LOW);
    alarmActive = false;
  }
  //Condição para acionar o Relé de disarme da rede elétrica:
  if (hassGLPLeak) {
    if(!relayActive){
      digitalWrite(RELAY_ACTUATOR, HIGH);
      relayActive = true;
    }
  }else{
    digitalWrite(RELAY_ACTUATOR, LOW);
    relayActive = false;
```

```
  }

}// Fim função.

////////////////////////////////////////////////////////////////////////////////////////////

// Função que realiza a correção de faixa da (temperatura), em millivolts (Adicionada após as análises
    ↪ realizadas no capítulo 4)

double adjustTemperatureRange (float milliVoltsTempValue ){
  //Equação polinomial de correção: Y = −0,0004∗(X)^2 + 1,2776(X) + 109,86
  double Y = 0;
  Y = (−0.0004 ∗ (milliVoltsTempValue∗milliVoltsTempValue)) + (1.2776 ∗ milliVoltsTempValue) +
      ↪ 109.86;
  return Y;

}// Fim da função


////////////////////////////////////////////////////////////////////////////////////////////


//variáveis globais para prints na Serial:

float oldGLPValue = −1;
float oldTempValue = −1;
bool oldFireSensorValue = false;

////////////////////////////////////////////////////////////////////////////////////////////

//TAREFAS (TASKS):

TaskHandle_t fireModule; // Tarefa 1
TaskHandle_t gassAndTempModule; // Tarefa 2
TaskHandle_t sendDataModule; // Tarefa 3


//SEMFORO MUTEX:

SemaphoreHandle_t actuatorsControl_semaphore; // Semáforo utilizado para controlar o fluxo do
    ↪ acionamento dos atuadores das Tarefas 1 e 2, por meio da função actuatorsControl();
```

```
////////////////////////////////////////////////////////////////////////////////////////

//------INCIO:

void setup() {
  //----------Inicialização da serial:
  Serial.begin(115200);

  //----------Configurações do ADC:
  //analogReadResolution(12);
  adc1_config_width(ADC_WIDTH_BIT_12); // configura a resolução do ADC1 para 12 bits
  adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_DB_6); // define a atenuação do
      ↪ ADC1 canal 6 (Sensor de temperatura)
  adc1_config_channel_atten(ADC1_CHANNEL_7, ADC_ATTEN_DB_6); // define a atenuação do
      ↪ ADC1 canal 7 (sensor de gás)


  //----------Preparação e definição dos pinos:
  pinMode(GLP_SENSOR,INPUT);
  pinMode(TEMP_SENSOR,INPUT);
  pinMode(FIRE_SENSOR,INPUT);
  pinMode(BUZZER_ALARM,OUTPUT);
  pinMode(WIFI_LED,OUTPUT);
  pinMode(DANGER_LED,OUTPUT);
  pinMode(RELAY_ACTUATOR,OUTPUT);


 //----------Criação do Semáforo MUTEX:
  actuatorsControl_semaphore = xSemaphoreCreateMutex();
  //Verifica se o semáforo foi criado:
  if (actuatorsControl_semaphore == NULL)
  {/* Se o semáforo não for criado, o funcionamento correto dos atuadores se torna comprometido.
    * Logo, a sinalização de erro interno érealizada, e nada mais éfeito: */
    Serial.println("Erro interno: não foi possivel criar o semáforo MUTEX");
    //Sinaliza a ocorrência de erro interno (LED WIFI e LED DANGER acesos de forma fixa e
        ↪ permanente)
    digitalWrite(DANGER_LED, HIGH);
    digitalWrite(WIFI_LED, HIGH);
    while(true);
  }

  ////////////////////////////////////////////////////////////////////////////////////////
```

```
  //------------Criação das tarefas:

  //------------Tarefa 1:

  xTaskCreatePinnedToCore(
    fireModuleCode, /* Função que contém o código da Task 1 (módulo de incêndio). */
    "fireModule", /* name of task. */
    10000, /* Tamanho da Pilha */
    NULL, /* Parmetro da Task */
    1, /* Prioridade da Task, definida como 1 */
    &fireModule, /* Task handle to keep track of created task */
    1);
  ////////////////////////////////////////////////////////////////////////////////////////

