RESIDÊNCIA TECNOLÓGICA EM SISTEMAS EMBARCADOS

HBR - UNIDADE DE BRASÍLIA - DF

PROJETO FINAL - CERRADOBITFIRE - SISTEMA DE MONITORAMENTO E PREVENÇÃO DE INCÊNDIOS NO CERRADO BRASILEIRO

ISAAC MARTINS DE OLIVEIRA BRAGA E SOUSA

MONITOR: DR. EDUARDO PEIXOTO FERNANDES DA SILVA

Brasília - DF

24 de fevereiro de 2025

Sumário

1 Escopo do Projeto

1.1 Apresentação do CerradoBitFire

O Cerrado brasileiro é um dos biomas mais ricos em biodiversidade do mundo e, apesar de em certo nível ser adaptado e dependente das queimadas naturais para manter seu ecossistema, concomitantemente, é um dos biomas mais vulneráveis a incêndios em função de ações antrópicas. Segundo a MapBiomas Brasil, 9,7 milhões de hectares foram queimados entre janeiro e dezembro de 2024, sendo que 85% (ou 8,2 milhões de hectares) ocorreram em áreas de vegetação nativa, onde houve um aumento de 47% em relação à média dos últimos 6 anos.

O Sistema de monitoramento de incêndios e queimadas no cerrado é feito por meio de imagens de satélite, plataformas como o Monitor do Fogo do MapBiomas e do Banco de Dados de Queimadas do INPE fornecem dados para o acompanhamento da região. Essas imagens de altíssima resolução despendem grande capital financeiro, além de que não possibilitarem a comunicação e monitoramento em tempo real dos recursos do cerrado.

Nesse sentido, surge a ideia do CerradoBitFire como alternativa inovadora baseada em sistemas embarcados capazes de monitorar variações de temperatura que possam indicar focos de incêndio em estágio inicial. Esse monitoramento em tempo real permite que ações preventivas sejam tomadas de forma mais eficiente, reduzindo significativamente os danos ambientais e os custos associados ao combate ao fogo.

Apesar de estar na fase de protótipo, o sistema demonstra um enorme potencial ao diferenciar-se de métodos tradicionais que são dependentes de monitoramento visual de satélites com atualizações periódicas, uma solução baseada em sistemas embarcados tem em seu cerne a possibilidade de respostas rápidas e precisas, o que é fundamental para conter incêndios antes que se tornem incontroláveis.

Além do impacto ambiental, essa tecnologia pode ser integrada a políticas públicas e sistemas de defesa civil, o que gera impacto direto no paradigma de proteção de áreas de preservação, fazendas e reservas indígenas. O potencial do projeto vai além do Cerrado, podendo ser adaptado para outros biomas que sofrem com queimadas, como a Amazônia e o Pantanal.

1.2 Requisitos e Funcionamento

O CerradoBitFire é um sistema embarcado projetado para monitorar elevações de temperatura que possam indicar riscos iminentes de incêndios no Cerrado brasileiro. Desenvolvido com um enfoque de baixo custo e alta eficiência, o sistema é baseado na placa de desenvolvimento BitDogLab, esta combina microcontroladores, sensores, atuadores e periféricos como display OLED, LEDs RGB, BUZZER entre outros. Sua proposta é fornecer monitoramento contínuo e emitir alertas antecipados que permitam uma resposta rápida a possíveis focos de incêndio, reduzindo danos ambientais e facilitando a atuação de equipes de emergência.

O funcionamento do sistema baseia-se na coleta de dados de temperatura por meio do sensor DS18B20, um dispositivo de alta precisão que utiliza comunicação 1-Wire para enviar informações ao microcontrolador. Essa comunicação ocorre por meio de apenas um fio de dados associado a um resistor ***pull-up*** para assegurar a estabilidade do sinal, além dos terminais VCC (alimentação do sensor em 3.3 V) e um GND (aterramento),

garantindo um design simples e econômico. O sensor DS18B20 tem ainda a vantagem de ser resistente à água, permitindo que o sistema continue operando em condições adversas, característica fundamental dos sistemas que atuam em tempo real, como durante chuvas ou em meio ao orvalho da manhã, fatores comuns no Cerrado.

Uma vez coletados, os valores de temperatura são processados pela Raspberry Pi Pico W e exibidos em tempo real no display OLED, proporcionando monitoramento constante e visualização imediata das condições ambientais. Para complementar a interface do sistema, um LED RGB controlado por modulação PWM (Pulse Width Modulation) altera sua cor de acordo com o nível de risco detectado. Quando a temperatura está dentro dos padrões normais, o LED permanece verde, indicando um ambiente seguro. Com o aumento da temperatura, a cor vai sendo alterada progessivamente, do amarelo, sinalizando condição de atenção e em casos críticos, onde o risco de incêndio se torna iminente, o LED passa para a cor vermelha, indicando perigo elevado.

Além dos sinais visuais, o sistema inclui um BUZZER, que entra em ação caso a temperatura ultrapasse um limite crítico previamente definido.É válido salientar que os alertas sonoros e luminosos do projeto atuam como abstrações para mecanismos de comunicação mais amplos, funcionando como um gatilho para envio de notificações remotas a órgãos competentes, como corpo de bombeiros, brigadas voluntárias e órgãos ambientais, de maneira que, o escalonamento desses sinais possibilitem sua integração com infraestruturas maiores como como redes LoRa, Wi-Fi e GSM.

Uma das principais vantagens do CerradoBitFire está no seu baixo consumo de energia, um fator essencial para sua aplicação em regiões do Cerrado marcadas por sua infraestrutura limitada. Muitas áreas propensas a incêndios não possuem acesso fácil à eletricidade ou redes de comunicação convencionais, tornando fundamental o uso de um sistema eficiente e de baixo consumo energético. A Raspberry Pi Pico W, por ser baseada em um microcontrolador RP2040, consome pouca energia e pode ser alimentada por painéis solares, baterias ou fontes alternativas, garantindo autonomia e funcionamento prolongado mesmo em locais isolados. Além disso, a escolha de componentes de fácil aquisição e baixo custo torna o sistema economicamente viável, permitindo que seja replicado e expandido para diferentes regiões sem demandar investimentos elevados. Dessa forma, o sistema se pociona como uma alternativa viável para a mitigação de desastres ambientais e a proteção de um dos biomas mais importantes do planeta.

