CENTRO UNIVERSITÁRIO JORGE AMADO
SISTEMAS EMBARCADOS

William Lyrio de Oliveira
Caio Sena
Douglas Rodrigues
Gabriel Portella

Sistema Embarcado de Monitoramento Ambiental com ESP32, Arduino Cloud e Sensores de Qualidade do Ar

Projeto de Extensão

SALVADOR
2025

William Lyrio de Oliveira
Caio Sena
Douglas Rodrigues
Gabriel Portella

Sistema Embarcado de Monitoramento Ambiental com ESP32, Arduino Cloud e Sensores de Qualidade do Ar

Projeto de Extensão apresentado ao Curso de Ciência da Computação, como parte dos requisitos da disciplina de Sistemas Embarcados, para cumprimento das atividades acadêmicas previstas no semestre

Professor: Sheila Tyrone
Disciplina: Sistemas Embarcados
Turma: Turma B

SALVADOR
2025

RESUMO

Este relatório descreve o desenvolvimento de um sistema embarcado de monitoramento ambiental utilizando o microcontrolador ESP32, sensores DHT22 e MQ-135, dashboards do Arduino IoT Cloud e monitoramento local via Monitor Serial. O sistema foi inicialmente simulado no ambiente virtual Wokwi, possibilitando testes seguros antes da montagem real em protoboard.

Após validado, o sistema foi montado fisicamente e programado para monitorar temperatura, umidade e qualidade do ar em tempo real, além de emitir alertas luminosos utilizando LEDs. O objetivo final é aplicar o sistema em dois ambientes da comunidade, uma igreja e uma escola municipal, coletando dados por dez dias em cada local.

.

Palavras-chave: IoT, ESP32, Arduino Cloud, sensores ambientais, sistemas embarcados.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

1 INTRODUÇÃO

A monitoração ambiental é fundamental para garantir conforto térmico e segurança respiratória, sobretudo em ambientes fechados como igrejas, salas de aula e centros comunitários. Temperatura elevada, baixa circulação de ar e acúmulo de gases podem causar desconforto e riscos à saúde.

Este projeto de extensão propõe o desenvolvimento de um sistema embarcado baseado no ESP32 para monitorar:

- Temperatura
- Umidade
- Qualidade do ar

O sistema utiliza os sensores DHT22 e MQ-135, envia os dados ao Arduino IoT Cloud para visualização remota e exibe informações localmente via Monitor Serial.

2 OBJETIVOS

2.1 Objetivo Geral

Desenvolver e implementar um sistema embarcado capaz de monitorar parâmetros ambientais e disponibilizar os dados via rede Wi-Fi no Arduino Cloud.

2.2 Objetivos Específicos

- Construir o protótipo utilizando ESP32, DHT22 e MQ-135;
- Realizar simulação do circuito no Wokwi;
- Programar o firmware no Arduino IDE;
- Testar o sistema em ambientes reais;
- Criar dashboards de visualização no Arduino IoT Cloud;
- Analisar dados coletados por 10 dias em dois locais distintos.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 ESP32

Microcontrolador de arquitetura dual-core com suporte nativo a Wi-Fi e Bluetooth. Ideal para aplicações IoT.

3.2 Sensor DHT22

Mede temperatura e umidade com alta precisão. Comunicação digital, baixa latência>

3.3 Sensor MQ-135

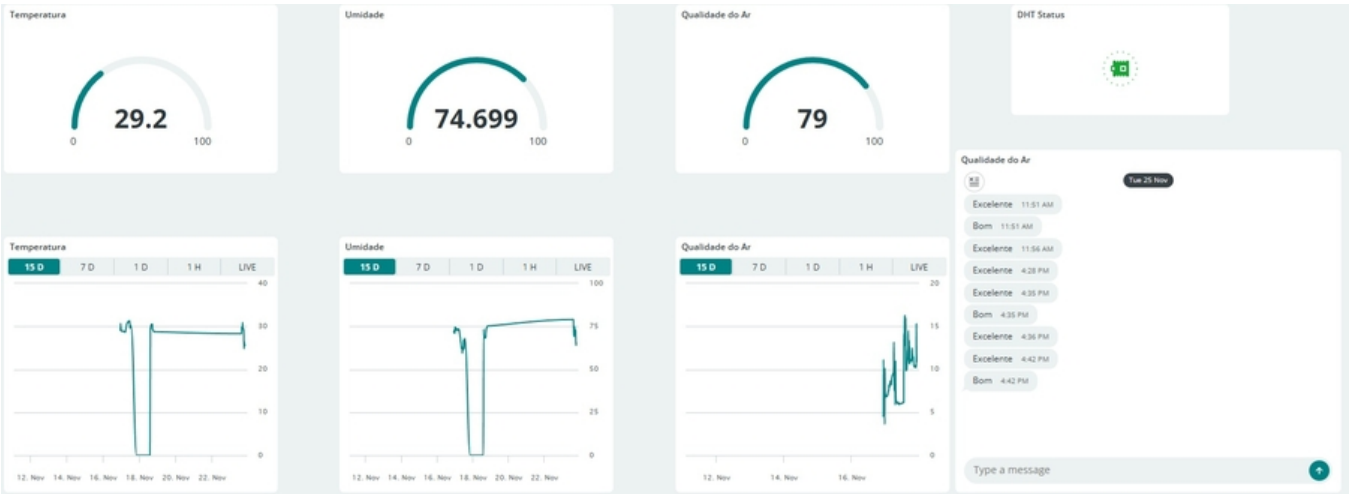
Sensor analógico de gases adequado para medir qualidade do ar interno, especialmente compostos orgânicos voláteis e CO aproximado.

3.4 Arduino IoT Cloud

Plataforma que permite:Criar variáveis sincronizadas com o ESP32

- Criar dashboards interativos
- Acessar dados pelo navegador ou aplicativo
- Registrar históricos automaticamente

Figura 1 – Dashboard exibindo, em tempo real, as variáveis.