  //------------Tarefa 2:

  xTaskCreatePinnedToCore(
    gassAndTempModuleCode, /* Função que contém o código da Task 2 (módulo de gás GLP e
        ↪ temperatura). */
    "gassAndTempModule", /* name of task. */
    10000, /* Tamanho da Pilha */
    NULL, /* Parmetro da Task 2 */
    1, /* Prioridade da Task 2, definida como 1 */
    &gassAndTempModule, /* Task handle para monitoramento da task*/
    1); /* A Task 2 rodará no núcleo 1*/
  ////////////////////////////////////////////////////////////////////////////////////////

  //Tarefa 3:

  xTaskCreatePinnedToCore(
    sendDataModuleCode, /* Função que contém o código da Task 3 (envio de dados). */
    "sendDataModule", /* name of task 3. */
    10000, /* Tamanho da Pilha */
    NULL, /* Parmetro da Task 3 */
    1, /* Prioridade da Task 3, definida como 1 */
    &sendDataModule, /* Task handle para monitoramento da task*/
    0);

} // Fim setup()

////////////////////////////////////////////////////////////////////////////////////////
```

```
//------------Código das Tarefas:

//------------TAREFA 01:

void fireModuleCode(void * pvParameters){
  Serial.print("Tarefa 1 executando no núcleo: ");
  Serial.println(xPortGetCoreID());

  //Código:
  while(true){

    ////////////////////////////////////////////////////////////////////////////////////

    //------------Realiza a leitura do sensor de chamas:

    fireSensorValue = digitalRead(FIRE_SENSOR);

    ////////////////////////////////////////////////////////////////////////////////////

    //------------Realiza o controle dos atuadores:

    xSemaphoreTake(actuatorsControl_semaphore, portMAX_DELAY ); // Solicita acesso ao semá
        ↪ foro, para controlar os atuadores, aguardando até conseguir acessar.
      //Atualiza a variável global de controle dos atuadores para a presença de chamas:
      fireSensorValue ? hassFire= false: hassFire = true; // o sensor trabalha em lógica invertida (
          ↪ HIGH para ausência de fogo, LOW para a presença).
      //Chama a função que controla os atuadores:
      actuatorsControl (hassFire,hassGLPLeak,hassHighTemp);
    xSemaphoreGive(actuatorsControl_semaphore); // libera o semáforo após realização das operaçõ
        ↪ es

    ////////////////////////////////////////////////////////////////////////////////////

    //------------Bloco para prints da presença de chamas no ambiente:
    if(fireSensorValue!= oldFireSensorValue){
        oldFireSensorValue = fireSensorValue;
        Serial.print("Tarefa 1 (chamas): ");
        Serial.printf("Presença de chamas: ");
        Serial.printf(fireSensorValue? "nao detectada \n":"detectada!! (PERIGO) \n");
    }
```

```
    ////////////////////////////////////////////////////////////////////////////////

    //Aguarda 1 segundo para reiniciar:
    delay(1000);
  }

} //Fim da Tarefa1

////////////////////////////////////////////////////////////////////////////////////

//------------Tarefa 02:
void gassAndTempModuleCode(void * pvParameters){

  Serial.print("Tarefa 2 executando no núcleo: ");
  Serial.println(xPortGetCoreID());

  //------------Código loop:
  while(true){

    ////////////////////////////////////////////////////////////////////////////////

    //------------Realiza a leitura do sensor de gás (MQ-6):

    //Coleta 50 leituras do sensor MQ-6,em um intervalo de 0.5s, e computa a média.
    analogGLPValue=0;
    adcAttachPin(GLP_SENSOR);
    for (int i=0; i<50; i++){
      analogGLPValue += adc1_get_raw(ADC1_CHANNEL_7); // realiza as leituras do valor analó
          ↪ gico diretamente no channel da GPIO 35
      delay (10);
    }
    analogGLPValue = analogGLPValue/50; // média das medições, para valores mais precisos

