GRADUAÇÃO EM INTELIGÊNCIA ARTIFICIAL

GLOBAL SOLUTION – 2º SEMESTRE

Sistema preditivo de incêndios florestais no Cerrado com uso de IA

Integrantes:
Luana Porto Pereira Gomes – RM: 559290
Luma Oliveira – RM: 560146
Priscilla Oliveira – RM: 560562
Paulo Bernardes – RM: 560401

Link do repositório GitHub:
https://github.com/luanaportop/Global_Solution_2Semestre

São Paulo – 2025

Introdução

A intensificação dos eventos climáticos extremos tem se tornado uma das principais preocupações da atualidade, refletindo diretamente nos desafios enfrentados por governos, instituições e populações ao redor do mundo. Enchentes, secas prolongadas, ondas de calor, deslizamentos e incêndios florestais vêm ocorrendo com maior frequência e intensidade, demandando soluções inovadoras que aliem tecnologia, inteligência de dados e planejamento estratégico. Nesse contexto, torna-se essencial o desenvolvimento de ferramentas capazes de prever, monitorar e mitigar os impactos desses eventos, ampliando a capacidade de resposta e prevenção frente às emergências ambientais.

Diante desse cenário, a Global Solution 2025, proposta pela FIAP, lança aos estudantes da área de Inteligência Artificial o desafio de criar soluções digitais fundamentadas em dados reais para o enfrentamento de desastres naturais. O projeto demanda a aplicação integrada de conhecimentos técnicos em lógica, programação, estruturação e análise de dados, com foco na construção de sistemas funcionais que possam gerar impacto social e ambiental positivo. A iniciativa também busca fomentar o trabalho colaborativo, a inovação e o uso de tecnologias emergentes, incentivando a criação de projetos que respondam de forma concreta a problemas complexos do mundo real.

Este relatório apresenta o desenvolvimento de uma solução voltada ao monitoramento e previsão de incêndios florestais no bioma Cerrado brasileiro, fenômeno que se destaca entre os mais recorrentes e destrutivos do país. A escolha do tema foi orientada por dados do portal *Disasters Charter* e aprofundada a partir da análise de informações fornecidas pelo sistema *BDQueimadas*, do Instituto Nacional de Pesquisas Espaciais (INPE). A proposta do grupo consiste na criação de um modelo preditivo baseado em técnicas de machine learning, utilizando uma base de dados consolidada que integra informações sobre focos de calor e variáveis climáticas históricas de 2020 a 2024.

A seguir, serão detaladas as etapas de coleta e tratamento dos dados, a construção da base analítica, os resultados da análise exploratória e as decisões tomadas para o desenvolvimento do modelo, visando contribuir de forma efetiva para a redução dos impactos causados pelas queimadas e para o fortalecimento de estratégias de prevenção e mitigação de riscos no Cerrado.

1. Escolha do tema

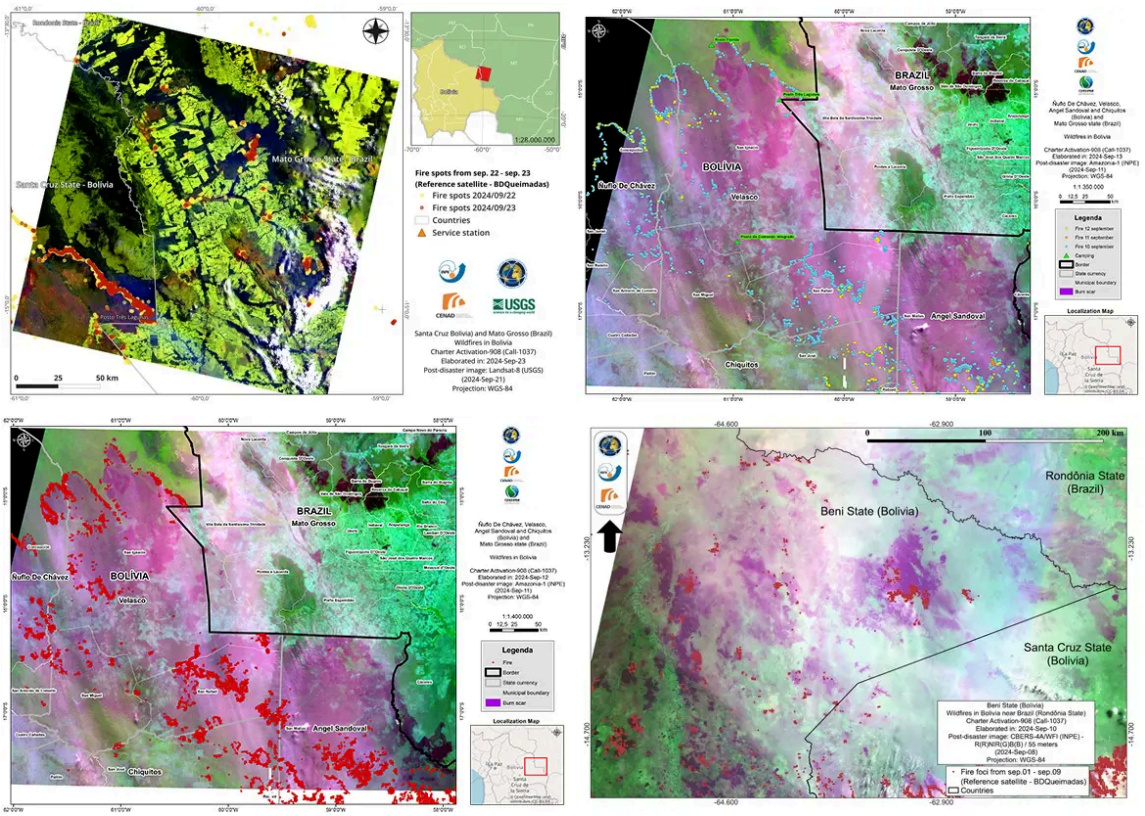


Figura 1 - Imagens do site Disaster Charter.

A escolha do tema teve início com a consulta ao portal Disaster Charter, que reúne dados globais sobre desastres naturais, imagens de satélite e relatórios técnicos. Embora existam diversos registros de incêndios na América do Sul, o único registro relacionado diretamente ao território brasileiro encontrado na plataforma está vinculado a um episódio de fogo ocorrido na Bolívia, em setembro de 2024 (Figura 1). Esse evento levou o governo boliviano a declarar estado de emergência nacional, devido à destruição de mais de 3,8 milhões de hectares de florestas e pastagens.

O relatório técnico desse caso faz menção direta à atuação de órgãos brasileiros, como o INPE (Instituto Nacional de Pesquisas Espaciais), e indica como fonte complementar o portal BDQueimadas, sistema brasileiro de monitoramento de focos de calor desenvolvido pelo INPE. Isso reforça a relevância do Brasil no contexto da resposta regional aos incêndios florestais e aponta para a existência de uma base de dados robusta e confiável, acessível publicamente.

Com base nessa indicação, exploramos o portal BDQueimadas, que disponibiliza séries históricas e conjuntos de dados detalhados sobre queimadas em

todos os biomas brasileiros. A partir da análise dessas informações, decidimos focar especificamente no bioma Cerrado, que tem sido sistematicamente o mais afetado por queimadas nos últimos anos, concentrando milhares de focos ativos anualmente.

O Cerrado é reconhecido por sua alta biodiversidade e importância ecológica, mas enfrenta forte pressão devido ao avanço do desmatamento, às práticas agropecuárias e às mudanças climáticas. Além disso, o portal BDQueimadas integra o Programa Cerrado, uma iniciativa de cooperação entre os governos do Brasil e do Reino Unido, com apoio do Banco Mundial, voltada à mitigação dos efeitos das mudanças climáticas e à gestão sustentável dos recursos naturais da região. Essa conexão entre fontes internacionais e iniciativas nacionais reforça a relevância da temática escolhida e fundamenta a proposta do projeto com base em dados reais e de impacto concreto.

2. Coleta e organização dos dados

Para a construção da base de dados do projeto, optamos por levantar informações referentes ao período de 2020 a 2024, considerando que esse recorte de cinco anos possibilita uma análise mais atualizada e relevante dos padrões de ocorrência de queimadas no Cerrado brasileiro. Os dados foram extraídos diretamente da plataforma BDQueimadas, mantida pelo INPE, que oferece séries históricas detalhadas sobre focos de calor em território nacional.

Com a base de dados inicial organizada, realizamos um filtro por município, de forma a identificar as localidades mais afetadas por incêndios no período analisado. O resultado revelou os 10 municípios com maior número de registros de queimadas, listados a seguir:

Município	Número de Registros
LAGOA DA CONFUSÃO - TO	175.652
MIRADOR - MA	128.756
PIUM - TO	121.768
FORMOSO DO ARAGUAIA - TO	113.472
BALSAS - MA	108.897
COCALINHO - MT	99.364
FORMOSA DO RIO PRETO - BA	90.093
ALTO PARNAÍBA - MA	87.686
PORTO MURTINHO - MS	84.002
SÃO DESIDÉRIO - BA	82.916

Com os municípios mais afetados identificados, iniciamos uma segunda etapa de coleta de dados, voltada às condições climáticas diárias nessas regiões ao longo do mesmo intervalo temporal (2020 a 2024). Para isso, utilizamos a API "Historical Weather" da plataforma Open-Meteo, que oferece dados históricos de clima de forma gratuita e estruturada.