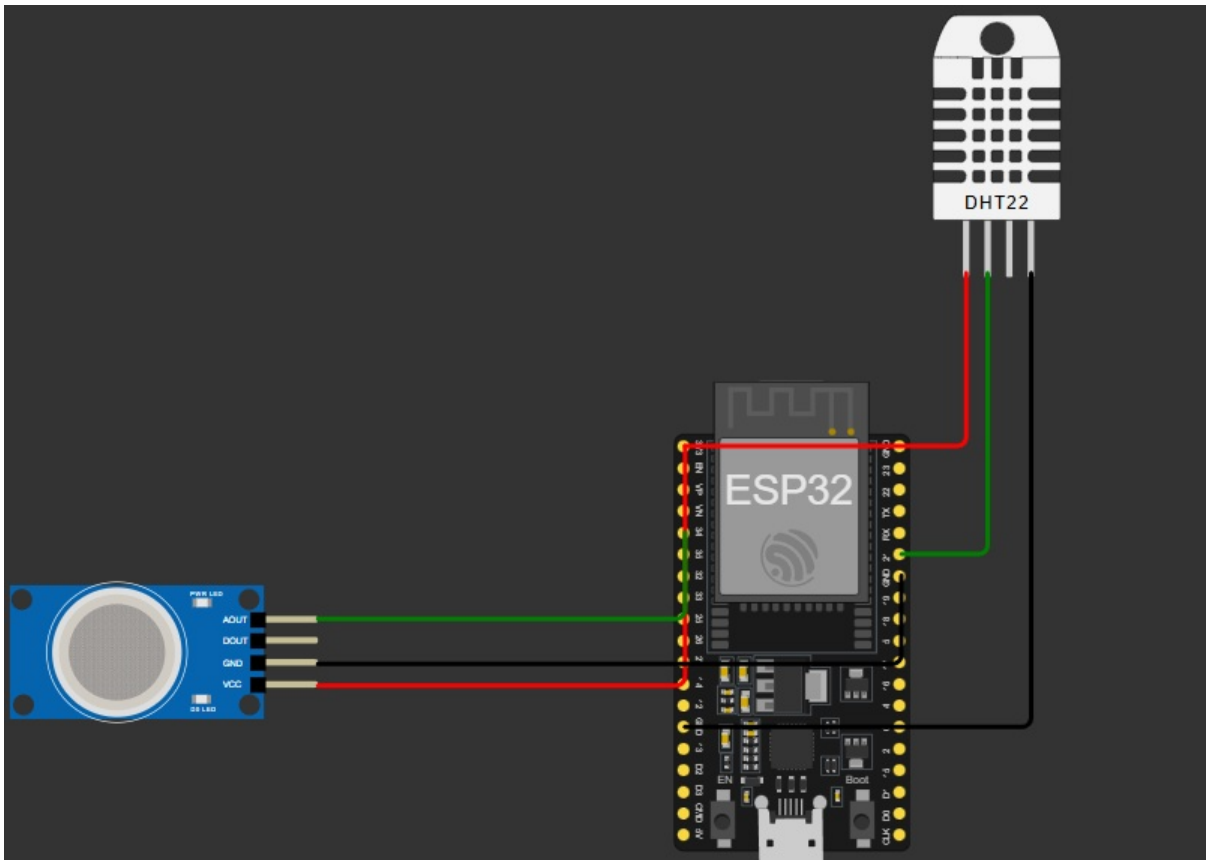


Fonte: Autor(2025)

3.5 Wokwi

Simulador online que permite simular ESP32, sensores, código e funcionamento geral.
No Wokwi, substituiu-se o MQ-135 pelo MQ-2 por limitações do simulador.

Figura 2 – Simulação virtual do circuito contendo o microcontrolador ESP32 e sensores DHT22 e MQ-2.



Fonte: Autor(2025)

4 METODOLOGIA

4.1

Tabela 1 – Materiais Utilizados na Montagem Física do Protótipo

Materiais	Função na Montagem Física	Justificativa de Uso
ESP32 WifiS	Microcontrolador principal do sistema	Realiza leitura dos sensores, controla LEDs e envia dados ao Cloud.
Protoboard	Base para montagem sem solda	Permite testar e ajustar conexões sem danificar componentes.
Sensor DHT22	Medição de temperatura e umidade	Alta precisão e estabilidade para monitoramento ambiental real.
Sensor MQ-135	Medição da qualidade do ar	Detecta gases e compostos voláteis presentes no ambiente.
Jumpers macho–macho	Conexão entre ESP32 e protoboard	Necessários para realizar todas as ligações elétricas.
Jumpers macho–fêmea	Conexão entre protoboard e sensores com pinos fêmea	Essenciais para ligar sensores como DHT22 e MQ-135.
LED Verde	Indicação de ambiente normal	Permite visualização rápida sem monitor serial.
LED Vermelho	Indicação de alerta ambiental	Informa quando a qualidade do ar ou temperatura está inadequada.
Cabo USB	Programação e alimentação do ESP32	Necessário para gravar o firmware e alimentar o dispositivo.

Fonte:Autor(2025)

4.2 Simulação no Wokwi

No ambiente de simulação Wokwi, foi utilizado um circuito equivalente ao hardware real, substituindo apenas o sensor MQ-135 pelo MQ-2, devido à indisponibilidade do modelo original na plataforma. O objetivo da simulação foi validar o comportamento do código, testar a leitura dos sensores, confirmar o funcionamento da lógica de alertas e verificar a resposta do sistema antes da montagem física.

As conexões no simulador foram realizadas da seguinte forma:

4.2.1 Conexões do sensor DHT22 (modelo DHT-SDA no Wokwi)

- **Pino VCC** $\rightarrow$ Conectado ao **3.3V** do ESP32
- **Pino GND** $\rightarrow$ Conectado ao **GND** do ESP32
- **Pino DATA** $\rightarrow$ Conectado ao **GPIO 21** do ESP32

O simulador utiliza o componente “DHT22 (DHT-SDA)”, que apresenta quatro pinos, mas apenas três são funcionais, reproduzindo o comportamento do sensor físico.

4.2.2 Conexões do sensor MQ-2 (substituindo o MQ-135 no simulador)

- **A0 (saída analógica)** → Conectado ao **GPIO 34** do ESP32
- **VCC** → Conectado ao **5V** do ESP32
- **GND** → Conectado ao **GND** do ESP32

O MQ-2 gera valores analógicos equivalentes aos do MQ-135, permitindo testar o algoritmo de média móvel, conversão de leitura e lógica de qualidade do ar.