    //------------Tratamento dos dados:

    percentGLPValue = (analogGLPValue * 100)/GLP_MAX_VALUE;

    ////////////////////////////////////////////////////////////////////////////////

    //------------Realiza a leitura do sensor de temperatura (LM35):

    //Coleta 50 leituras do sensor, e computa a média.;
```

```
analogTempValue=0;
adcAttachPin(TEMP_SENSOR);
for (int i=0; i<50; i++){
    analogTempValue += adc1_get_raw(ADC1_CHANNEL_6); // realiza as leituras do valor analó
        ↪ gico diretamente no channel da GPIO 34
    delay(10);
 }
analogTempValue = analogTempValue/50; // média das medições, para valores mais precisos


 //----------Tratamento dos dados

  milliVoltsTempValue = (ADC_Vref_mV * analogTempValue)/ADC_resolution; //transforma o
      ↪ valor analógico lido para millivolts

 //----Aplica a correção de faixa (detalhada no capítulo 4)

 if(percentGLPValue<10){ //Caso não exista vazamentos de gás, e exista valores lidos na GPIO
     ↪ para a temperatura:
  lastValidMilliVoltsTempValue = milliVoltsTempValue; // armazena o último valor válido da
      ↪ temperatura (em mV)
   if(analogTempValue>0) {//se existirem valores analógicos, aplica a correção
     adjustedMilliVoltsTempValue = adjustTemperatureRange(milliVoltsTempValue); //aplica a
         ↪ correção de faixa (mV)
     tempCelsius = adjustedMilliVoltsTempValue *0.1; // transforma a faixa ajustada em grau
         ↪ Celsius (1 grau Celsius a cada 10mV - LM35)
   } else
       tempCelsius = 0;
 }else{ //Caso exista vazamentos acima de 10%, utiliza o último valor válido (em mV) para não
     ↪ computar ruído:
   adjustedMilliVoltsTempValue = adjustTemperatureRange(lastValidMilliVoltsTempValue); //
       ↪ aplica a correção de faixa, considerando o último valor válido (mV)
   tempCelsius = adjustedMilliVoltsTempValue * 0.1; // transforma o último valor válido da
       ↪ temperatura em grau Celsius.
 }

 ////////////////////////////////////////////////////////////////////////////////////////

 //----------Realiza o controle dos atuadores:

 xSemaphoreTake(actuatorsControl_semaphore, portMAX_DELAY ); // Solicita acesso ao semá
     ↪ foro, para controlar os atuadores, aguardando até conseguir acessar.
```

```
    //Atualiza a variável global de controle dos atuadores para o a detecção de vazamento de gás:
     analogGLPValue > GLP_SAFE_LIMIT_VALUE ? hassGLPLeak = true : hassGLPLeak = false
        ↪ ; // verifica se o valor lido excede os 10% da escala analógica
    //Atualiza a variável global de controle dos atuadores para a detecção de temperatura elevada:
    tempCelsius > 55 ? hassHighTemp = true : hassHighTemp = false; // verifica se a temperatura
        ↪ excede o limite de segurança
    //Chama a função que controla os atuadores:
    actuatorsControl (hassFire,hassGLPLeak,hassHighTemp);
  xSemaphoreGive(actuatorsControl_semaphore); // libera o semáforo após realização das operaçõ
      ↪ es

  ////////////////////////////////////////////////////////////////////////////////////////

  //----------Bloco para prints da presença de gás GLP e a temperatura do ambiente:

    if(analogGLPValue != oldGLPValue){
      oldGLPValue = analogGLPValue; //atualiza ultimo valor lido
      Serial.println("Tarefa 2 GLP:" );
      Serial.printf("Valor ADC do gás GLP = %d\n",analogGLPValue);
      //Serial.printf("ADC millivolts value = %f\n",milliVoltsGLPValue);
      Serial.printf("Porcentagem de GLP value = %f%s\n",percentGLPValue,"%");
    }

  if (tempCelsius != oldTempValue){
      oldTempValue = tempCelsius;
      Serial.println("Tarefa 2 Temperatura:" );
      Serial.printf("Temperatura em millivolts = %f\n",milliVoltsTempValue);
      Serial.printf("Temperatura = %f%s\n",tempCelsius,"C");
  }