As variáveis climáticas coletadas para cada município foram:

- Temperatura máxima (temp_max)

- Temperatura mínima (temp_min)
- Temperatura média (temp_med)
- Umidade mínima (umidade_min)
- Umidade média (umidade_med)
- Umidade máxima (umidade_max)
- Velocidade máxima do vento (vento_max)
- Data (data)
- Estado e cidade

Por fim, realizamos um *merge* entre os dados de queimadas e os dados climáticos, utilizando como chave de junção as colunas de município e data, obtendo assim um dataframe consolidado que reúne tanto os registros de focos de calor quanto as condições ambientais associadas a cada um deles. Essa base integrada serve como insumo para as próximas etapas de análise e desenvolvimento da solução com técnicas de machine learning.

3. Análise exploratória de dados

A análise exploratória realizada com o conjunto de dados evidenciou aspectos fundamentais para o desenvolvimento do modelo de previsão de risco de queimadas. Com mais de um milhão de registros, o dataset oferece uma base estatística robusta que permite identificar correlações relevantes entre variáveis ambientais e a ocorrência de risco de fogo.

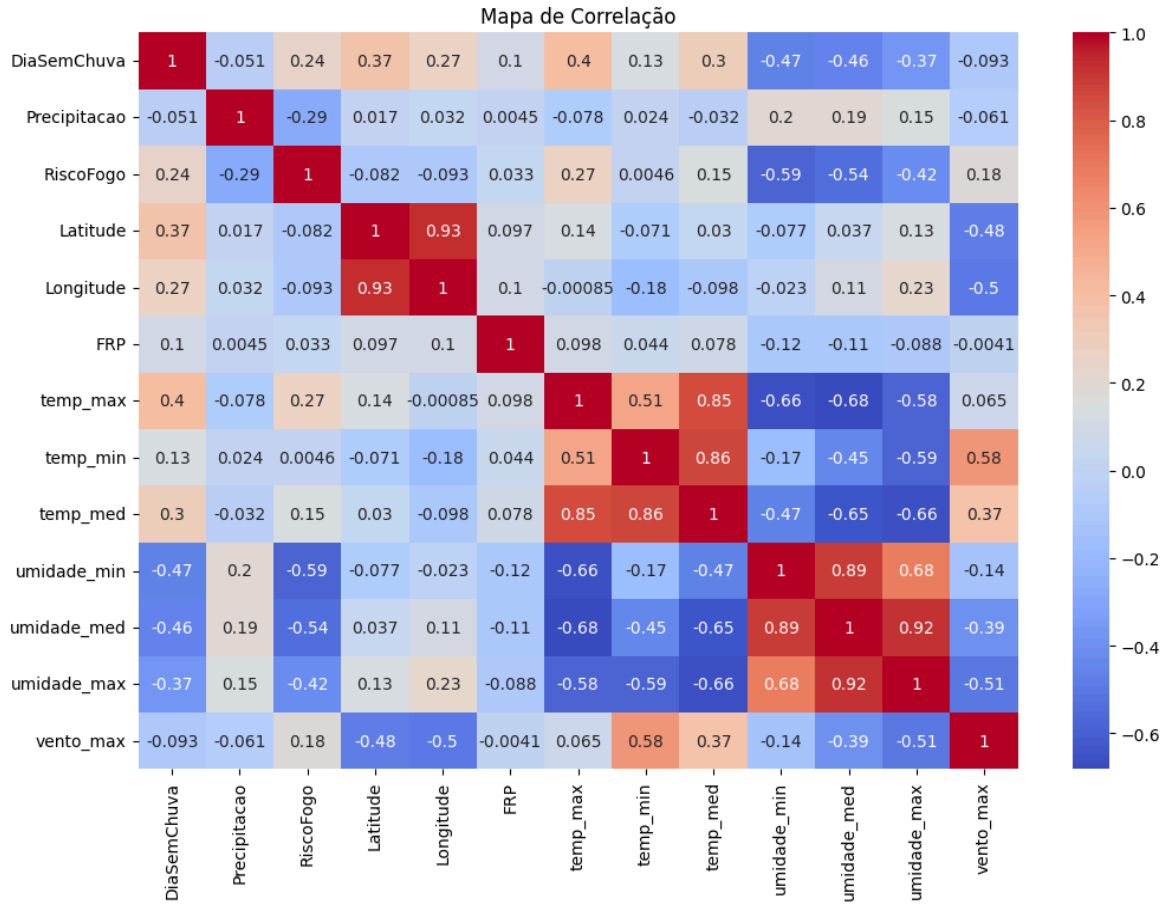


Figura 2 - Matriz de confusão.

A matriz de confusão (Figura 2) gerada a partir dos dados indica que as variáveis de temperatura (máxima, mínima e média) possuem correlação moderada a forte com a variável de risco de fogo (RiscoFogo), sendo a temperatura máxima a que apresenta a associação mais significativa. Esse resultado aponta que o aumento das temperaturas está diretamente relacionado ao incremento do risco de queimadas. Em contrapartida, a variável umidade mínima apresenta uma correlação negativa forte com o risco, o que corrobora com a compreensão de que ambientes

mais secos são mais suscetíveis à propagação do fogo. A umidade média e a umidade máxima também apresentam correlação negativa, embora um pouco mais moderada.

Além disso, observou-se que a variável vento máximo tem uma correlação positiva moderada com o risco de fogo, sugerindo que ventos mais intensos contribuem para o espalhamento das chamas. A variável precipitação, por sua vez, apresenta uma correlação fraca e negativa com o risco, o que sugere que, no curto prazo, a ocorrência de chuvas não impacta de forma significativa na redução do risco de queimadas. Ainda assim, é importante considerar seu efeito no contexto mais amplo da umidade do solo ao longo do tempo.

A análise também revelou correlações espaciais relevantes, como a forte relação entre latitude e longitude, que se refletem em padrões geográficos de incidência de queimadas. Esses padrões podem estar relacionados às características climáticas e ambientais específicas de determinadas regiões. A análise temporal dos dados destacou variações sazonais nas temperaturas máximas, revelando períodos do ano com maior propensão à ocorrência de queimadas. Tais informações são relevantes para a calibragem do modelo de machine learning, permitindo que ele seja mais sensível a essas oscilações ao longo do tempo.

Outro ponto importante da análise foi a contagem de registros com valores de RiscoFogo superiores a 0.7, os quais foram considerados eventos críticos. A frequência com que esses eventos ocorrem reforça a necessidade de um sistema automatizado de alerta que permita respostas rápidas e precisas em situações de alto risco.

Dessa forma, as variáveis que mais se destacaram na análise e que serão priorizadas para a modelagem do risco são: temperatura máxima, umidade mínima e vento máximo (Figura 3). Além de sua relevância estatística, essas variáveis podem ser monitoradas em tempo real por meio de sensores integrados ao microcontrolador ESP32, possibilitando a construção de um sistema inteligente e responsivo para o monitoramento e combate a queimadas.

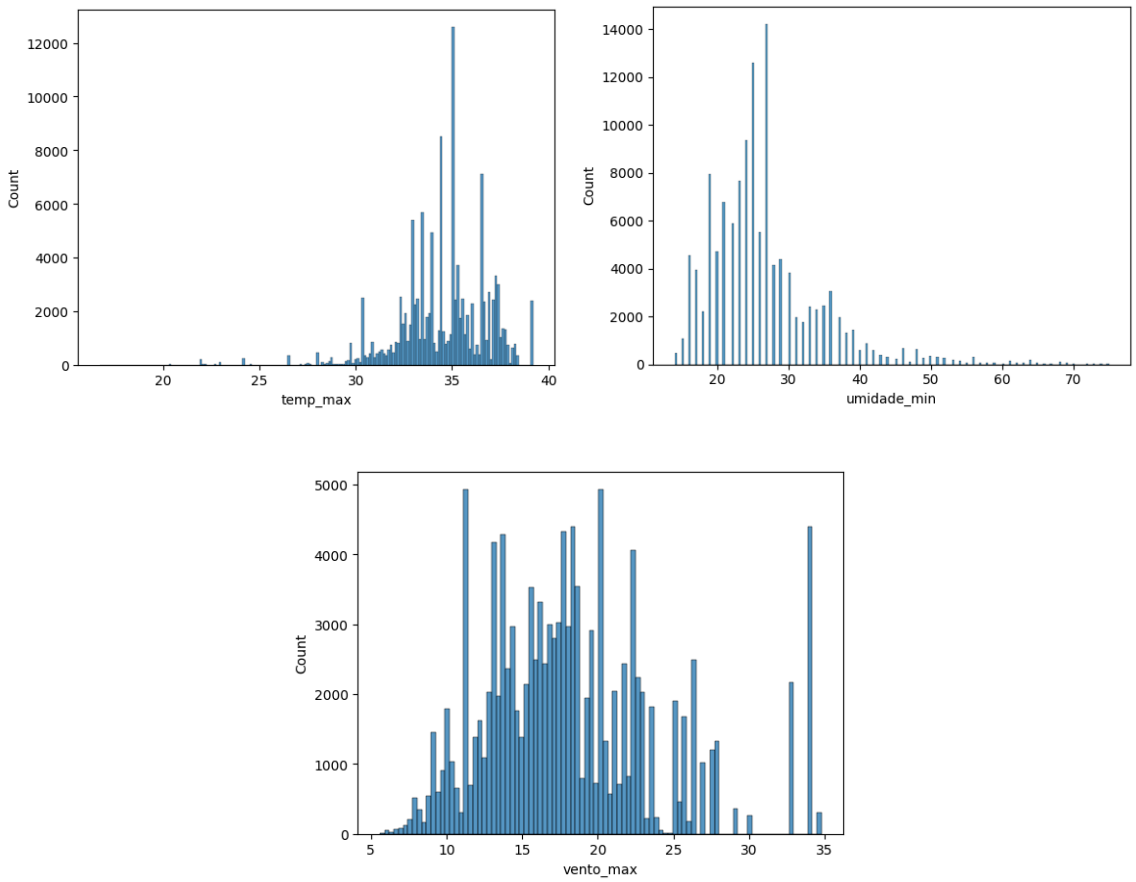


Figura 3 - Gráficos de temperatura máxima, umidade mínima e vento máximo.