1.3 Originalidade

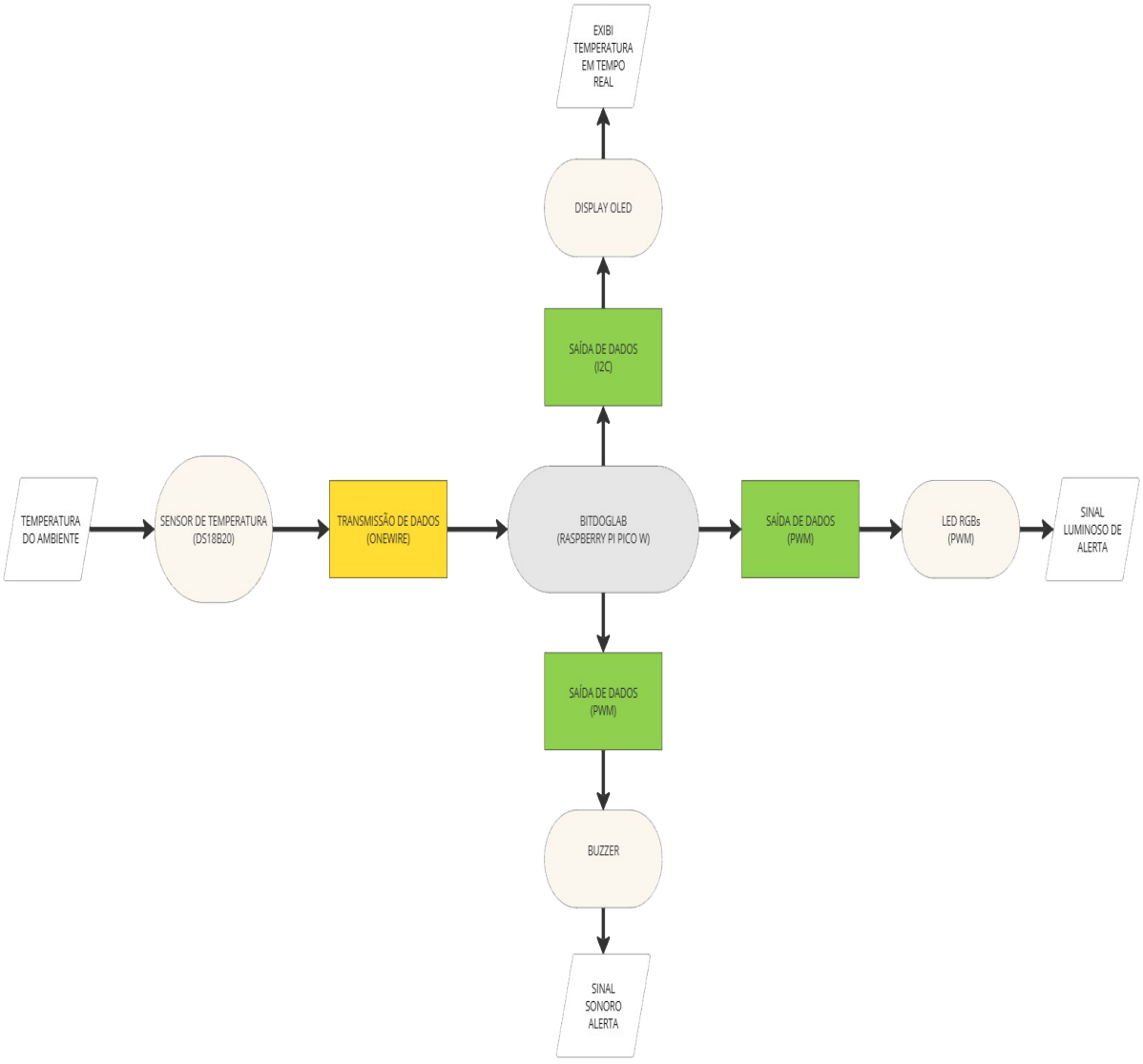
O campo de monitoramento de temperatura para detecção de incêndios tem sido explorado em diversos estudos e projetos. Muitos projetos existentes no mercado ou em pesquisa utilizam plataformas amplamente conhecidas, como Arduino, ESP32 ou mesmo o Raspberry Pi aliadas aos sensores de comunicação com ambiente e majoritariamente combinadas com Python ou MicroPython. Apesar da eficiência elevada dessa versão otimizada baseada em Python, para serviços de operação em ***real time*** a eficiência e confiabilidade são fatores inegociáveis, o que já destaca o CerradoBitFire por ser baseado totalmente na linguaguem C que possui eficiência comprovadamente superior às linguagens de alto nível.

Além das características já mencionadas, é importante destacar que muitos dos projetos existentes na área de monitoramento ambiental, especialmente os que utilizam Arduino, ESP32 e Python, são frequentemente mais voltados para aplicações agrícolas, com

foco na verificação da umidade do solo e monitoramento de temperatura relacionada ao crescimento de plantas. Esses sistemas, embora essenciais para a gestão agrícola inteligente, têm uma abordagem centrada em monitoramento de condições ambientais para a otimização da irrigação e da produção agrícola, não sendo projetados especificamente para detecção de incêndios ou monitoramento de riscos de fogo em ambientes naturais, como o Cerrado.

2 Hardware

2.1 Diagrama em Blocos



2.2 Função e Configuração de cada Bloco

- **Temperatura do Ambiente**: O sistema começa com a leitura da temperatura ambiente. Essa informação é capturada pelo sensor DS18B20, que é capaz de medir a temperatura de forma precisa e confiável. Segundo o Corpo de Bombeiros do Goiás (CBMGO), a temperatura máxima que o solo pode chegar em função da incidência solar é de cerca de 75°C, o que não é suficiente para causar ignição do fogo, no entanto, já é sinal de alerta para risco, o que será explorado pelo PWM dos LEDs e do Buzzer.

- **Sensor DS18B20**: Este componente coleta os dados de temperatura do ambiente. Ele é conectado ao sistema via comunicação OneWire, que permite a transferência de dados com um único fio de comunicação. O sensor envia a temperatura para a Raspberry Pi Pico, que é o cérebro do sistema onde esses dados serão processados.

- **Transmissão de Dados 1-Wire (One Wire)**: O sensor DS18B20 utiliza uma interface OneWire, um protocolo de comunicação serial desenvolvido pela Dallas Semiconductor (agora Maxim Integrated). Como o nome sugere, ele utiliza apenas um único fio para comunicação de dados, além de um fio de terra (GND) e um fio VCC (alimentação de 3.3 V), o que o torna simples e eficiente para aplicações onde a redução de cabos é importante.