  ////////////////////////////////////////////////////////////////////////////////////////

  //Aguarda 3 segundos para reiniciar
  delay(3000);
  }

} // Fim da Tarefa 2

////////////////////////////////////////////////////////////////////////////////////////

//----------//Tarefa 03:
```

```
void sendDataModuleCode(void * pvParameters){

  Serial.print("Tarefa 3 executando no núcleo: ");
  Serial.println(xPortGetCoreID());

  ////////////////////////////////////////////////////////////////////////////////

  //----------Inicia conexão WiFi:
      //Aciona o LED indicativo de conexão


  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //inicia comunicação com wifi com rede definica
      ↪ anteriormente
  Serial.print("Conectando a rede WiFi"); //imprime "Conectando ao wifi"
  while (WiFi.status() != WL_CONNECTED)
  {
      digitalWrite(WIFI_LED,HIGH);
      delay(250);
      digitalWrite(WIFI_LED,LOW);
      delay(250);
  }
  Serial.println("Tarefa 3: Conexão WiFi estabelecida com sucesso!"); //imprime pulo de linha

  ////////////////////////////////////////////////////////////////////////////////

  //----------Inicia conexão com o Firebase:

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  ////////////////////////////////////////////////////////////////////////////////

  //----------Código loop:

  while (true){
   if(WiFi.status() == WL_CONNECTED){ // Verifica se o WiFi permanece conectado
      //Envia os dados para o Firebase:
      Firebase.setFloat("monitoredEnvironment/gassPercent", percentGLPValue); //presença de gás
          ↪ GLP (%)
      Firebase.setFloat("monitoredEnvironment/temperature", tempCelsius); // valor da temperatura (
          ↪ C)
      Firebase.setBool("monitoredEnvironment/hasFire", hassFire); //Presença de chamas (verdadeiro
          ↪ /falso)
```

```
    } else { // Caso o WiFi não esteja mais conectado:
                    //Pisca o LED WiFi de forma lenta (1 em 1 segundo), indicado problema de conexão
                      digitalWrite(WIFI_LED,HIGH);
                      delay(1000);
                      digitalWrite(WIFI_LED,LOW);
                      delay(1000);
                }
  }

}// Fim da Tarefa 3

//////////////////////////////////////////////////////////////////////////////////////////////
void loop() {
  /*
    * O loop não realiza nenhuma ação, visto que toda a lógica éexecutada diretamente nas Tasks.
    */