4. Modelagem com Machine Learning

Após a análise exploratória e o tratamento do conjunto de dados, partimos para a construção de modelos de Machine Learning com o objetivo de prever o risco de queimadas com base em variáveis ambientais como temperatura, umidade, vento e precipitação. A variável-alvo do modelo foi *RiscoFogo*, um indicador contínuo de risco.

Pré-processamento

Antes da modelagem, foram realizadas etapas importantes de preparação dos dados:

- **Seleção de variáveis relevantes** com base na análise de correlação: *temp_max, temp_min, umidade_min, umidade_med, vento_max e Precipitacao*.

- **Padronização** das variáveis numéricas com *StandardScaler*, garantindo que cada feature tenha média 0 e desvio padrão 1.

- **Tratamento de valores faltantes** com imputação da média utilizando *SimpleImputer*.

- **Divisão dos dados** em conjuntos de treino e teste (80/20), assegurando a validação do desempenho dos modelos.

Modelos avaliados

Três algoritmos foram utilizados e comparados:

1. Random Forest Regressor

Desempenho:

- MAE: 0.0198
- R^2: 0.816

Análise:
O modelo Random Forest apresentou o melhor desempenho entre os modelos testados. A alta acurácia (R^2 de 0.816) e o erro absoluto médio baixo mostram que o modelo consegue prever com grande precisão o risco de queimadas. Isso o torna a

melhor opção para integrar ao sistema automatizado de monitoramento e alerta.

2. Gradient Boosting Regressor

Desempenho:

- MAE: 0.0284
- R^2: 0.7297

Análise:
Embora apresente um bom desempenho, ficou abaixo do Random Forest. No entanto, é um modelo robusto, eficiente e que pode ser considerado em contextos específicos onde interpretabilidade e regularização sejam importantes.

3. MLPRegressor (Rede Neural Artificial)

Desempenho:

- MAE: 0.0320
- R^2: 0.6847

Análise:
A rede neural mostrou-se promissora, mas com desempenho inferior aos modelos baseados em árvores. O tempo de treinamento e o risco de overfitting podem torná-la menos indicada para aplicações em tempo real com dados ambientais.

Comparativo dos modelos

Modelo	**MAE**	**R^2**
Random Forest	0.0198	0.816
Gradient Boosting	0.0284	0.7297
MLPRegressor (RN)	0.0320	0.6847

A comparação reforça a escolha do **Random Forest** como modelo ideal para prever o risco de queimadas neste projeto. Além do desempenho superior, sua estrutura permite interpretabilidade e boa generalização.

A etapa de Machine Learning reforçou os achados da análise exploratória: temperatura máxima, umidade mínima e vento máximo são variáveis críticas para o aumento do risco de queimadas. O modelo final será acoplado ao sistema de monitoramento com ESP32, permitindo a coleta de dados em tempo real e a emissão de alertas baseados em previsões confiáveis.

5. Sistemas de monitoramento de risco de incêndios com ESP32

Apresentamos aqui o desenvolvimento de dois protótipos de sistemas embarcados utilizando o microcontrolador ESP32, voltados à detecção de risco de incêndios no bioma Cerrado brasileiro. Cada sistema foi implementado e simulado na plataforma Wokwi, utilizando sensores e componentes eletrônicos para avaliar diferentes condições ambientais que favorecem a ocorrência de incêndios florestais.

5.1 Sistema 1 – Simulação de cidades com dados climáticos históricos

Este circuito simula um monitor climático (Figura 4) baseado em dados reais dos últimos cinco anos de 10 cidades do Cerrado. Ele avalia temperatura, umidade e vento, indicando risco de incêndio sempre que temperatura > 35 °C e umidade < 30%.

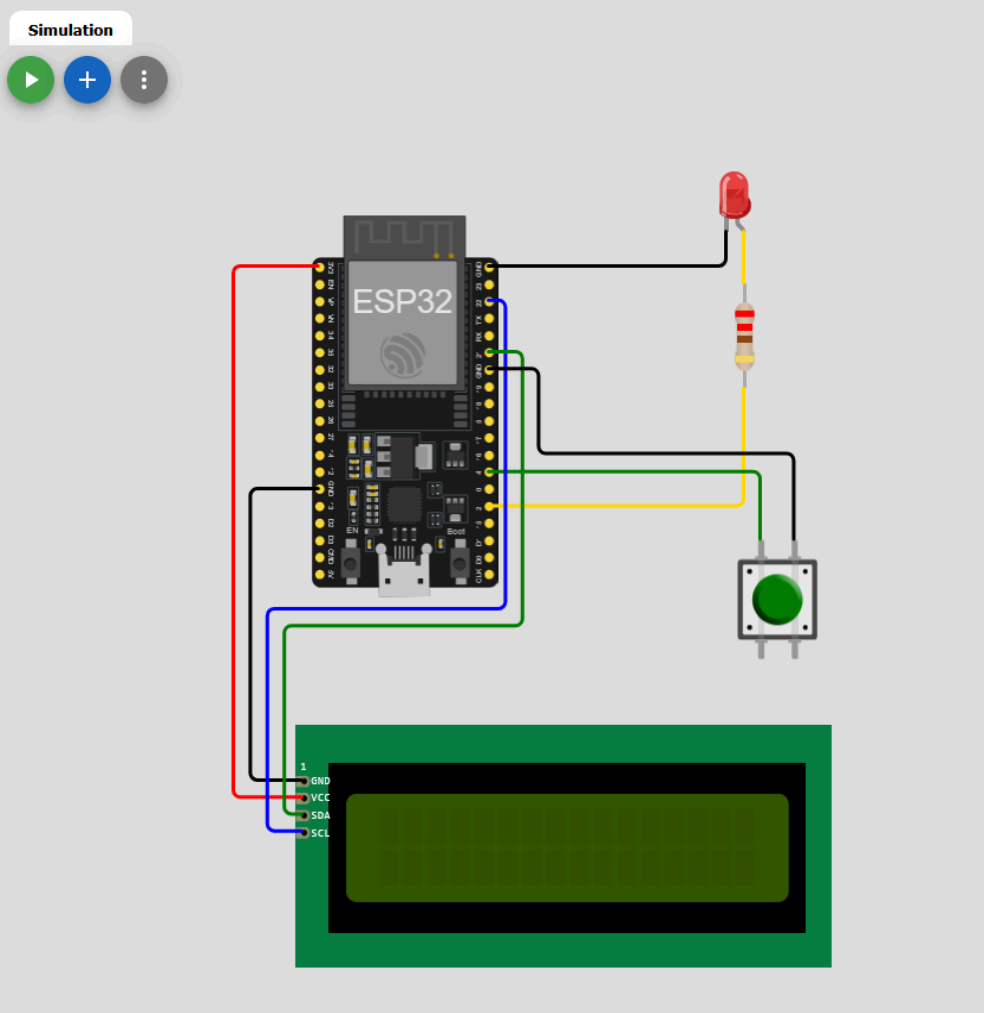


Figura 4 - Sistema 1: monitor climático.

Componentes Utilizados

- ESP32
- Display LCD 16x2 com I2C
- LED (alerta de risco)
- Botão (para alerta manual)

Funcionamento

- O sistema exibe dados de cada cidade sequencialmente.
- Um LED acende automaticamente quando há risco climático.
- Um botão aciona um alerta manual de emergência no sistema.