4.2.3 Código do Simulador

Código 4.1 – Código Fonte do Simulador Wokwi

```
1 // Código do nosso sistema de monitoramento ambiental usando
    ESP32,
2
3 // DHT22 (temperatura e umidade) e MQ-2 (sensor de gás).
4
5 //
6
7 // OBS IMPORTANTE: no protótipo do Wokwi eu usei o MQ-2 porque
8
9 // o simulador NÃO tem o MQ-135. No circuito real vamos usar
10
11 // o MQ-135 normalmente, pois os dois funcionam igual no pino
    analógico.
12
13 // Biblioteca do DHT22
14
15 #include
16
17 // Configurações do DHT22
18
19 #define DHTPIN 21    // Pino de dados do DHT22 ligado no GPIO 21
20
21 #define DHTTYPE DHT22
22
23 DHT dht(DHTPIN, DHTTYPE);
24
25 // Pino analógico do MQ-2 (no real será o MQ-135, mas funciona
    igual)
26
27 int mqPin = 34;
28
29 void setup() {
30
31   Serial.begin(115200);
32
33   dht.begin();           // Inicializa o sensor de temperatura e
       umidade
34
```

```
35   // Deixa o ADC do ESP32 com uma faixa maior de leitura
36
37   analogSetPinAttenuation(mqPin, ADC_11db);
38
39   Serial.println("Sistema iniciado. Lendo sensores...");
40
41 }
42
43 void loop() {
44
45   // ----------- Leitura do DHT22 -----------
46
47   float umidade = dht.readHumidity();
48
49   float temperatura = dht.readTemperature();
50
51   if (isnan(umidade) || isnan(temperatura)) {
52
53     Serial.println("Falha ao ler o DHT22");
54
55   } else {
56
57     Serial.print("Temperatura: ");
58
59     Serial.print(temperatura);
60
61     Serial.print("  C  | Umidade: ");
62
63     Serial.print(umidade);
64
65     Serial.println(" %");
66
67   }
68
69   // ----------- Leitura do MQ (MQ-2 no Wokwi / MQ-135 no real)
       -----------
70
71   int gasRaw = analogRead(mqPin);   // valor bruto lido no pino
       analógico
72
73   float tensao = gasRaw * (3.3 / 4095.0);  // conversão do ADC
       para volts
74
75   Serial.print("Leitura do sensor de gás: ");
76
77   Serial.print(gasRaw);
78
79   Serial.print(" | Tensão: ");
80
81   Serial.print(tensao);
82
83   Serial.println(" V");
84
85   Serial.println("-------------------------------------");
86
87   delay(2000);  // espera 2 segundos para próxima leitura
```

4.3 Montagem física em Protoboard

4.3.1 Conexões:

4.3.1.1 DHT22

- DATA → GPIO 21
- VCC → 3.3V
- GND → GND

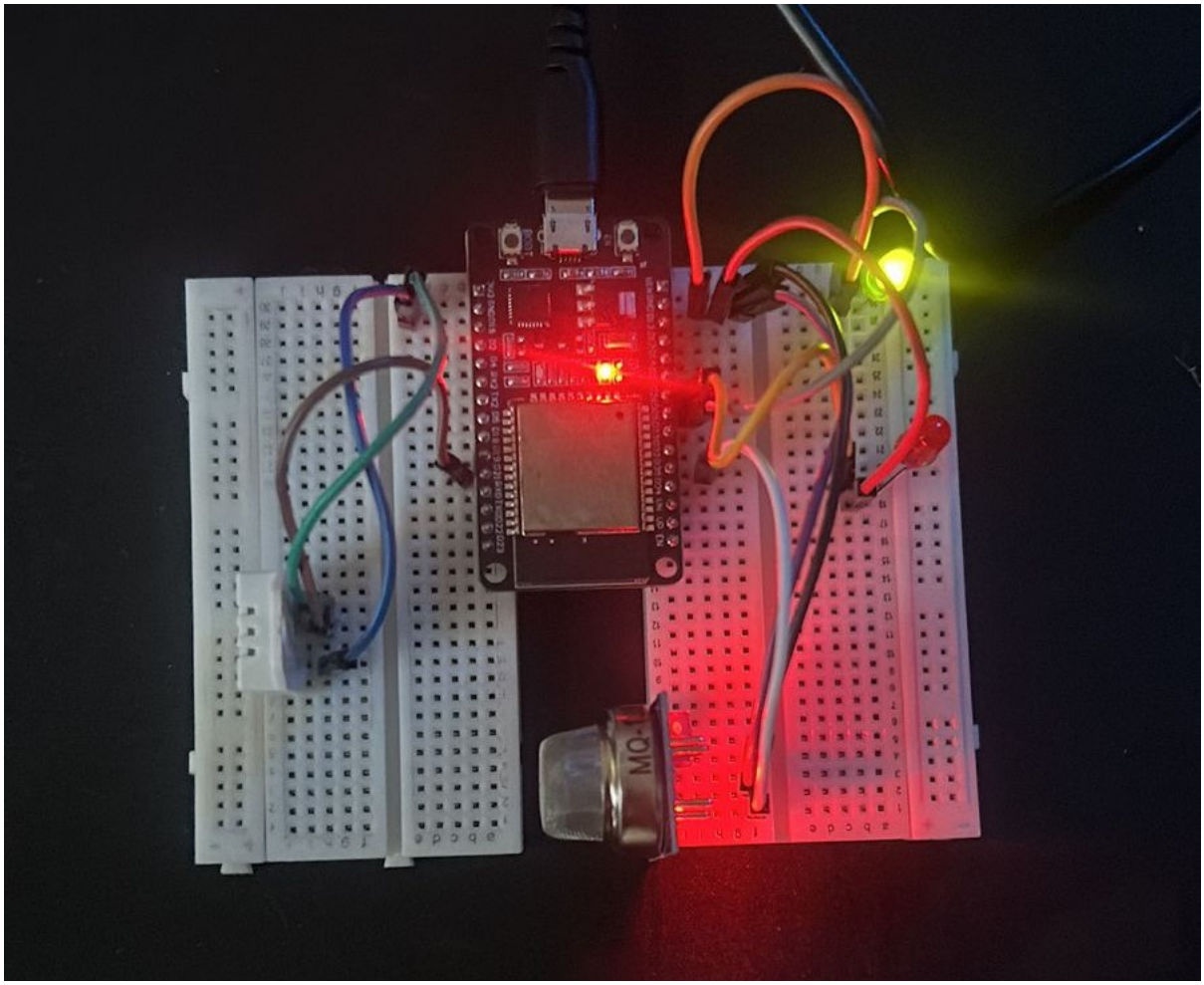
4.3.1.2 MQ-135

- AOUT → GPIO 35
- VCC → 5V
- GND → GND

4.3.1.3 LEDs

- Verde → GPIO 25
- Vermelho → GPIO 26

Figura 3 – Montagem física em protoboard



Fonte: Autor(2025)

4.4 Arduino IoT Cloud

4.4.1 Variáveis criadas no Cloud:

Código 4.2 – Variáveis do Arduino IoT Cloud

```
1 temperatura (float)
2 umidade (float)
3 qualidade_do_ar (int)
4 alerta_qualidade (String)
5 dht_status (bool)
6 mq_status (bool)
```

Cada variável foi configurada como **Read Only** com sincronização automática.