- **BitDogLab (Raspberry Pi Pico W)**: A BitDogLab é uma placa de desenvolvimento baseada na Raspberry Pi Pico W, essa por sua vez, possui o microcontrolador RP2040 como unidade de processamento, os dados coletados pelo Sensor DS18B20 são transmitidos para nosso sistema e então a partir desse processamento é feita a comunicação e gestão dos periféricos de acordo com a temperatura medida.

- **Saída de Dados (I2C)**: A comunicação I2C é usada para enviar os dados processados pela Raspberry Pi Pico para o display OLED. A interface I2C facilita a comunicação entre múltiplos dispositivos com um número reduzido de fios, proporcionando simplicidade e eficiência no design.

- **Display OLED**: O display OLED é utilizado para mostrar a temperatura do ambiente em tempo real. Ele é alimentado pela Raspberry Pi Pico, que transmite os dados de temperatura para visualização. O display permite que os usuários monitorem facilmente a condição ambiental local.

- **Saída de Dados (PWM)**: O controle do PWM (Pulse Width Modulation) é usado para ajustar a intensidade do LED RGB e do BUZZER. O PWM permite um controle preciso da intensidade luminosa e sonora, permitindo que o sistema gere diferentes níveis de alerta de acordo com a temperatura.

- **LED RGB (PWM)**: Os LEDs RGB são utilizados como indicadores visuais dos níveis de risco. Dependendo da temperatura, os LEDs mudam de cor (verde, amarelo e vermelho), proporcionando uma forma rápida e clara de identificar a gravidade da situação.

- **BUZZER (PWM)**: O buzzer é responsável por emitir um sinal sonoro quando a temperatura atinge um limite de risco. O controle do PWM permite variar a

intensidade do som, proporcionando alertas mais fortes e claros conforme o nível de risco aumenta.

2.3 Especificações e Lista de Materiais

O CerradoBitFire foi projetado para atender as necessidades de um bioma marcada por elevadas temperaturas, grande vulnerabilidade a incêndios naturais e provocados, bem como regiões de baixa cobertura de infraestrutura e energia de maneira que a escolha dos componentes e a estrutura do sistema foram cuidadosamente planejadas para garantir baixo consumo de energia, operação autônoma, resistência às condições ambientais e eficácia na detecção de riscos de incêndio, tudo isso a baixíssimo custo financeiro. A enxuta lista de materiais destaca o baixo custo financeiro, alta disponibilidade dos dispositivos utilizados resistência a condições adversas.

- **Placa de Desenvolvimento BitDogLab (1 Unidade):** É uma placa de desenvolvimento baseada na Raspberry Pi Pico W, essa por sua vez, possui o microcontrolador RP2040 como unidade de processamento. Essa vai ser o cérebro de nosso sistema, vai processar os dados coletados pelo sensor de entrada, isto é, o sensor de temperatura DS18B20, a partir desse processamento irá realizar a comunicação para os demais periféricos, a saber, o controle do PWM para os LEDs RGBs bem como para nosso Buzzer, também é responsável pelo comunicação com nosso display OLED, que irá atualizar em tempo real o valor da temperatura medido.

- **Buzzer (1 Unidade)**: É uma interface de saída que vai emitir um sinal sonoro quando acionado, ele funciona de maneira similar a um auto-falante comum, possui uma bobina eletromagnética e uma membrana de vibração. Quando um sinal elétrico variável é aplicado a bobina, ele cria um campo magnético que iterage com o imã que movimenta a bobina, por consequência a membrana que fará ressonância com o ar que por fim gera a onda sonora. Esse periférico já vem conectado a nossa BitDogLab no pino GPIO21, por padrão.

- **Display OLED (1 Unidade)**: É uma interface de saída que irá exibir em tempo real o valor da temperatura do ambiente em graus celsius (°C). Esse periférico já vem por padrão em nossa BitDogLab conectados nos pinos SDA GPIO14 e SCL GPIO15. Seu protocolo de comunicação é o barramento I2C. Para seu funcionamento devem ser inseridas algumas bibliotecas em nosso projeto o que será abordado na descrição do software.

- **LEDs RGB (2 Unidades: 1 cor vermelho e 1 cor verde):** Esses LEDs já vem por padrão na BitDogLaB, eles são conectados por padrão nos pinos GPIO11 e GPIO13, verde e vermelho, respectivamente, sua comunicação pelo PWD permite que o sinal luminoso tenha intensidade modulada de acordo com as variações da temperatura para representar o nível de alerta e risco de cada faixa de temperatura; Ele inicialmente, começa na cor verde (não há risco iminente), conforme a temperatura aumenta a corda vai sendo progressivamente modificada para o amarelo (indicando que está em condição de alerta, mas não necessariamente risco iminente de fogo) e por fim, caso a temperatura alcance um valor crítico o LED se torna integralmente vermelho, o que serve de alerta máximo para risco iminente de incêndio.

- **Sensor DS18B20 (1 Unidade)**: É um sensor digital de temperatura fabricado pela Maxim Integrated, que utiliza o protocolo 1-Wire para comunicação. Ele é amplamente utilizado em aplicações de medição de temperatura devido à sua precisão, facilidade de uso, baixo custo e alta eficiência energética. Sua faixa de medição é de -55°C a +125°C, possuindo alta precisão de ±0,5°C na faixa de -10°C a +85°C. Em suma, esse sensor converte a temperatura em um valor digital, que será lido pelo nosso microcontrolador. É fulcral destacar que esse sensor é altamente resistente a água, fator indispensável uma vez que o projeto é pensado para ser aplicado em regiões de campo aberto, isto é, sujeitas a chuvas e orvalho. Sua conexão é feita por um cabo de dados associado a resistor para garantir a estabilidade do sinal que é ligado ao pino GPIO16 da BitDogLab, além de mais dois, cabos, um VCC ligado no Pino de alimentação 3.3 V e o fio de aterramento que será ligado no GND da placa.
- **Cabo USB de Dados (1 Unidade):** Esse cabo é o responsável pela conexão da BitDogLab com o computador, essa faz toda a comunicação e inserção do software em nosso microcontrolador.
- **Protoboard 400 Pontos (1 Unidade):** Conhecida também como matriz de contato, esse dispostivo nos permite construir circuitos de forma temporária e excelente instrumento para diversos testes. Vamos utilizar a mesma para facilitar as conexões entre o sensor DS18B20 e o resto de nosso circuito. Vale ressaltar que há modelos de protoboard com menores quantidades de pontos que também poderiam ser utilizadas, já que em nosso projeto não iremos utilizar nem 20 desses pontos.
- **Jumpers (5 Unidades: 3 macho/fêmea e 2 macho/macho):** Jumpers são utilizados para fazer ligações entre os componentes sem precisar fazer uso de soldas. Vamos utilizar os mesmos para fazer a comunicação entre a protoboard e a BitDogLab.
- **Resistor *pull-up* de 4,7 $k\Omega$ (1 Unidade)**: Este será utilizado na ligação entre o cabo de dados do do sensor DS18B20 e a BitDogLab. Esse resistor é fundamental para manter a estabilidade do sinal.