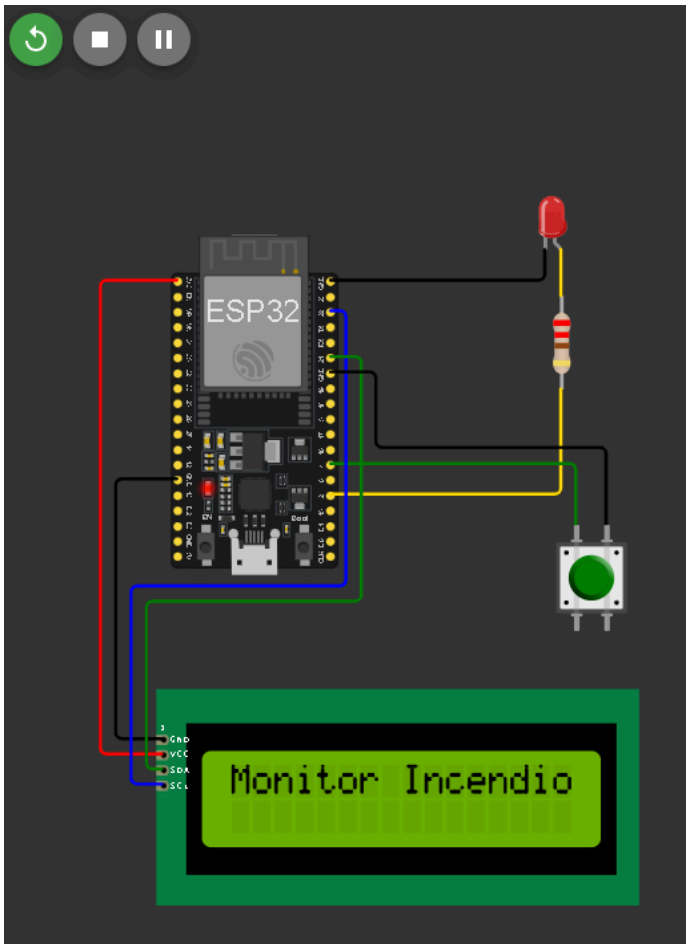
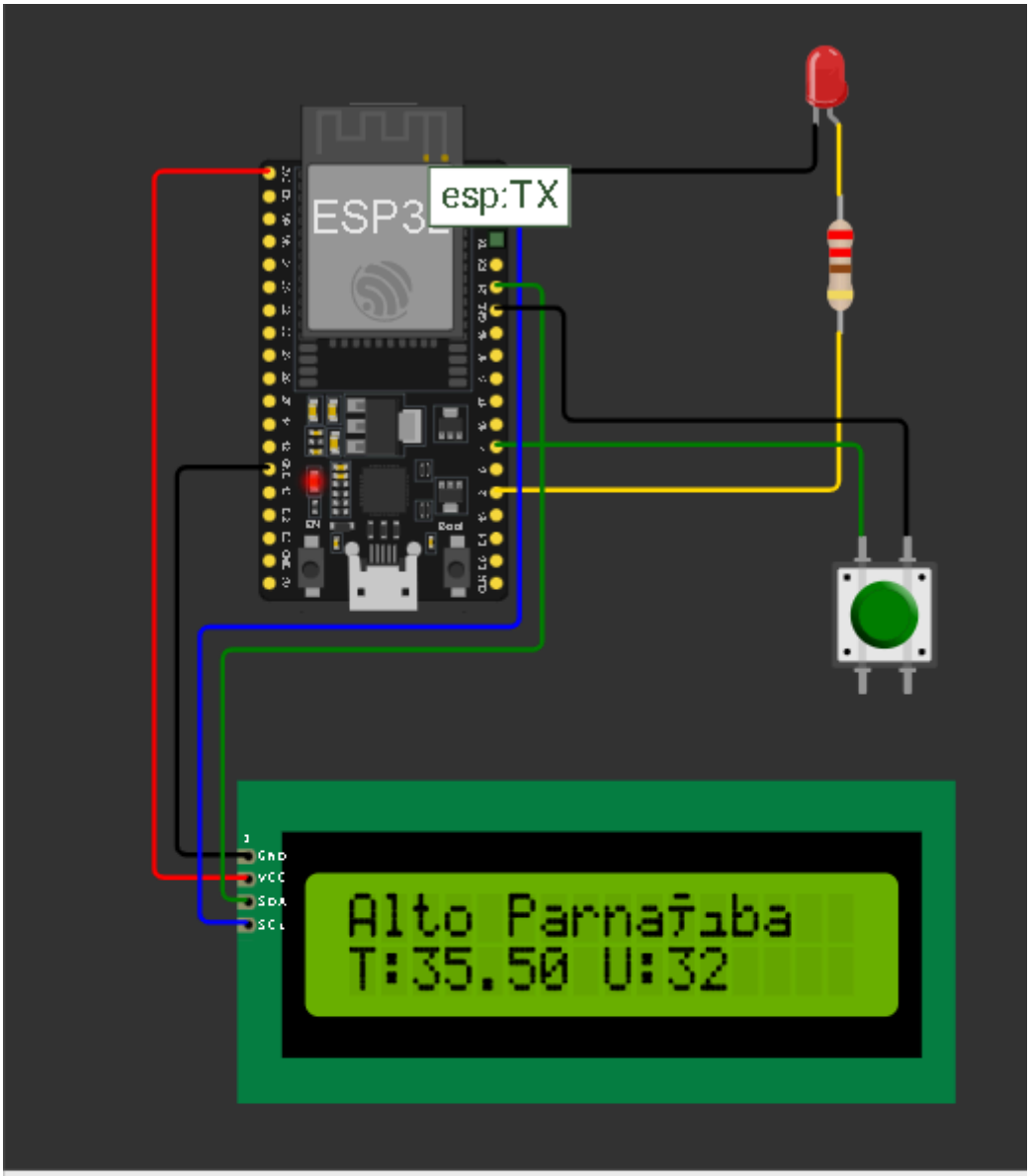
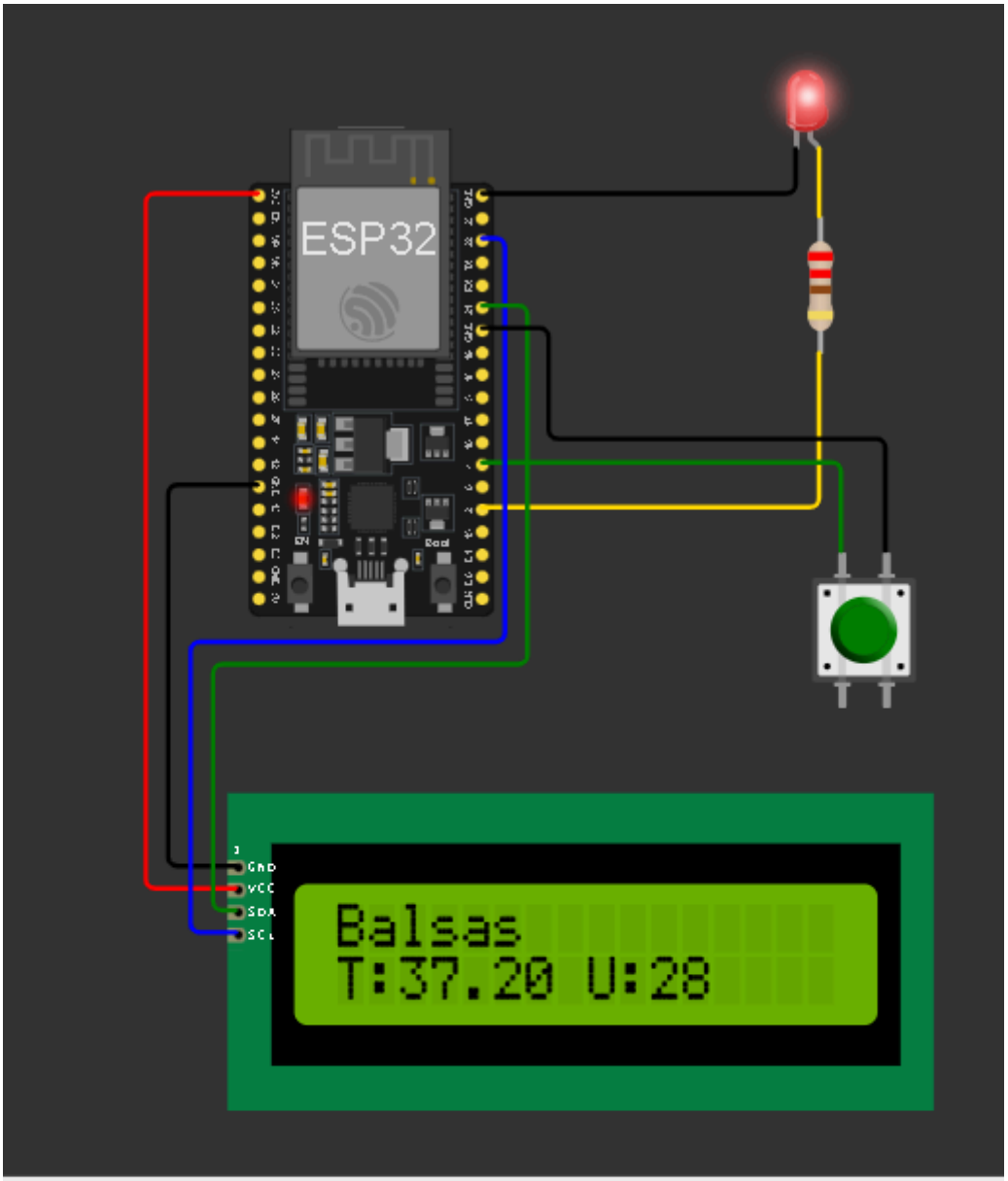


Figura 5 - Iniciando Monitoramento de Incêndio.




```
Cidade: Formosa do Rio Preto
Temp: 36.90
Umidade: 29
Vento: 4.80
🔥 RISCO ALTO
------
Cidade: Alto Parnaíba
Temp: 35.50
Umidade: 32
Vento: 3.70
Risco baixo
```

Figura 6 - Monitoramento: "Risco Baixo" - Alerta por LED desligado.




```
Cidade: Formoso do Araguaia
Temp: 39.00
Umidade: 19
Vento: 6.50
🔥 RISCO ALTO
------
Cidade: Balsas
Temp: 37.20
Umidade: 28
Vento: 5.30
🔥 RISCO ALTO
```

Figura 7 - Monitoramento: "Risco Alto" - Alerta por LED ligado.