} // Fim
```

APÊNDICE B – Programação do Aplicativo *FireSafe*

B.1 Tela de Inicialização

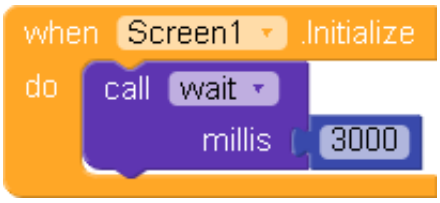

Figura B.1.1 – Bloco de inicialização da tela

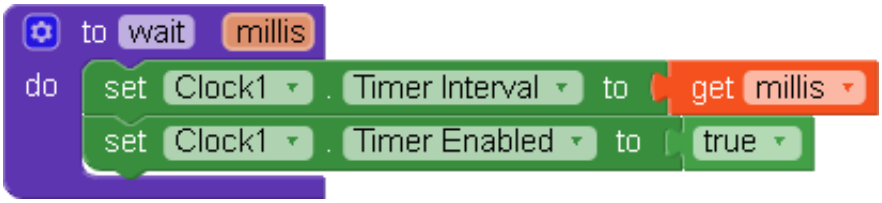

Figura B.1.2 – Bloco que aciona o contador (3 segundos)

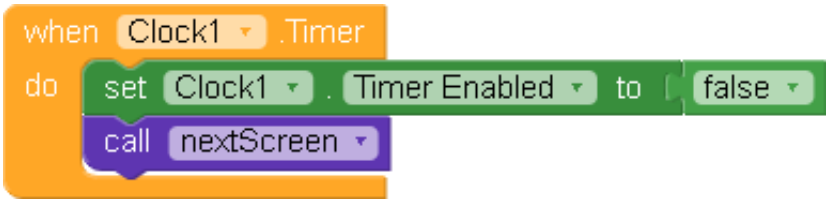

Figura B.1.3 – Bloco que aciona a próxima tela (*Screen*)

Figura B.1.4 – Bloco de chamada da próxima tela

B.2 Tela de Monitoramento

Figura B.2.1 – Declaração da variável global de porcentagem de gás GLP

Figura B.2.2 – Declaração da variável global de presença de chamas

Figura B.2.3 – Declaração da variável global de temperatura

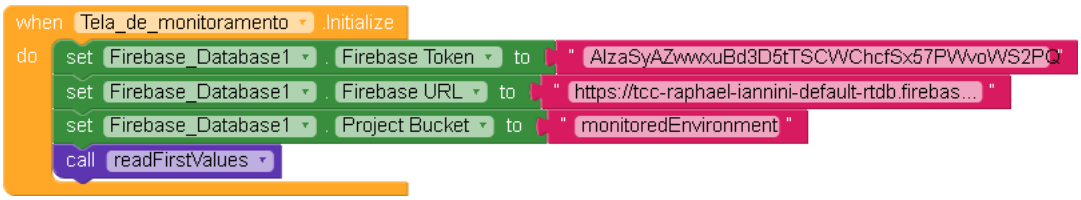