2.4 Descrição da Pinagem utilizada e Circuito Completo do Hardware

A baixo, veremos como foi feita a distribuição do conjunto de pinos utilizados de forma a garantir uma comunicação eficiente, com baixo consumo de energia e alta confiabilidade, tornando o CerradoBitFire uma solução robusta para monitoramento ambiental e alerta de incêndios em regiões críticas do Cerrado.

2.4.1 Pinos de Comunicação I2C:

- **Pino SDA (GPIO 14):** Utilizado para a comunicação I2C (Inter-Integrated Circuit) com o display OLED. O protocolo I2C permite a comunicação eficiente entre múltiplos dispositivos utilizando apenas duas linhas de sinal (SDA e SCL).
- **Pino SCL (GPIO 15):** Responsável pelo clock do barramento I2C, garantindo a sincronização da troca de dados entre a Raspberry Pi Pico e o display OLED.

2.4.2 Pino de Comunicação OneWire:

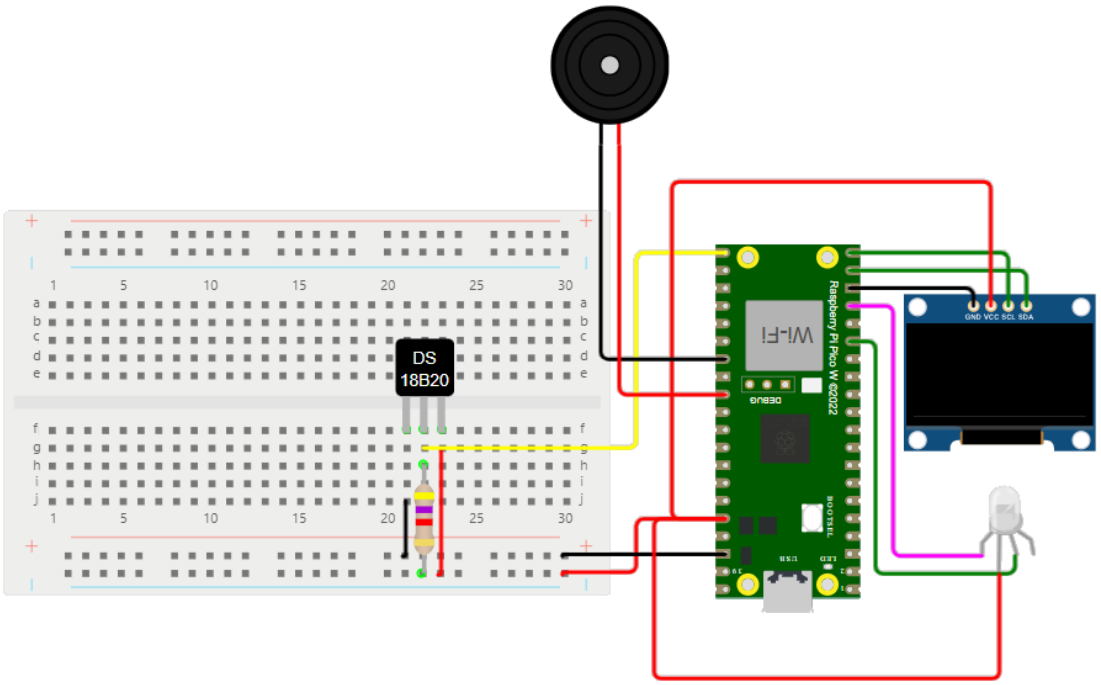
- **Pino GPIO 16:** Responsável pela comunicação OneWire com o sensor de temperatura DS18B20. O protocolo OneWire é um barramento serial que utiliza apenas um fio de dados, além dos pinos de alimentação (VCC e GND). O resistor de pull-up é necessário para estabilizar a linha de dados.

2.4.3 Pinos de Controle dos LEDs (PWM):

- **PINOS GPIO 11(Verde) e GPIO 13(Vermelho):** Os LEDs são controlados por modulação por largura de pulso (PWM - Pulse Width Modulation). Os LEDs operam em intensidades variáveis, o duty cycle dos sinais PWM para representar o nível de perigo: **Temperatura Baixa**, LED verde brilhante, LED vermelho apagado. **Temperatura Intermediária**, Transição gradual entre verde e vermelho. **Temperatura crítica**, LED vermelho brilhante, indicando alto risco de incêndio.