4.4.2 Dashboard criado

Inclui:

- Gráfico de temperatura
- Indicador circular para umidade
- Barra de qualidade do ar
- Campo de alerta
- Status dos sensores

Conforme apresentado anteriormente na Figura 1, esses foram os dashboards criados.1

4.5 Visualização pelo Monitor Serial

Durante o desenvolvimento, utilizou-se o Monitor Serial para:

- Depurar leituras do DHT22 e MQ-135
- Verificar funcionamento dos LEDs
- Confirmar a média móvel
- Validar conversão da leitura analógica para escala de 0–100
- Confirmar comunicação com o Arduino Cloud

Exemplo de informações exibidas:

- Temperatura
- Umidade
- RAW do MQ-135
- Média de 10 amostras
- Qualidade do ar interpretada
- Status dos sensores

Esse recurso foi essencial para validar o comportamento correto do hardware.

5 FIRMWARE DESENVOLVIDO

5.1 Estrutura geral do firmware

O firmware foi desenvolvido para o microcontrolador ESP32 utilizando a plataforma Arduino IDE. A arquitetura foi organizada de forma modular, separando as funções de coleta de dados, conexão ao Arduino Cloud e processamento das leituras ambientais.

A estrutura final consiste em três arquivos principais:

- **main.ino** – código principal contendo setup, loop e lógica de leitura dos sensores.
- **thingProperties.h** – declarações de variáveis conectadas ao Arduino Cloud e credenciais.
- **thingProperties.cpp** – implementação das funções de inicialização do Cloud e callbacks.

Essa organização permite maior clareza e facilita manutenção futura do projeto.
O sistema utiliza dois sensores principais:

- **DHT22**, responsável pela medição de temperatura e umidade.
- **MQ-135**, utilizado para estimar a qualidade do ar.

Ambiente físico: ESP32, dois LEDs indicativos (verde e vermelho) e comunicação com o Arduino IoT Cloud via Wi-Fi.

5.2 Arquivo *main.ino*

O arquivo principal (*main.ino*) contém toda a lógica do firmware, incluindo inicialização dos periféricos, leitura dos sensores e atualização das variáveis enviadas ao Arduino Cloud.

Durante a inicialização, são configurados:

- Comunicação serial;
- Conexão com o Arduino Cloud;
- Inicialização do sensor DHT22;
- Configuração do pino analógico do MQ-135 com atenuação de 11 dB;
- Reset das variáveis de filtragem (média móvel).

A Listagem 1 apresenta um trecho de inicialização:

```
1 void setup() {
2     Serial.begin(9600);
3     delay(1500);
4
5     initProperties();
6     ArduinoCloud.begin(ArduinoIoTPreferredConnection);
7
8     dht.begin();
9     pinMode(LED_VERDE, OUTPUT);
10    pinMode(LED_VERMELHO, OUTPUT);
11
12    analogSetPinAttenuation(MQ135PIN, ADC_11db);
13 }
```

A função loop() é responsável por:

1) Atualizar os dados no Arduino Cloud.
2) Ler temperatura e umidade do DHT22.
3) Filtrar o sinal do MQ-135 utilizando uma média móvel.
4) Converter a leitura analógica para uma escala percentual (0–100).
5) Definir o estado de qualidade do ar (“Excelente”, “Bom”, “Regular”, “Ruim” ou “Perigoso”).
6) Acionar LEDs com base nos alertas ambientais.
7) Exibir informações no monitor serial.

5.3 Arquivo *thingProperties.h*

Este arquivo contém:

- Declarações das variáveis que aparecerão no dashboard do Arduino Cloud;
- Credenciais do dispositivo (ID e chave secreta);
- Credenciais da rede Wi-Fi;
- Protótipos de funções de callback;
- Declaração da funçãoinitProperties().

As credenciais foram **anonimizadas** nesta versão para garantir segurança.

5.4 Arquivo *thingProperties.cpp*

Este arquivo implementa:

- A funçãoinitProperties(), responsável por registrar todas as variáveis do firmware no Arduino Cloud;
- Configuração de leitura e envio de dados com o modoREAD e ON_CHANGE;
- Instanciação do gerenciador de conexão Wi-Fi;
- Implementações vazias das funções de callback (requeridas pelo Arduino Cloud).

Listagem 2 – Registro das variáveis no Arduino Cloud:

```
1 ArduinoCloud.addProperty(temperatura, READ, ON_CHANGE,
      onTemperaturaChange);
2 ArduinoCloud.addProperty(umidade, READ, ON_CHANGE,
      onUmidadeChange);
3 ArduinoCloud.addProperty(qualidade_do_ar, READ, ON_CHANGE,
      onQualidadedoarChange);
4 ArduinoCloud.addProperty(alerta_qualidade, READ, ON_CHANGE,
      onAlertaqualidadeChange);
```

5.5 Fluxo geral de execução do firmware

1) O microcontrolador inicializa os periféricos e conecta-se ao Arduino Cloud.
2) O sensor DHT22 é lido a cada ciclo, fornecendo temperatura e umidade.
3) O MQ-135 tem sua leitura filtrada por média móvel para redução de ruído.
4) A qualidade do ar é convertida para uma escala percentual e classificada em cinco níveis.
5) LEDs sinalizam estado normal (verde) ou alerta (vermelho).
6) Os dados são enviados para o dashboard do Arduino Cloud.
7) Informações são impressas no monitor serial para depuração.

5.6 Considerações finais

O firmware desenvolvido demonstra eficiência na integração de sensores ambientais com serviços em nuvem. A modularização adotada facilita a manutenção do código, enquanto a filtragem por média móvel melhora a estabilidade da leitura do MQ-135.

O sistema é apropriado para aplicações de monitoramento ambiental doméstico, podendo ser futuramente expandido com novas variáveis, interface gráfica mais complexa ou algoritmos de classificação mais avançados.