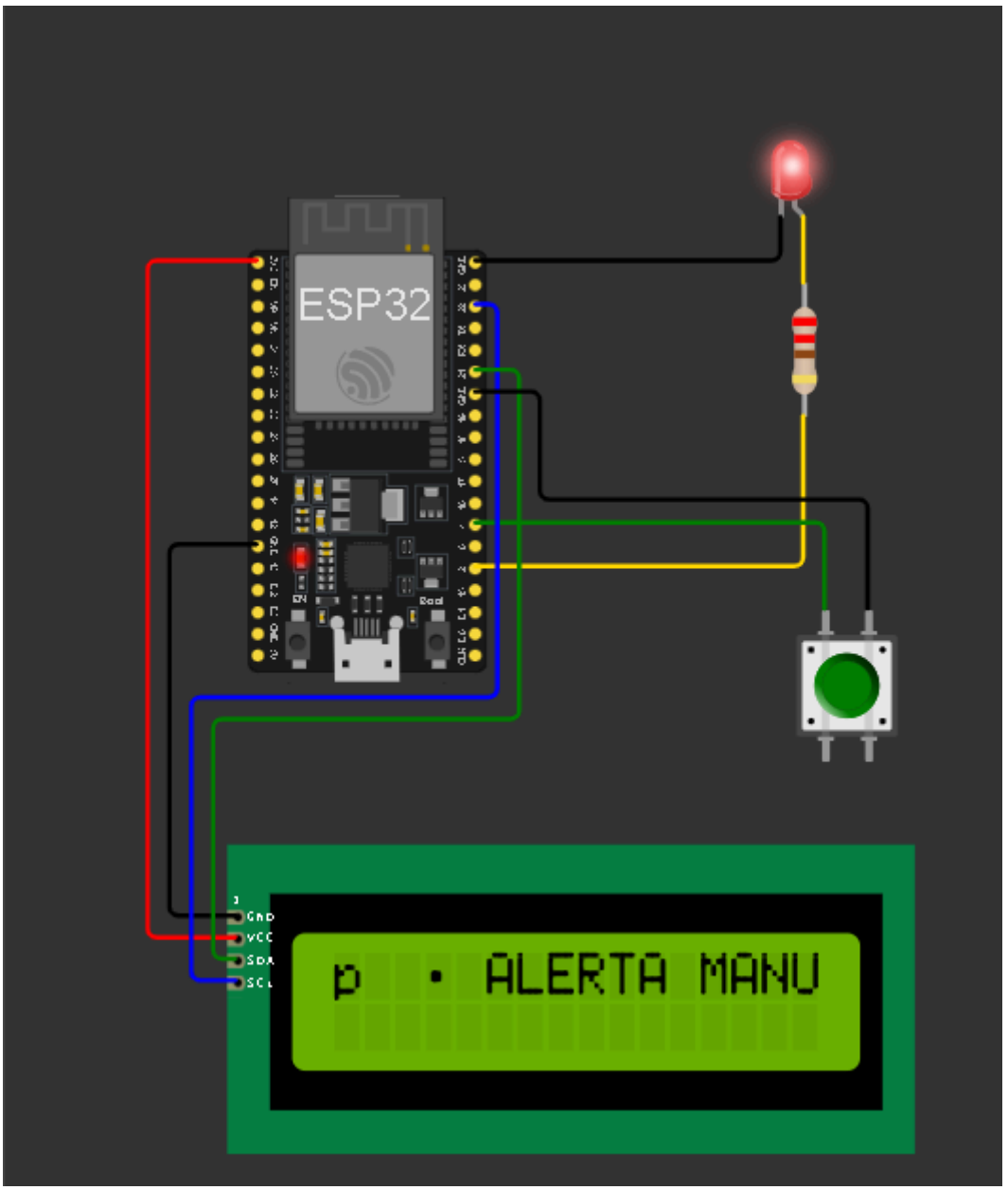


Figura 8 - Acionamento do botão de "Alerta Manual".

5.2 Sistema 2 – Monitoramento com sensores reais (simulados)

Esse sistema monitora condições ambientais (Figura 5) em tempo real usando sensores simulados de:

- Temperatura e umidade (DHT22)
- Fumaça (MQ2)
- Luminosidade (LDR)

A detecção de risco é feita com base em valores de referência definidos para cada sensor.

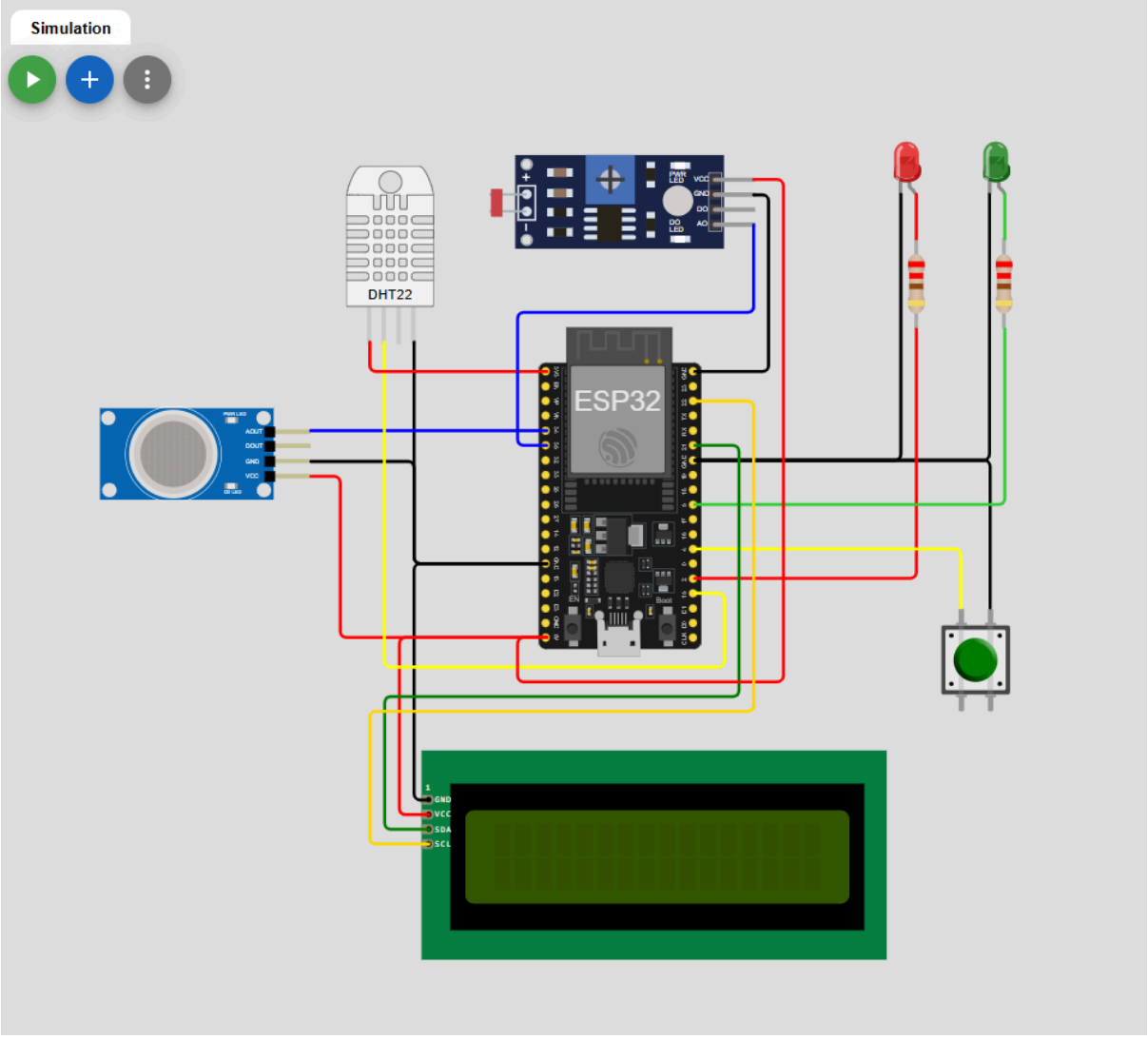


Figura 9 - Sistema 2: monitor de condições ambientais.