2.4.4 Pino de Controle do Buzzer (PWM):

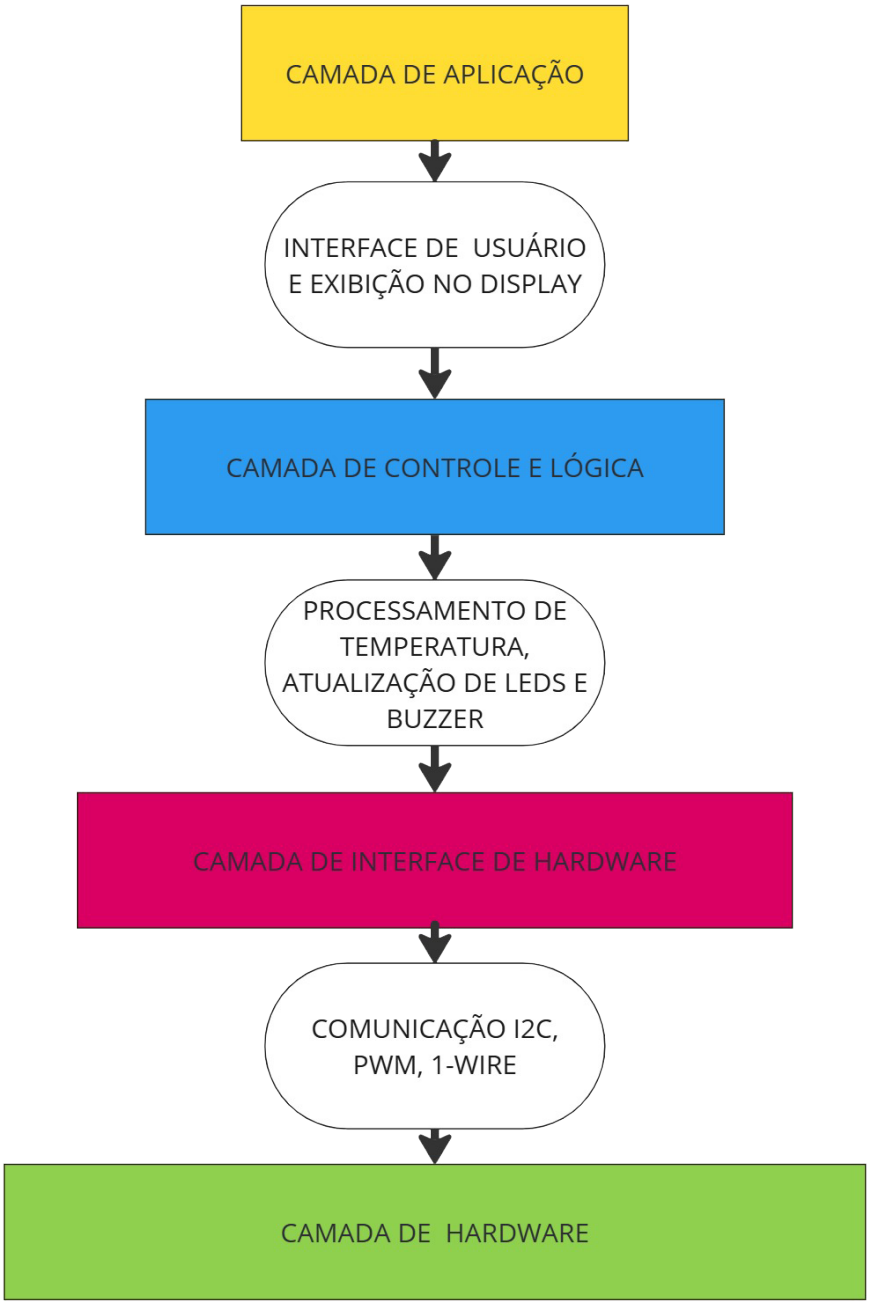
Pino GPIO 21: Conectado ao buzzer ativo, sendo controlado por PWM para modular a frequência e intensidade do som gerado. Em caso de temperatura crítica o código emite uma saída para este pino acionando um padrão de alerta sonoro mais agressivo.



3 Software

Todo o código e as bibliotecas para o desenvolvimento desse projeto estão todas hospedadas no GitHub com domínio público para consultas e modificações no link: https://github.com/Zazamartins/CerradoBitFire_1.0

3.1 Diagrama de camadas de Software



3.2 Descrição das Funcionalidades de cada Bloco

3.2.1 Camada de Hardware:

Engloba os componentes físicos do sistema, como a placa de desenvolvimento BitDogLab (baseada na Raspberry Pi Pico W), sensore DS18B20, LEDs RGB, buzzer e o display OLED. Essa camada é responsável por processar sinais elétricos e transformar interações físicas e analógicas em dados utilizáveis pelo software.

3.2.2 Camada de Interface de Hardware:

Responsável pela comunicação entre o software e os componentes físicos. Essa camada inclui a implementação da comunicação I2C para o display OLED (ssd1306.h), a comunicação OneWire para o sensor DS18B20 (ds18b20/temp.h) e a geração de sinais PWM para LEDs e buzzer.

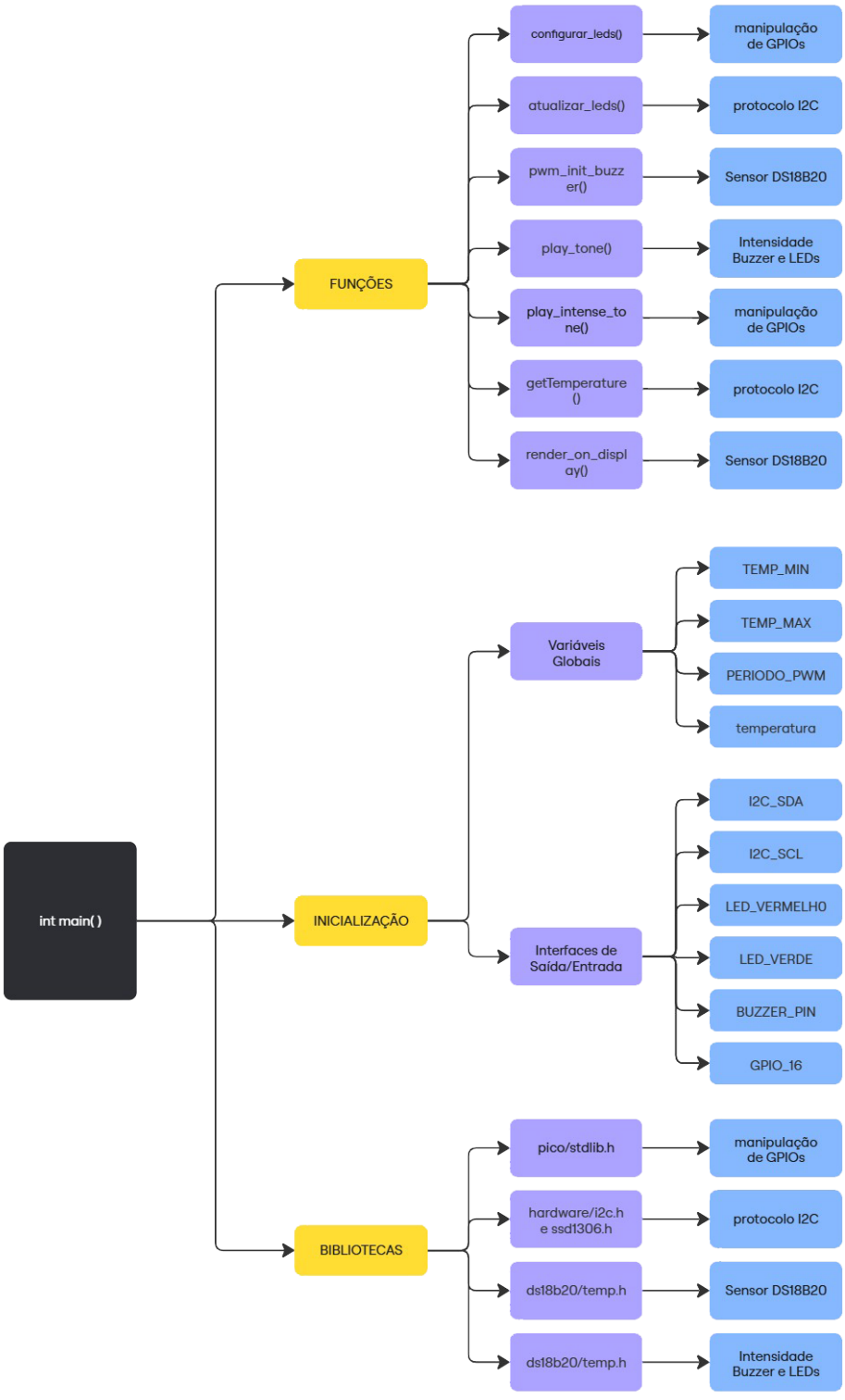
3.2.3 Camada de Controle e Logística:

Processa os dados brutos dos sensores e define a lógica para exibição e acionamento dos sinais de alerta, mplementa a lógica de controle para os LEDs RGB, ajustando suas cores de acordo com os níveis de temperatura além de conter funções para mapeamento da temperatura e conversão de sinais digitais em comandos físicos.

3.2.4 Camada de Aplicação:

Responsável pela execução principal do programa, coordenando as interações entre sensores, atuadores e a interface com o usuário. Monitora continuamente a temperatura ambiente, decide quando ativar alertas e exibe as informações no display OLED, isto é, garante que o sistema funcione conforme os requisitos estabelecidos para monitoramento de incêndios.

3.3 Fluxograma Software



3.4 Bibliotecas

O código faz uso de diversas bibliotecas para as mais diversas comunicações entre os periféricos e o microcontrolador. A biblioteca *pico/stdlib.h* é a base do sistema, fornecendo funções essenciais para a inicialização e controle do Raspberry Pi Pico. Ela inclui, por exemplo, a função *stdio_ init_ all()*, que inicializa a comunicação serial e outros periféricos básicos. Já a biblioteca *pico/binary_ info.h* é usada para incluir informações binárias no firmware, o que facilita a depuração e a identificação do programa durante a execução.

Para controlar o display OLED SSD1306, o software utiliza a biblioteca *ssd1306.h*. Essa biblioteca oferece funções para inicializar o display, renderizar texto e atualizar a tela. A comunicação com o display é feita via I2C, e a biblioteca *hardware/i2c.h* é responsável por configurar os pinos SDA e SCL, além de definir a frequência do clock I2C. Resistores de pull-up são habilitados nos pinos I2C para garantir uma comunicação estável.

O controle dos LEDs e do buzzer é realizado por meio da biblioteca *hardware/pwm.h*, que permite configurar e gerenciar o PWM (Pulse Width Modulation). O PWM é usado para ajustar a intensidade dos LEDs e gerar tons no buzzer. A biblioteca *hardware/clocks.h* é utilizada para configurar o clock do sistema, necessário para cálculos de frequência no PWM.

Por fim, a leitura da temperatura é feita com o sensor DS18B20, e a biblioteca *ds18b20/temp.h* é responsável por essa tarefa. Ela inclui funções para verificar a presença do sensor e ler a temperatura. Essa biblioteca se comunica com o sensor usando o protocolo 1-Wire, que é implementado diretamente no código.

3.5 Inicialização

O processo começa com a chamada da função *stdio_ init_ all()*, que inicializa a comunicação serial e outros periféricos básicos do Raspberry Pi Pico. Essa função é essencial para garantir que o microcontrolador esteja pronto para interagir com dispositivos externos, como o display OLED e o sensor de temperatura.

Em seguida, os pinos dos LEDs (verde e vermelho) são configurados como saídas PWM. Isso é feito usando a função *gpio_ set_ function()*, que define a função dos pinos como PWM. O período do PWM é configurado com *pwm_ set_ wrap()*, e o PWM é habilitado com *pwm_ set_ enabled()*. Esse processo é repetido para ambos os LEDs, garantindo que eles estejam prontos para serem controlados com base na temperatura.

O buzzer também é configurado como uma saída PWM. O pino do buzzer é inicializado com *gpio_ set_ function()*, e o divisor de clock é ajustado com *pwm_ config_ set_ clkdiv()* para gerar frequências audíveis. O PWM é então inicializado com *pwm_ init()*, e o buzzer é desligado inicialmente com *pwm_ set_ gpio_ level(pin, 0)*.

A comunicação com o display OLED é configurada em seguida. Os pinos I2C (SDA e SCL) são definidos com *gpio_ set_ function()*, e resistores de pull-up são habilitados para garantir uma comunicação estável. O display SSD1306 é inicializado com a função *ssd1306_ init()*, e a área de renderização é definida usando a estrutura *render_ area*. Essa estrutura especifica as colunas e páginas do display que serão utilizadas para exibir o conteúdo.

O sensor de temperatura DS18B20 é verificado para garantir que a comunicação esteja funcionando corretamente. Isso é feito com a função *presence()*, que verifica se o sensor

está presente no pino especificado. Se a comunicação falhar, uma mensagem de erro é exibida. Caso contrário, o sistema está pronto para ler a temperatura.

Por fim, o buffer de memória para o display OLED é inicializado com *memset()*, e o comprimento do buffer é calculado com *calculate_render_area_buffer_length()*. Esse buffer é onde os dados gráficos são armazenados antes de serem renderizados no display.

3.6 Configuração dos Registros

A configuração dos registros é uma etapa crítica para garantir o funcionamento adequado dos periféricos. No caso do PWM, os registros são configurados para controlar a intensidade dos LEDs e gerar tons no buzzer. A função *pwm_set_wrap()* define o valor máximo do contador PWM, enquanto *pwm_set_gpio_level()* ajusta o duty cycle para controlar a intensidade dos LEDs ou o volume do buzzer.

Para o I2C, os registros são configurados para definir a frequência do clock e habilitar a comunicação. A função *i2c_init()* inicializa o barramento I2C com a frequência desejada, e *gpio_set_function()* configura os pinos SDA e SCL para a função I2C. Resistores de pull-up são habilitados com *gpio_pull_up()* para garantir uma comunicação estável.