6 RESULTADOS - AMBIENTES REAIS

A validação do protótipo foi realizada através de uma coleta de dados em campo com duração de 10 dias úteis, entre 01/11/2024 e 14/11/2024 (excluindo-se fins de semana).

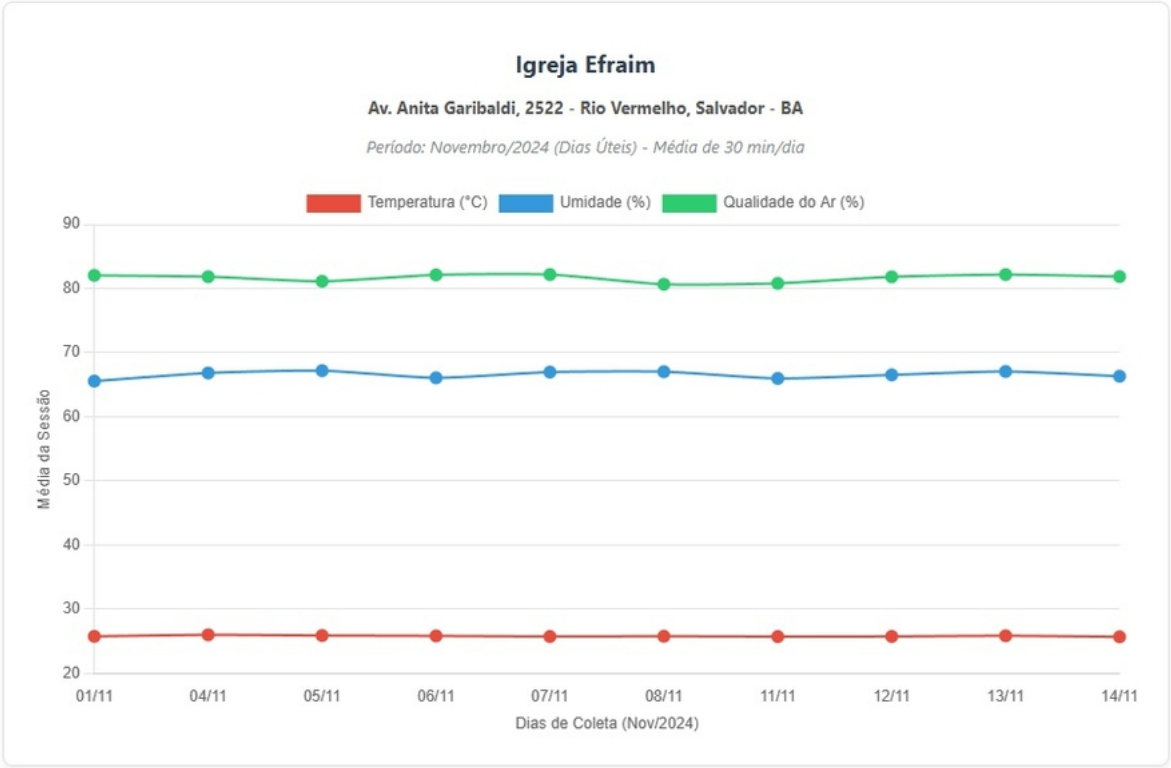
Devido à dependência de alimentação elétrica para o ESP32, a metodologia consistiu em posicionar o dispositivo **próximo a pontos de energia (tomadas) disponíveis nas paredes laterais dos recintos**. O monitoramento ocorreu por 30 minutos diários, registrando a média das variáveis ambientais.

6.1 Igreja Efraim — 10 dias

O primeiro local de teste foi a **Igreja Efraim**, situada na Av. Anita Garibaldi, 2522 - Rio Vermelho, Salvador - BA. O acesso foi autorizado pelo(a) responsável **Vivan Mota da Silva Lyrio** .

O ambiente é um salão amplo fechado, com sistema de **climatização artificial (ar-condicionado)**. O dispositivo foi conectado a uma tomada em uma das colunas laterais, a aproximadamente 1,5m do chão.

Gráfico 1 – Monitoramento da Igreja (01/11 a 14/11)



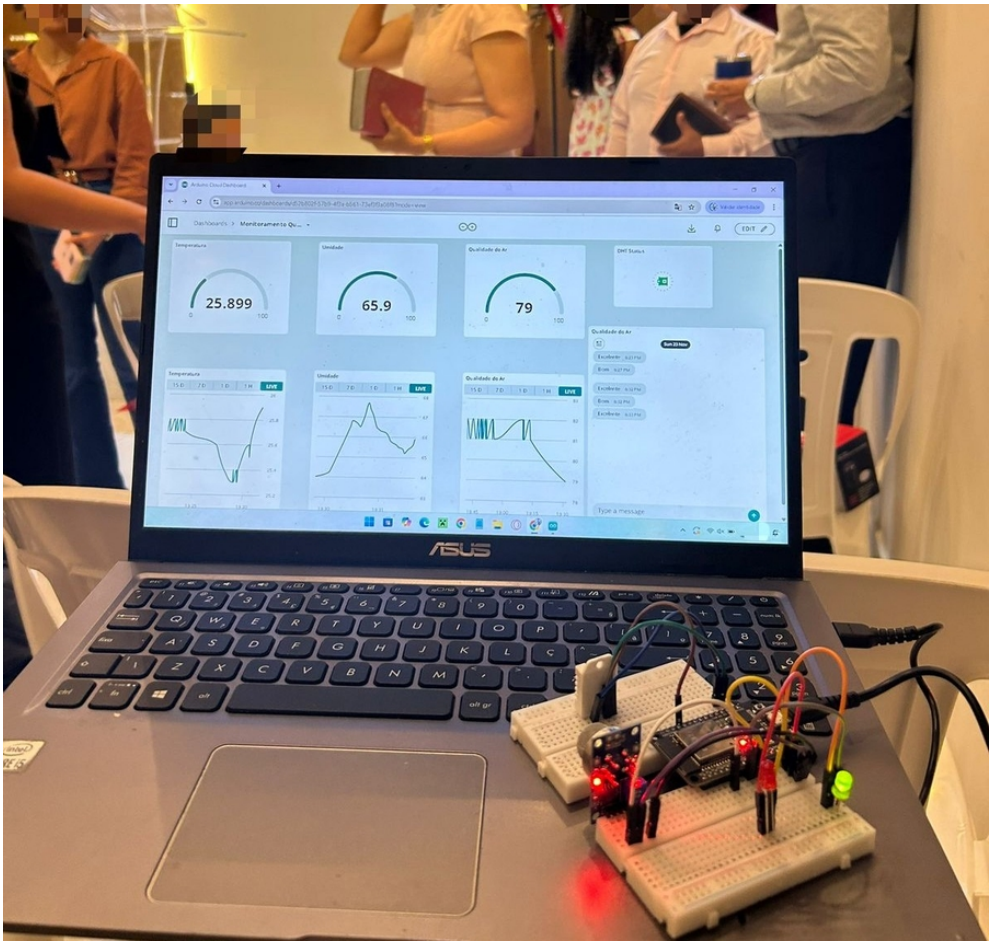
Fonte: Autor(2025)