Componentes Utilizados

- ESP32
- DHT22
- Sensor MQ2
- Sensor LDR
- LCD 16x2 com I2C
- LED vermelho (alerta de risco)
- LED verde (ambiente seguro)
- Botão (alerta manual)

Funcionamento

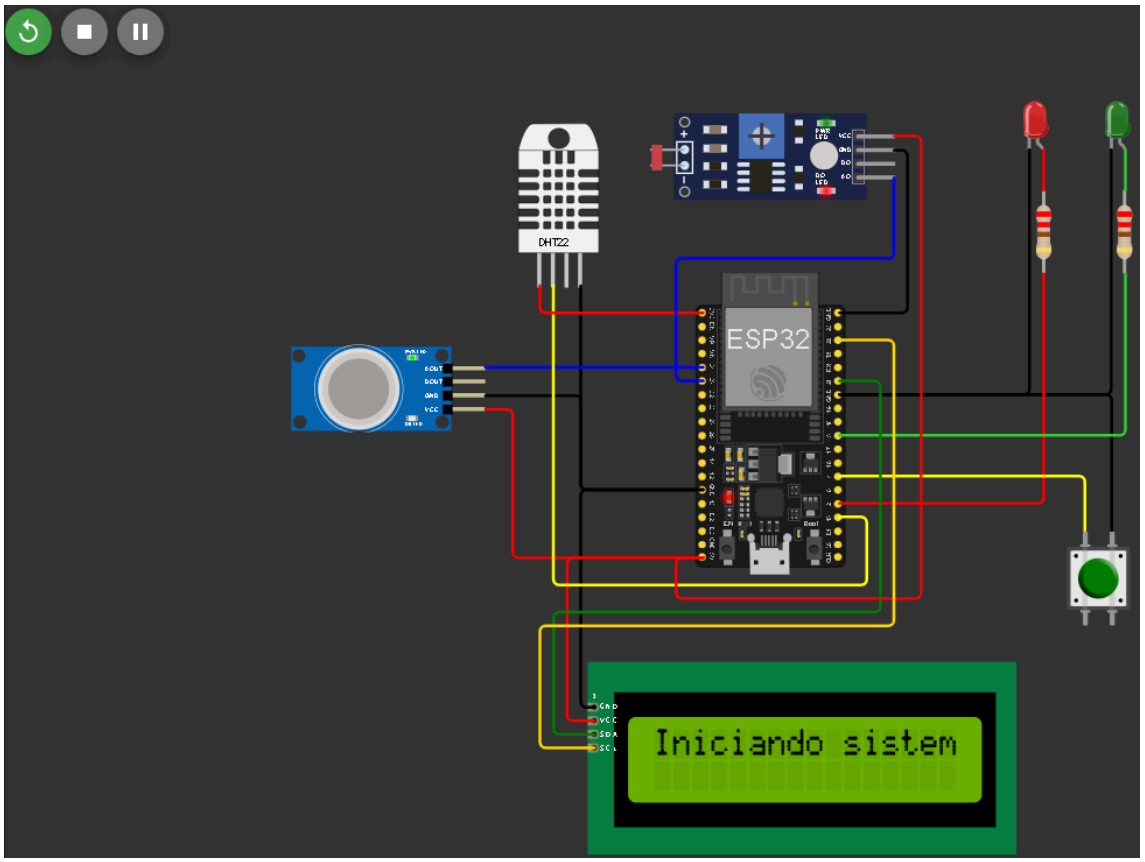
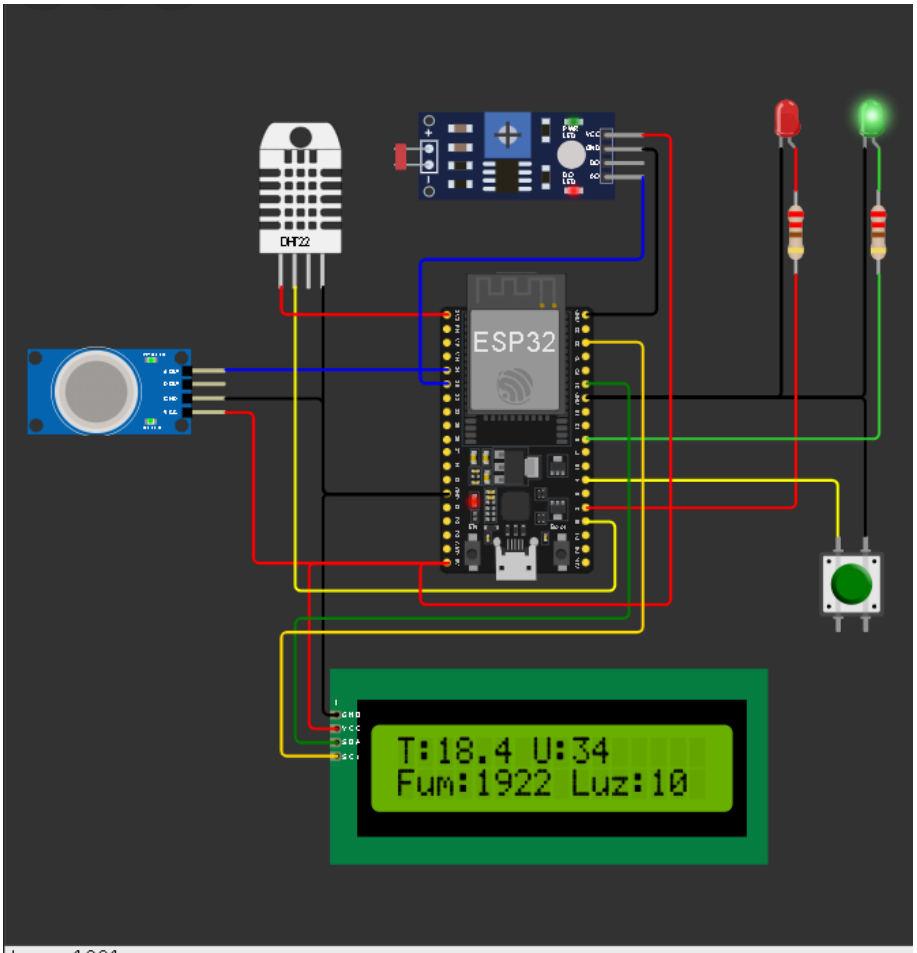


Figura 10 - Sistema iniciando.

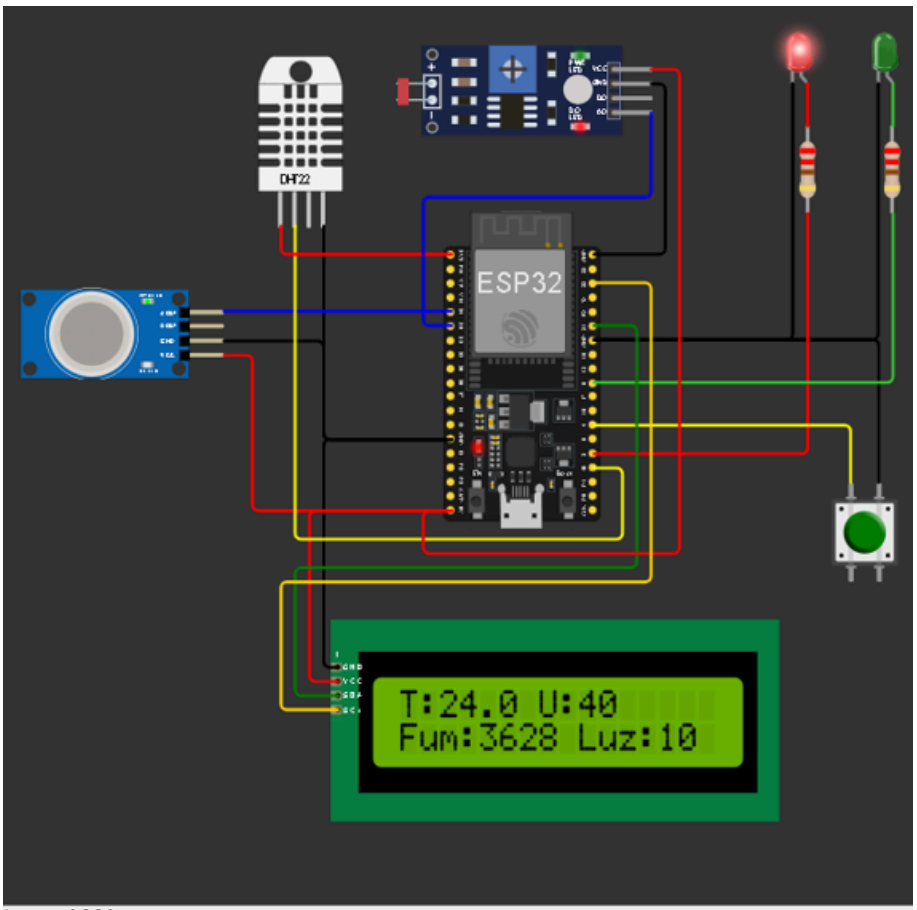
- O sistema mede os níveis ambientais a cada 3 segundos.
- Se alguma das condições de risco for detectada (fumaca alta, pouca luz, calor e seca), o LED vermelho acende.
- Um botão pode acionar o alerta manualmente.
- Os dados são exibidos tanto no LCD quanto no Serial Monitor.




```
Luz: 1001
Ambiente normal

---- LEITURA ----
Temperatura: 18.40 C
Umidade: 34.00 %
Fumaca: 1922
Luz: 1001
Ambiente normal
```

Figura 11 - Simulação de um ambiente "normal".




```
Luz: 1001
>> RISCO DETECTADO <<

---- LEITURA ----
Temperatura: 24.00 C
Umidade: 40.00 %
Fumaca: 3628
Luz: 1001
>> RISCO DETECTADO <<
```

Figura 12 - Simulação de um ambiente de "risco".