No caso do sensor DS18B20, os registros são configurados para ler a temperatura. A função *presence()* verifica se o sensor está presente, e *getTemperature()* realiza a leitura da temperatura. Essas funções interagem diretamente com os registros do sensor para obter os dados necessários.

3.7 Estrutura e Formato de Dados

A estrutura e o formato dos dados são essenciais para o funcionamento do sistema. O buffer de memória para o display OLED, representado pelo array *ssd*, armazena os dados gráficos que serão renderizados na tela. Esse buffer é inicializado com *memset()* e seu comprimento é calculado com *calculate_render_area_buffer_length()*.

A estrutura *render_area* define a área de renderização no display OLED. Ela contém os campos *start_column*, *end_column*, *start_page* e *end_page*, que especificam as colunas e páginas do display que serão utilizadas. Essa estrutura é usada para calcular o comprimento do buffer e renderizar o conteúdo na tela.

Os dados de temperatura são armazenados em uma variável do tipo *float* e convertidos para uma string com *snprintf()* antes de serem exibidos no display. A string é então centralizada na tela com base no comprimento do texto.

3.8 Organização da Memória

A organização da memória é cuidadosamente planejada para garantir o funcionamento eficiente do sistema. O buffer de memória para o display OLED, representado pelo array *ssd*, é armazenado na RAM. Esse buffer é onde os dados gráficos são armazenados antes de serem renderizados no display.

As variáveis temporárias, como a temperatura e as strings de texto, também são armazenadas na RAM. O comprimento do buffer é calculado dinamicamente com base na área de renderização, o que permite uma utilização eficiente da memória disponível.

O código e as bibliotecas são armazenados na memória flash do Raspberry Pi Pico. Isso inclui as funções de inicialização, configuração dos registros e controle dos periféricos. A

organização da memória é projetada para maximizar a eficiência e garantir que o sistema funcione de maneira confiável.

3.9 Protocolo de Comunicação

O sistema utiliza dois protocolos de comunicação principais: I2C e 1-Wire. O I2C é usado para comunicação com o display OLED SSD1306. Os pinos SDA e SCL são configurados com *gpio_set_function()*, e a frequência do clock é definida com *i2c_init()*. Resistores de pull-up são habilitados para garantir uma comunicação estável.

O protocolo 1-Wire é usado para comunicação com o sensor de temperatura DS18B20. Esse protocolo é implementado diretamente no código, com funções como *presence()* para verificar a presença do sensor e *getTemperature()* para ler a temperatura. O protocolo 1-Wire é simples e eficiente, ideal para comunicação com sensores de temperatura.

3.10 Formato do Pacote de Dados

formato do pacote de dados varia dependendo do periférico. No caso do display OLED, os dados são enviados como um buffer de bytes, onde cada byte representa um pixel ou um conjunto de pixels. O buffer é renderizado na tela com a função *render_on_display()*.

Para o sensor DS18B20, os dados de temperatura são lidos como um valor de 16 bits e convertidos para um valor de ponto flutuante. Esse valor é então formatado como uma string e exibido no display OLED.

No caso do I2C, os pacotes de dados incluem um endereço de dispositivo, um comando e os dados a serem transmitidos. O display OLED SSD1306 utiliza comandos específicos para configurar a tela e enviar dados gráficos.

4 Execução do Projeto

4.1 Metodologia

O desenvolvimento deste projeto baseia-se na metodologia de Waine Wolf para sistemas embarcados, que compreende as etapas de levantamento de requisitos, especificação, definição da arquitetura, escolha e integração de componentes, além da validação final do sistema. Além disso, adotamos uma abordagem iterativa de refinamento sucessivo, permitindo ajustes durante a implementação para garantir maior eficiência e confiabilidade ao sistema.

O primeira etapa de fato foi a pesquisa de projetos semelhantes nesta área, e nessas buscar se deparar com relativa escassez dessa linha específica de pesquisa, salientando principalmente o fato que a maioria esmagadora das pesquisas de hoje divulgadas na internet são feitas utilizando-se a linguagem python, seja por sua popularidade, facilidade em sua sintaxe ou mesmo por comodidade, mas, que sempre contrasta com o fato que os sistemas embarcados devem ter resposta em tempo real e portanto, eficiência e robustez são elementos indispensáveis ao se pensar em um projeto dessa magnitude. O segundo ponto é que as pesquisas de monitoramento de temperatura estão em sua totalidade voltadas para áreas de agricultura e manejamento do solo, evidentemente de importância

vital para o aumento de produtividade, mas, de fato, o monitoramento de chamas no cerrado é uma lacuna que não foi explorada com tanta profundidade.

Nesse sentido, O projeto visa criar um sistema de monitoramento de temperatura utilizando a placa BitDogLab, que possui um microcontrolador Raspberry Pi Pico W, um display OLED, LEDs RGB, buzzer e um sensor de temperatura DS18B20. O objetivo principal é alertar sobre aumentos anormais de temperatura no Cerrado, facilitando ações preventivas contra incêndios.

4.2 Preparação do Ambiente de Desenvolvimento

Para o desenvolvimento do software, utilizamos o Visual Studio Code (VSCode) como ambiente de desenvolvimento integrado (IDE). A configuração inicial envolveu a instalação de extensões e pacotes essenciais para compilação e depuração do código no Raspberry Pi Pico W. As ferramentas indispensáveis foram o **Pico SDK**, Biblioteca oficial do Raspberry Pi Pico, necessária para manipulação de hardware; **CMake**, Sistema de build utilizado para compilar e organizar o código-fonte; **Extensões do C/C++ para VSCode**, Ferramentas que garantem compatibilidade com a linguagem C e facilitam a escrita e depuração do código.

4.3 Bibliotecas

Constitui talvez o maior desafio do projeto, pelos mesmos motivos citados a cima, devido a popularidade da linguagem Python, os exemplos e modelos de bibliotecas e projetos são feitos quase que exclusivamente nessa linguagem, de maneira que alguns protoclos como o 1-wire implementados em C não são atualizados há algum tempo e nem tão fáceis de encontrarmos suas bibliotecas de modo funcional. Cada uma dessas bibliotecas desempenham um papel fundamental na manipulação dos periféricos e na organização do código, permitindo uma implementação modular e eficiente.