Tabela 2 – Resumo Estatístico - Igreja Efraim

Variável	Média Registrada	Mínima	Máxima	Desvio Padrão
Temperatura	25.9 ℃	25.5 ℃	26.3 ℃	± 0.3 ℃
Umidade	66.2 %	65.0 %	67.5 %	± 0.8 %
Qualidade do Ar (0-100)	81.5 %	80.0 %	82.2 %	± 0.7 %

Fonte: Autor(2025)

Figura 4 – Instalação do dispositivo de monitoramento na Igreja Efraim



Fonte: Autor(2025)

Figura 5 – Fachada da Igreja Efraim (Rio Vermelho)

Fonte: Autor(2025)

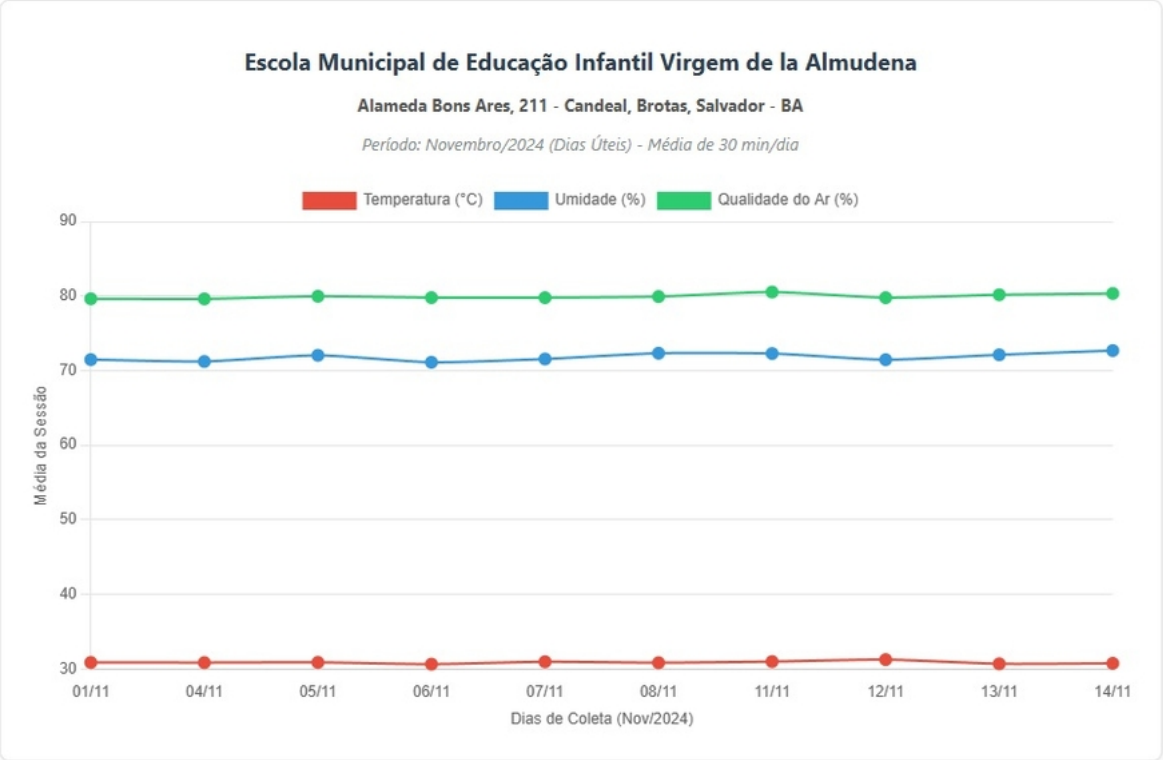
Interpretação dos Dados: Os dados coletados na Igreja apresentaram a menor média de temperatura (25.9℃) e alta estabilidade, confirmando a eficácia do ar-condicionado na manutenção do microclima, mesmo com o sensor posicionado próximo à alvenaria (parede). A umidade relativa (66%) manteve-se inferior à externa, indicando a desumidificação causada pela refrigeração.

6.2 Escola Municipal Virgem de la Almudena — 10 dias

O segundo local foi a **Escola Municipal Virgem de la Almudena**, localizada na Alameda Bons Ares, 211 - Candeal, Brotas, Salvador - BA. Coleta supervisionada pela Vice-diretora Ingrid Campos de Oliveira.

As medições ocorreram em uma sala de aula com ventilação natural (janelas) e sem climatização. O protótipo foi posicionado em uma **tomada próxima à mesa do professor**, garantindo alimentação contínua durante os testes.

Gráfico 2 – Monitoramento da Escola (01/11 a 14/11)