5.3 Considerações

Esses dois sistemas representam etapas complementares no desenvolvimento de uma solução de monitoramento de incêndios com baixo custo e alto potencial de aplicação:

- O Sistema 1 simula como o ESP32 pode reagir a dados históricos e fornece um modelo útil para integrar informações de bancos de dados reais.

- O Sistema 2 usa sensores ambientais que já podem ser implementados no campo, sendo um MVP funcional para coleta e resposta automática a dados físicos.

Resultados esperados

Espera-se que o modelo de previsão desenvolvido, com base em técnicas de machine learning, especialmente com o uso do algoritmo Random Forest Regressor, seja capaz de estimar com alta precisão o risco de queimadas no bioma Cerrado. A partir das variáveis ambientais mais relevantes — temperatura máxima, umidade mínima e vento máximo —, o sistema poderá gerar alertas antecipados para eventos críticos, com R^2 de 0.816 e MAE de 0.0198, conforme demonstrado nos testes.

Além da previsão precisa, espera-se que a integração desses dados com sensores em tempo real, por meio de microcontroladores como o ESP32, viabilize um sistema automatizado e responsivo. Tal sistema deverá possibilitar o monitoramento contínuo das condições ambientais e o disparo de alertas preventivos, contribuindo diretamente para estratégias de mitigação e resposta a incêndios florestais.

Conclusão

O projeto demonstrou a viabilidade técnica da construção de uma solução preditiva para o risco de queimadas no Cerrado, a partir da análise integrada de dados climáticos e históricos de focos de calor. A escolha por utilizar dados reais — extraídos de fontes como BDQueimadas e Open-Meteo — permitiu a elaboração de um modelo robusto e ajustado à realidade ambiental do bioma analisado.

A análise exploratória evidenciou as principais variáveis associadas ao risco de fogo e orientou a seleção de atributos relevantes para a modelagem. O modelo Random Forest destacou-se pelo melhor desempenho preditivo, reforçando seu potencial de aplicação em sistemas de alerta precoce. A base integrada, aliada ao uso de sensores e microcontroladores, aponta para a possibilidade concreta de implementação de uma ferramenta eficaz de monitoramento ambiental, com impactos positivos na prevenção de desastres naturais e na preservação do ecossistema do Cerrado.

Anexos

1. Códigos para estrutura do banco de dados

```
import pandas as pd
import os

# Caminho da pasta onde estão os arquivos
caminho_dos_arquivos = r'C:\Users\lumal\fiap\gs2\dados'

# Lista com os nomes dos arquivos
nomes_arquivos = [
    'focos_qmd_inpe_2020-01-01_2020-05-25_24.409249.csv',
    'focos_qmd_inpe_2020-05-26_2021-05-25_16.138616.csv',
    'focos_qmd_inpe_2021-05-26_2022-05-25_48.459106.csv',
    'focos_qmd_inpe_2022-05-26_2023-05-25_16.303481.csv',
    'focos_qmd_inpe_2023-05-26_2024-05-25_08.225194.csv',
    'focos_qmd_inpe_2024-05-26_2025-05-27_53.654434.csv'



]

# Lista para armazenar os DataFrames
lista_df = []

# Ler e adicionar cada CSV à lista
for nome in nomes_arquivos:
    caminho_completo = os.path.join(caminho_dos_arquivos, nome)
    df = pd.read_csv(caminho_completo)
    lista_df.append(df)

# Concatenar todos os DataFrames
df_completo = pd.concat(lista_df, ignore_index=True)

# Salvar em um novo arquivo CSV
saida = os.path.join(caminho_dos_arquivos, 'dados_combinados.csv')
df_completo.to_csv(saida, index=False)

print(f"Arquivo combinado salvo em: {saida}")
```

```
import pandas as pd
import os

# Caminho do arquivo
caminho_arquivo = r'C:\Users\lumal\fiap\gs2\dados\dados_combinados.csv'

# Lê o CSV
df = pd.read_csv(caminho_arquivo)

# Garante que a coluna 'Municipio' não tenha valores nulos
df = df.dropna(subset=['Municipio'])

# Conta quantas vezes cada município aparece
top_10_municipios = df['Municipio'].value_counts().head(10)

print("Top 10 municípios com mais entradas:")
print(top_10_municipios)

# Filtra o DataFrame para incluir apenas esses 10 municípios
df_top_10 = df[df['Municipio'].isin(top_10_municipios.index)]

# Salva o novo DataFrame em um novo arquivo
caminho_saida = os.path.join(os.path.dirname(caminho_arquivo),
'top_10_municipios.csv')
df_top_10.to_csv(caminho_saida, index=False)

print(f"\nDados filtrados salvos em: {caminho_saida}")
```

```
import pandas as pd

# --- 1. Ler os arquivos ---
df_incendio =
pd.read_csv('C:/Users/lumal/fiap/gs2/dados/top_10_municipios.csv')
df_clima =
pd.read_csv('C:/Users/lumal/fiap/gs2/dados/dados_clima_com_umidade_min_
max.csv')

# --- 2. Padronizar datas ---
df_incendio['Data'] =
pd.to_datetime(df_incendio['DataHora'].str[:10].str.replace('/', '-'))
df_clima['data'] = pd.to_datetime(df_clima['data'])
```

```python
# --- 3. Padronizar nomes de cidades e estados para caixa alta ---
df_incendio['Municipio'] = df_incendio['Municipio'].str.upper()
df_clima['cidade'] = df_clima['cidade'].str.upper()

df_clima['estado'] = df_clima['estado'].str.upper()

# --- 4. Substituir nomes completos dos estados por siglas ---
estados_siglas = {
    'TOCANTINS': 'TO',
    'MARANHÃO': 'MA',
    'MATO GROSSO': 'MT',
    'BAHIA': 'BA',
    'MATO GROSSO DO SUL': 'MS'
}
df_incendio['Estado'] = df_incendio['Estado'].replace(estados_siglas)

# --- 5. Fazer o merge dos DataFrames ---
df_merged = pd.merge(
    df_incendio,
    df_clima,
    left_on=['Estado', 'Municipio', 'Data'],
    right_on=['estado', 'cidade', 'data'],
    how='inner'
)

# --- 6. Remover colunas redundantes e salvar ---
df_merged = df_merged.drop(columns=['DataHora', 'estado', 'cidade',
'data'])

# --- 7. Salvar o DataFrame final ---
df_merged.to_csv('C:/Users/lumal/fiap/gs2/dados/banco_dados_unificado.c
sv', index=False)

print("✅ Banco de dados unificado salvo com sucesso!")
```

2. Códigos de análise de dados

```
# Global Solution
```

```
## Análise de dados
```

```
### Exploração de dados
import pandas as pd
import numpy as np

dados = pd.read_csv('df.csv')

dados.head()
```

```
dados.tail()
```

```
dados.shape
```

```
dados.dtypes
```

```
dados.describe()
```

```
dados.isnull().sum()
```

```
### Análise de variáveis
import matplotlib.pyplot as plt
import seaborn as sns
dados.head()
```

```
#### Satélite
dados.Satelite.value_counts()
```

```
sns.countplot(dados.Satelite)
```

```
#### País
dados.Pais.value_counts()
```

```
#### Estado
dados.Estado.value_counts()
```

```
sns.countplot(dados.Estado)
```

```
#### Bioma
dados.Bioma.value_counts()
```

```
#### DiaSemChuva
dados.DiaSemChuva.value_counts().sort_index()
```

```
sns.histplot(dados.DiaSemChuva)
```

```
#### Precipitação
dados.Precipitacao.value_counts().sort_index()
```

```
sns.histplot(dados.Precipitacao)
```

```
#### RiscoFogo
dados.RiscoFogo.value_counts().sort_index()
```

```
sns.histplot(dados.RiscoFogo)
```

```
#### temp_max
dados.temp_max.value_counts().sort_index()
```

```
sns.histplot(dados.temp_max)
```

```
#### temp_min
dados.temp_min.value_counts().sort_index()
```

```
sns.histplot(dados.temp_min)
```

```
#### temp_med
dados.temp_med.value_counts().sort_index()
```

```
#### umidade_min
dados.umidade_min.value_counts().sort_index()
```

```
sns.histplot(dados.umidade_min)
```

```
#### umidade_med
dados.umidade_med.value_counts().sort_index()
```

```
sns.histplot(dados.umidade_med)
```

```
#### umidade_max
dados.umidade_max.value_counts().sort_index()
```

```
sns.histplot(dados.umidade_max)
```

```
#### vento_max
```

```
dados.vento_max.value_counts().sort_index()
```

```
sns.histplot(dados.vento_max)
```

```
### Tratamento de dados
dados.dtypes
```

```
dados.isnull().sum()
```

```
# substituindo valores incoerentes ou ausentes
dados.replace(-999, 0, inplace=True)
```

```
# Converte a coluna 'Data' para formato de data
dados.Data = pd.to_datetime(dados.Data, errors='coerce')
```

```
# Remove duplicatas
dados.drop_duplicates(inplace=True)
```

```
dados.head()
```

```
dados.to_csv('df_limpo.csv', index=False)
```

```
dados.describe()
```

```
### Correlação
dados_numericas = dados.select_dtypes(include=['float64', 'int64'])
# Calcula a correlação apenas com as colunas numéricas
correlacao = dados_numericas.corr()
correlacao['RiscoFogo']
```

```
plt.figure(figsize=(12, 8))
sns.heatmap(correlacao, annot=True, cmap='coolwarm')
plt.title('Mapa de Correlação')
plt.show()
```

```
# Análise temporal
dados.groupby(pd.Grouper(key='Data',
freq='M'))['temp_max'].mean().plot(title='Média da Temperatura Máxima')
```

```
# Análise de enventos críticos
alto_risco = dados[dados['RiscoFogo'] > 0.7]
print("Total de registros com alto risco:", len(alto_risco))
```