Figura B.2.4 – Bloco de inicialização da tela, que invoca a leitura dos primeiros valores no *Firebase*

Figura B.2.5 – Bloco que concretiza a solicitação de leitura dos primeiros valores no *Firebase*

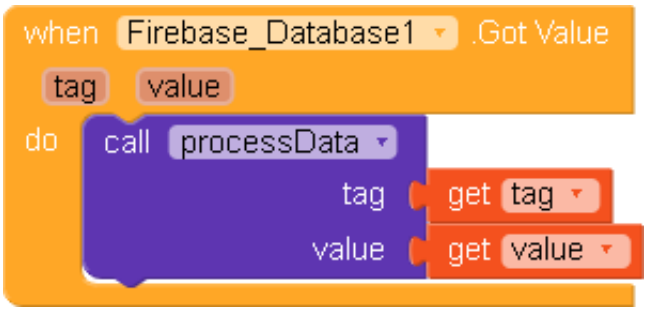

Figura B.2.6 – Bloco que aciona o processamento de dados após retorno de valores do *Firebase*

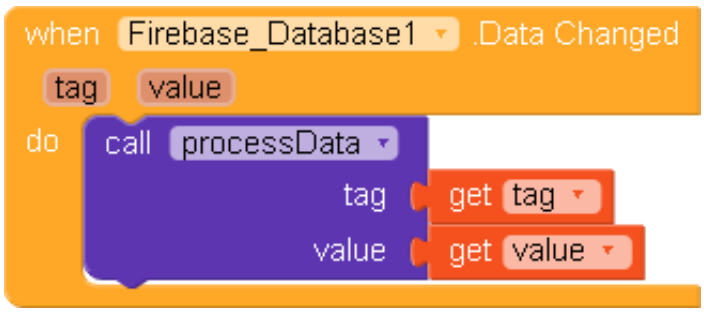

Figura B.2.7 – Bloco que aciona o processamento de dados sempre que os valores salvos no *Firebase* forem atualizados

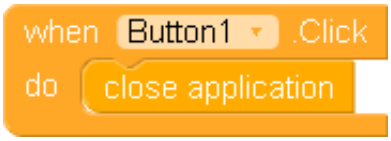

Figura B.2.8 – Bloco que encerra o aplicativo por meio do botão

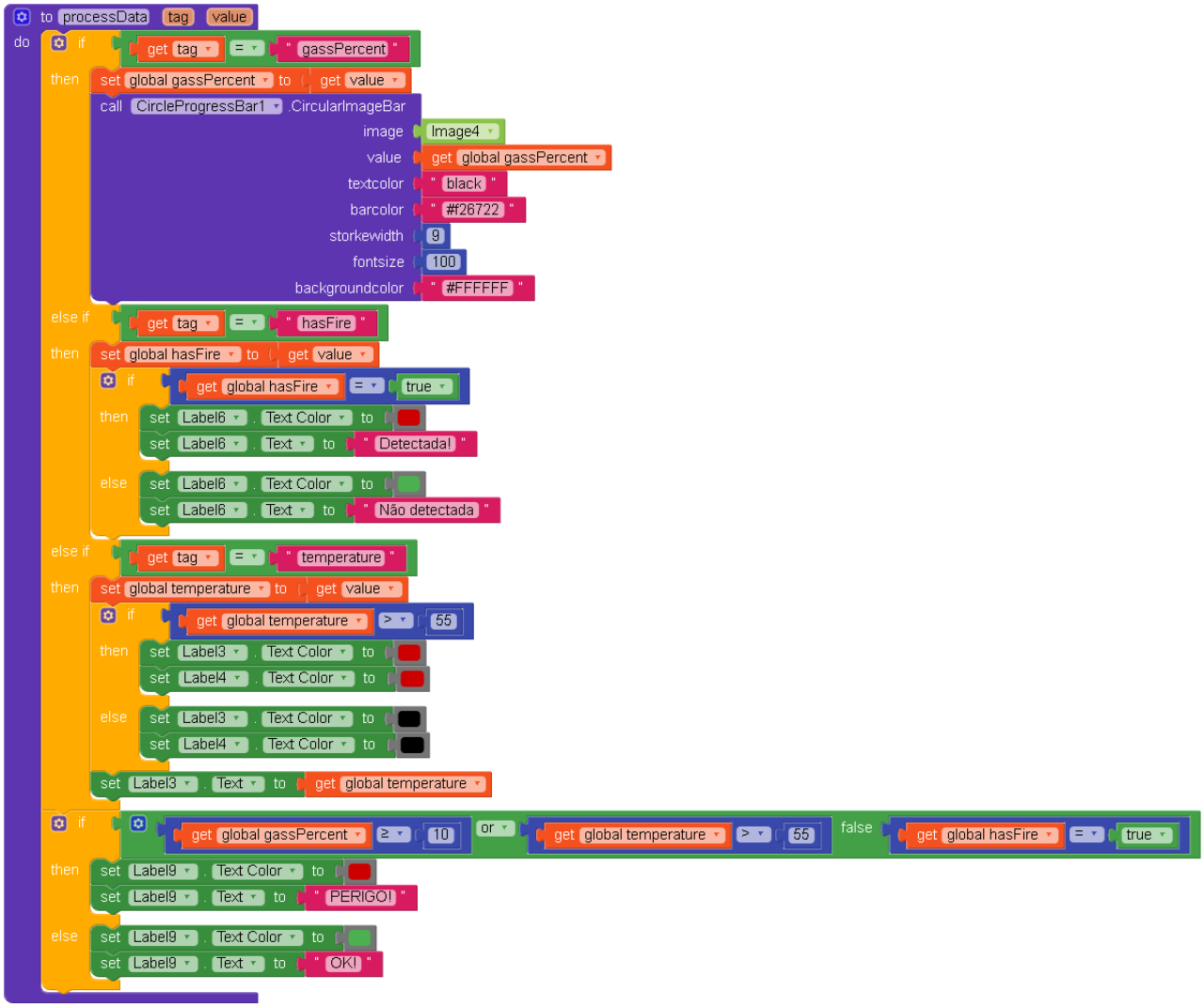

Figura B.2.9 – Bloco de processamento dos dados