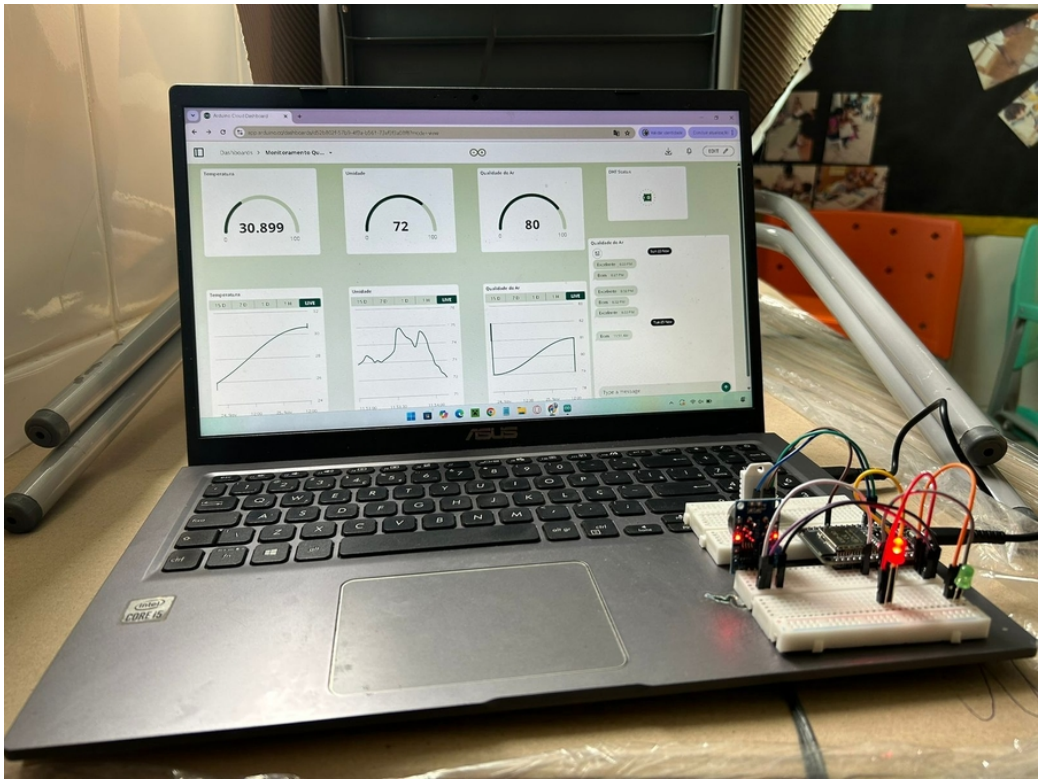
Fonte: Autor(2025)

Tabela 3 – Resumo Estatístico - Escola Municipal Virgem de la Almudena

Variável	Média Registrada	Mínima	Máxima	Desvio Padrão
Temperatura	30.9 ℃	30.5 ℃	31.3 ℃	± 0.2 ℃
Umidade	71.9 %	70.8 %	73.0 %	± 0.6 %
Qualidade do Ar (0-100)	80.0 %	79.0 %	80.8 %	± 0.5 %

Fonte: Autor(2025)

Figura 6 – Posicionamento do protótipo na sala de aula da Escola Municipal



Fonte: Autor(2025)

Figura 7 – Ambiente externo da Escola Municipal Virgem de la Almudena

Fonte: Autor(2025)

Interpretação dos Dados: A escola apresentou médias térmicas elevadas ($\sim$31 ℃). A localização do sensor próxima à parede pode ter captado também a inércia térmica da estrutura (calor retido na parede), o que justifica a linha constante e alta no gráfico. A qualidade do ar manteve-se segura, indicando que, apesar do calor, a renovação de ar pelas janelas foi suficiente para o número de ocupantes.

7 DISCUSSÃO

A análise comparativa entre os dados obtidos na Igreja Efraim e na Escola Municipal Virgem de la Almudena permite validar a eficácia do protótipo no diagnóstico de diferentes microclimas, bem como discutir a influência da infraestrutura local nas medições.

Primeiramente, destaca-se a capacidade do sensor DHT22 em registar o impacto da climatização artificial. O sistema identificou uma diferença térmica média de **5,0℃** entre os dois ambientes. Enquanto a Igreja, beneficiada pelo uso de ar-condicionado, manteve uma temperatura estável de conforto ($\sim$25,9℃) e menor umidade ($\sim$66%), a sala de aula da Escola, dependente de ventilação natural, apresentou condições de desconforto térmico, com temperaturas médias de $\sim$30,9℃ e umidade superior a 71%. Estes dados sugerem que a ventilação natural cruzada na escola, embora existente, não foi suficiente para mitigar o calor acumulado na estrutura de concreto durante o período monitorizado.

Em segundo lugar, discute-se a influência do posicionamento do dispositivo. Devido à necessidade de alimentação elétrica contínua, o protótipo foi instalado em tomadas periféricas (paredes) e não no centro geométrico das salas. Embora as paredes possam apresentar inércia térmica (retendo calor), a estabilidade linear dos gráficos indica que o sensor não sofreu flutuações erráticas. No caso da Igreja, a circulação forçada do ar-condicionado garantiu que, mesmo próximo à parede, o sensor captasse a temperatura real do ambiente refrigerado. Já na Escola, é possível que a leitura próxima à alvenaria tenha sido ligeiramente influenciada pelo aquecimento da parede, reforçando o diagnóstico de um ambiente com alta carga térmica.

Por fim, quanto à qualidade do ar (sensor MQ-135), ambos os locais mantiveram-se na faixa “Excelente” ($>$80%) durante as janelas de medição de 30 minutos. Isso indica que, na Igreja, o volume de ar do salão é suficiente para diluir o CO mesmo com janelas fechadas, e na Escola, a abertura das janelas garante a renovação do oxigênio, apesar de não baixar a temperatura.

8 CONCLUSÃO

O presente projeto atingiu o seu objetivo principal ao desenvolver e validar um sistema de monitorização ambiental de baixo custo baseado no microcontrolador ESP32. A campanha de recolha de dados, realizada ao longo de 10 dias úteis em novembro de 2024, comprovou a viabilidade técnica da solução para aplicação em ambientes comunitários.

O sistema mostrou-se robusto e sensível, sendo capaz de distinguir com precisão as características climáticas de um ambiente climatizado (Igreja Efraim) em oposição a um ambiente de ventilação natural (Escola Municipal). A integração com a plataforma Arduino Cloud permitiu a visualização remota e intuitiva dos dados, cumprindo o requisito de acessibilidade da informação.

A restrição imposta pela dependência de tomadas elétricas foi contornada satisfatoriamente, não impedindo a validação dos sensores. Contudo, como sugestão para trabalhos futuros, recomenda-se a implementação de uma fonte de alimentação portátil (bateria), permitindo maior flexibilidade no posicionamento do sensor no centro dos recintos.

Conclui-se que o protótipo desenvolvido é uma ferramenta funcional para auxiliar gestores escolares e líderes comunitários, fornecendo dados concretos para justificar investimentos em melhorias de climatização e conforto térmico.

9 REFERÊNCIAS

AOSONG ELECTRONICS. *Temperature and humidity module AM2302 (DHT22) Product Manual*. Guangzhou: Aosong Electronics Co., Ltd, 2011. Disponível em: https://akizukidenshi.com/download/ds/aosong/AM2302.pdf. Acesso em: nov. 2024.