3. Códigos de Machine Learning

```
# Instalando bibliotecas
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.impute import SimpleImputer
```

```
# Selecionando features e target
features = ['temp_max', 'temp_min', 'umidade_min', 'umidade_med',
'vento_max', 'Precipitacao']
X = dados[features]
y = dados['RiscoFogo']

# Dividindo em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Cria o imputer para preencher com média
imputer = SimpleImputer(strategy='mean')

# Aplica ao treino e teste
X_train_scaled = imputer.fit_transform(X_train_scaled)
X_test_scaled = imputer.transform(X_test_scaled)
```

```
print("Primeiras linhas dos dados padronizados (treino):")
print(X_train_scaled[:5])
```

```
### Random Forest
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)
```

```
# Avalia o modelo
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(" Modelo Random Forest")
print("MAE:", mae)
print("R²:", r2)

# previsões e reais para comparação
df_resultados = pd.DataFrame({
    'RiscoFogo_real': y_test.values,
    'RiscoFogo_predito': y_pred
})
print(df_resultados.head())
```

```
### Gradient Boosting
modelo_gb = GradientBoostingRegressor(n_estimators=100,
learning_rate=0.1, random_state=42)

# Treina o modelo
modelo_gb.fit(X_train_scaled, y_train)

#  Faz previsões com os dados de teste
y_pred_gb = modelo_gb.predict(X_test_scaled)

#  Avalia o desempenho
mae_gb = mean_absolute_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

print("Modelo Gradient Boosting")
print(f"MAE: {mae_gb:.4f}")
print(f"R²: {r2_gb:.4f}")

# previsões e reais para comparação
df_resultados_gb = pd.DataFrame({
    'RiscoFogo_real': y_test.values,
    'RiscoFogo_predito': y_pred_gb
})
print(df_resultados_gb.head())
```

```
### MLPRegressor
# Cria o modelo MLPRegressor (Rede Neural)
```

```
# hidden_layer_sizes=(100,) → 1 camada escondida com 100 neurônios
# max_iter=500 → número máximo de iterações (pode ajustar)
modelo_nn = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500,
random_state=42)

# 🚀 Treina o modelo com os dados padronizados
modelo_nn.fit(X_train_scaled, y_train)

# 🔎 Faz previsões com os dados de teste
y_pred_nn = modelo_nn.predict(X_test_scaled)

# 📊 Avalia o desempenho
mae_nn = mean_absolute_error(y_test, y_pred_nn)
r2_nn = r2_score(y_test, y_pred_nn)

print("Modelo de Rede Neural (MLPRegressor)")
print(f"MAE: {mae_nn:.4f}")
print(f"R²: {r2_nn:.4f}")

# previsões e reais para comparação
df_resultados_nn = pd.DataFrame({
    'RiscoFogo_real': y_test.values,
    'RiscoFogo_predito': y_pred_nn
})
print(df_resultados_nn.head())
```

4. Códigos de sensores

4.1 Sensor 1

```
#include <LiquidCrystal_I2C.h>

#define LED 2
#define BUTTON 4

LiquidCrystal_I2C lcd(0x27, 16, 2);

struct Clima {
```

```
  const char* cidade;
  float temperatura;
  int umidade;
  float vento;
};

Clima dados[] = {
  {"Lagoa da Confusão", 36.5, 25, 4.2},
  {"Mirador", 38.1, 22, 5.0},
  {"Pium", 34.8, 31, 3.9},
  {"Formoso do Araguaia", 39.0, 19, 6.5},
  {"Balsas", 37.2, 28, 5.3},
  {"Cocalinho", 40.3, 18, 7.1},
  {"Formosa do Rio Preto", 36.9, 29, 4.8},
  {"Alto Parnaíba", 35.5, 32, 3.7},
  {"Porto Murtinho", 33.1, 34, 2.9},
  {"São Desidério", 38.7, 23, 5.9}
};

int indexAtual = 0;

void setup() {
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();
  pinMode(LED, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);

  lcd.setCursor(0, 0);
  lcd.print("Monitor Incendio");
  delay(2000);
  lcd.clear();
}

void loop() {
  // Botão pressionado por 3 segundos ativa o alerta Alerta manual!
  if (digitalRead(BUTTON) == LOW) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("🔥 ALERTA MANUAL!");
    digitalWrite(LED, HIGH);
    delay(3000);
    digitalWrite(LED, LOW);
```

```
    // Aguarda o botão ser solto
    while (digitalRead(BUTTON) == LOW) {
      delay(10);
    }

    lcd.clear();
    return;  // evita exibir cidade logo após o alerta
  }

  // Exibição normal das cidades
  Clima atual = dados[indexAtual];

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(atual.cidade);
  lcd.setCursor(0, 1);
  lcd.print("T:"); lcd.print(atual.temperatura);
  lcd.print(" U:"); lcd.print(atual.umidade);

  bool risco = (atual.temperatura > 35.0 && atual.umidade < 30);
  digitalWrite(LED, risco ? HIGH : LOW);

  Serial.println("------");
  Serial.print("Cidade: "); Serial.println(atual.cidade);
  Serial.print("Temp: "); Serial.println(atual.temperatura);
  Serial.print("Umidade: "); Serial.println(atual.umidade);
  Serial.print("Vento: "); Serial.println(atual.vento);
  Serial.println(risco ? "🔥 RISCO ALTO" : "Risco baixo");

  delay(4000);  // tempo de exibição de cada cidade
  indexAtual = (indexAtual + 1) % 10;
}
```

4.2 Sensor 2

```
#include <DHT.h>
#include <LiquidCrystal_I2C.h>

#define DHTPIN 15
#define DHTTYPE DHT22
```

```
#define MQ2_PIN 34
#define LDR_PIN 35
#define BUTTON 4
#define LED_VERMELHO 2
#define LED_VERDE 5

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();

  pinMode(MQ2_PIN, INPUT);
  pinMode(LDR_PIN, INPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  pinMode(LED_VERMELHO, OUTPUT);
  pinMode(LED_VERDE, OUTPUT);

  dht.begin();
  lcd.setCursor(0, 0);
  lcd.print("Iniciando sistema");
  delay(2000);
  lcd.clear();
}

void loop() {
  float temperatura = dht.readTemperature();
  float umidade = dht.readHumidity();
  int fumaca = analogRead(MQ2_PIN);
  int luz = analogRead(LDR_PIN);
  bool alertaManual = digitalRead(BUTTON) == LOW;

  // Condição de risco
  bool riscoAlto = (temperatura > 35.0 && umidade < 30.0) || fumaca >
2000 || luz < 1000;

  // Exibe no LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:"); lcd.print(temperatura, 1);
  lcd.print(" U:"); lcd.print(umidade, 0);
```

```
  lcd.setCursor(0, 1);
  lcd.print("Fum:"); lcd.print(fumaca);
  lcd.print(" Luz:"); lcd.print(luz / 100);

  // Envia para o Serial Monitor
  Serial.println("---- LEITURA ----");
  Serial.print("Temperatura: "); Serial.print(temperatura);
Serial.println(" C");
  Serial.print("Umidade: "); Serial.print(umidade); Serial.println("
%");
  Serial.print("Fumaca: "); Serial.println(fumaca);
  Serial.print("Luz: "); Serial.println(luz);
  Serial.println(riscoAlto ? ">> RISCO DETECTADO <<" : "Ambiente
normal");
  Serial.println();

  // Alerta manual
  if (alertaManual) {
    digitalWrite(LED_VERMELHO, HIGH);
    digitalWrite(LED_VERDE, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" ALERTA MANUAL!");
    Serial.println("⚠️ ALERTA MANUAL ATIVADO!");
    delay(3000);
  }
  else if (riscoAlto) {
    digitalWrite(LED_VERMELHO, HIGH);
    digitalWrite(LED_VERDE, LOW);
  } else {
    digitalWrite(LED_VERMELHO, LOW);
    digitalWrite(LED_VERDE, HIGH);
  }

  delay(3000);
}
```