A biblioteca *pico/stdlib.h* fornece funções essenciais para a manipulação de GPIOs, possibilitando a configuração de pinos de entrada e saída, além de rotinas básicas para controle do fluxo de execução. Já a *pico/binary_info.h* é utilizada para depuração e fornecimento de metadados no binário gerado, facilitando a análise do código durante testes e validações.

Para a comunicação com dispositivos externos, a *hardware/i2c.h* desempenha um papel essencial, permitindo a interface com o *display OLED* via protocolo *I2C*, garantindo uma exibição clara e responsiva das informações coletadas pelo sistema. Além disso, a *hardware/pwm.h* é empregada no controle de modulação por largura de pulso (PWM), sendo responsável por ajustar a intensidade dos LEDs e controlar a emissão sonora do *buzzer*, conforme os níveis de temperatura detectados.

A biblioteca *hardware/clocks.h* gerencia os *clocks* internos do microcontrolador, assegurando a correta sincronização entre os periféricos e garantindo a estabilidade do funcionamento do sistema. Para a leitura da temperatura ambiente, a biblioteca *ds18b20/temp.h* foi implementada, permitindo a comunicação com o sensor *DS18B20* via protocolo *OneWire*, utilizando um único pino de dados para transmissão das medições.

Por fim, para a exibição das informações no *display OLED*, utilizamos a biblioteca *inc/ssd1306.h*, que facilita a renderização dos caracteres na tela e possibilita o controle do

conteúdo exibido em tempo real. Dessa forma, a combinação dessas bibliotecas permite uma implementação otimizada e confiável do sistema, garantindo que todos os componentes operem de maneira eficiente e integrada.

4.4 Desenvolvimento do Software

A implementação do código seguiu uma abordagem top-down, onde primeiramente foram configurados os periféricos e, em seguida, as funcionalidades de leitura, exibição e alerta foram integradas de maneira modular. Essa organização garante um desenvolvimento estruturado e facilita futuras modificações ou expansões no sistema.

A estrutura do código foi dividida em camadas bem definidas para modularizar o projeto e permitir um melhor controle das interações entre hardware e software. A Camada de Hardware é responsável pela comunicação direta com os sensores e atuadores, garantindo que os sinais elétricos sejam corretamente processados e transmitidos. A Interface de Hardware compreende os drivers que fazem a ponte entre os componentes físicos e o software, incluindo o controle dos LEDs, buzzer, display OLED e o sensor de temperatura.

Na Camada de Controle e Lógica, os dados de temperatura coletados pelo sensor são processados, e a lógica de alerta é aplicada, determinando o acionamento progressivo dos LEDs e do buzzer conforme os níveis de temperatura estabelecidos. Por fim, a Camada de Aplicação gerencia a interação final com o usuário, exibindo a temperatura no display e garantindo a execução correta do sistema conforme o fluxo operacional esperado. Essa organização em camadas melhora a manutenção e escalabilidade do software, permitindo ajustes e otimizações sem impactar o funcionamento global do sistema.

4.5 Testes de Integração do Sistema

A fase de testes foi essencial para garantir a confiabilidade e o funcionamento correto do sistema em condições reais de operação. Inicialmente, cada componente foi testado individualmente para verificar sua resposta e desempenho em diferentes condições. O sensor de temperatura foi submetido a testes de precisão, assegurando que as leituras fossem consistentes e confiáveis. Os LEDs e o buzzer foram testados para garantir que os sinais de alerta fossem ativados corretamente conforme a variação da temperatura.

4.6 Discussão dos Resultados

Os resultados obtidos ao longo do desenvolvimento e teste do CerradoBitFire demonstram que o sistema apresenta uma base sólida para monitoramento e alerta de incêndios, com potencial para ser escalonado e implementado em larga escala. A utilização de um sistema embarcado de baixo custo, baseado na BitDogLab, Raspberry Pi Pico W, permite que a solução seja replicada e aplicada em diferentes regiões do Cerrado, onde a incidência de incêndios é alta e os recursos para monitoramento muitas vezes são escassos.

Embora o protótipo atual utilize LEDs e um buzzer para sinalização local, esses elementos são apenas abstrações -dentro das limitações do próprio projeto- de métodos reais de comunicação que podem ser escalonados. A lógica de alerta implementada pode ser facilmente adaptada para ativar transmissões via redes sem fio, como LoRa, Wi-Fi ou GSM, possibilitando a notificação remota de instituições responsáveis pela prevenção e

combate a incêndios, como o Corpo de Bombeiros e órgãos ambientais. Dessa forma, o CerradoBitFire poderia se integrar a uma infraestrutura de monitoramento distribuído, ampliando seu impacto e eficiência.

A viabilidade do projeto é reforçada por sua adaptabilidade às condições adversas do Cerrado. O sensor DS18B20 utilizado no sistema é resistente à água, tornando o dispositivo adequado para operar mesmo em períodos de chuva e alta umidade, garantindo medições consistentes e contínuas. Além disso, o baixo consumo energético do hardware permite que o sistema funcione em locais remotos, utilizando painéis solares ou baterias, eliminando a necessidade de infraestrutura elétrica complexa.

Portanto, os resultados obtidos indicam que o CerradoBitFire é uma solução viável para auxiliar no monitoramento e detecção precoce de incêndios florestais. A escalabilidade do sistema, aliada ao seu baixo custo e facilidade de implementação, o torna uma alternativa promissora para proteção ambiental, permitindo que áreas vulneráveis sejam monitoradas de forma contínua e eficiente. Com aprimoramentos futuros, como integração com redes de sensores e inteligência artificial para análise preditiva, o projeto pode se tornar ainda mais robusto e eficaz na prevenção de desastres ambientais.

4.7 Vídeo de Demonstração

Foi gravado um vídeo de demonstração de como o projeto funciona e este foi hospedado na plataforma do YouTube, no seguinte link:
https://youtu.be/Q-vknaMpwL4

5 Referências

Frequência de Fogo no Brasil, *plataforma Brasil Mapbiomas*

Cerrado perde 1,1 milhão de hectares, *Ipam*

Datasheet DS18B20

Prototipagem de Monitor de Temperatura e Umidade, *repositorio utfpr*

Documento 2451, *lyceumonline.usf*

Medidor de Temperatura com Micropython

Artigo 016 - BTSym18, *Repositório Unicamp*

Trabalho de Graduação - TG_SI

Fogo no Cerrado, *CMBGO*

Área Queimada no Brasil, *Brasil MapBiomas*