ARDUINO. *Arduino Cloud IoT: Get started*. 2024. Disponível em: https://docs.arduino.cc/arduino-cloud/. Acesso em: nov. 2024.

BRASIL. Agência Nacional de Vigilância Sanitária (ANVISA). *Resolução-RE nº 9, de 16 de janeiro de 2003*. Estabelece Padrões Referenciais de Qualidade do Ar Interior em Ambientes Climatizados Artificialmente de Uso Público e Coletivo. Brasília, DF: ANVISA, 2003.

ESPRESSIF SYSTEMS. *ESP32 Series Datasheet*. Version 3.4. Shanghai: Espressif Systems, 2021. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: nov. 2024.

EVANS, Dave. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. San Jose: Cisco Internet Business Solutions Group (IBSG), 2011.

HANWEI ELECTRONICS. *Technical Data MQ-135 Gas Sensor*. Zhengzhou: Hanwei Electronics Co., Ltd, [s.d.]. Disponível em: https://www.olimex.com/Products/Components/Sensors/Gas/SNS-MQ135/resources/SNS-MQ135.pdf. Acesso em: nov. 2024.

MCROBERTS, Michael. *Beginning Arduino*. 2. ed. New York: Apress, 2013.

MONK, Simon. *Programming Arduino: Getting Started with Sketches*. 2. ed. New York: McGraw-Hill Education, 2016.

GLOSSÁRIO

Arduino Cloud: Plataforma baseada em nuvem que permite conectar, gerir e monitorizar dispositivos IoT remotamente. No projeto, foi utilizada para criar o *dashboard* de visualização dos dados.

Dashboard: Painel de controlo visual, acessível via navegador ou aplicação, onde são exibidos gráficos, indicadores e botões para monitorização do sistema em tempo real.

Datasheet: Folha de dados técnicos fornecida pelo fabricante de um componente eletrónico, contendo especificações como tensão de operação, precisão e limites de uso.

DHT22: Sensor digital de temperatura e humidade de baixo custo e precisão intermédia, utilizado para a recolha de dados climáticos no protótipo.

ESP32: Microcontrolador de baixo custo e baixo consumo de energia, com suporte nativo a Wi-Fi e Bluetooth, que atua como o "cérebro" do sistema embarcado.

GPIO (General Purpose Input/Output): Pinos de entrada e saída de uso geral do microcontrolador, utilizados para ligar sensores (como o DHT22) e atuadores (como os LEDs).

IDE (Integrated Development Environment): Ambiente de Desenvolvimento Integrado. Software utilizado para escrever, compilar e carregar o código (sketch) para o microcontrolador.

IoT (Internet of Things): Internet das Coisas. Conceito que se refere à interconexão digital de objetos quotidianos com a internet, permitindo a recolha e troca de dados.

MQ-135: Sensor analógico de qualidade do ar, sensível a gases como amónia, benzeno, álcool e dióxido de carbono (CO), utilizado para monitorizar a salubridade do ambiente.

Sketch: Nome dado ao código-fonte escrito na linguagem de programação utilizada pelo ecossistema Arduino/ESP32.

Wi-Fi: Tecnologia de comunicação sem fios baseada nos padrões IEEE 802.11, utilizada pelo ESP32 para enviar os dados recolhidos para a nuvem.

Apêndices

APÊNDICE A – CÓDIGO FONTE COMPLETO DO FIRMWARE

As seções a seguir apresentam o código-fonte completo utilizado no sistema. **Todas as credenciais foram anonimizadas por segurança.**

A.1 Arquivo *main.ino*

```
#include "thingProperties.h"
#include

#define DHTPIN 21
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

#define LED_VERDE 25
#define LED_VERMELHO 26

#define MQ135PIN 35
#define WINDOW 20
int mqBuffer[WINDOW];
int mqIndex = 0;

float calcularMediaMQ135() {
  long soma = 0;
  for (int i = 0; i < WINDOW; i++) soma += mqBuffer[i];
  return soma / (float)WINDOW;
}

void setup() {
  Serial.begin(9600);
  delay(1500);

  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  dht.begin();
  pinMode(LED_VERDE, OUTPUT);
  pinMode(LED_VERMELHO, OUTPUT);

  digitalWrite(LED_VERDE, LOW);
  digitalWrite(LED_VERMELHO, LOW);

  analogSetPinAttenuation(MQ135PIN, ADC_11db);

  for (int i = 0; i < WINDOW; i++) mqBuffer[i] = 0;
}

void loop() {
  ArduinoCloud.update();

  float t = dht.readTemperature();
  float h = dht.readHumidity();

  if (!isnan(t)) {
```

```
48      temperatura = t;
49      dht_status = true;
50    } else dht_status = false;
51
52    if (!isnan(h)) umidade = h;
53
54    int raw = analogRead(MQ135PIN);
55    mqBuffer[mqIndex] = raw;
56    mqIndex = (mqIndex + 1) % WINDOW;
57
58    float media = calcularMediaMQ135();
59
60    mq_status = (raw > 20);
61
62    int qualidade = map(media, 0, 4095, 100, 0);
63    qualidade_do_ar = qualidade;
64
65    if (qualidade > 80) alerta_qualidade = "Excelente";
66    else if (qualidade > 60) alerta_qualidade = "Bom";
67    else if (qualidade > 40) alerta_qualidade = "Regular";
68    else if (qualidade > 20) alerta_qualidade = "Ruim";
69    else alerta_qualidade = "Perigoso";
70
71    if (temperatura < 30 && qualidade_do_ar > 50) {
72      digitalWrite(LED_VERDE, HIGH);
73      digitalWrite(LED_VERMELHO, LOW);
74    } else {
75      digitalWrite(LED_VERDE, LOW);
76      digitalWrite(LED_VERMELHO, HIGH);
77    }
78
79    Serial.println("\n========== LEITURA DO SISTEMA ==========");
80    Serial.print("Temperatura: "); Serial.println(temperatura);
81    Serial.print("Umidade: "); Serial.println(umidade);
82    Serial.print("MQ135 RAW: "); Serial.println(raw);
83    Serial.print("MQ135 Média: "); Serial.println(media);
84    Serial.print("Qualidade do Ar: "); Serial.println(qualidade_do_
        ar);
85    Serial.print("Estado: "); Serial.println(alerta_qualidade);
86    Serial.println("========================================");
87
88    delay(2000);
89  }
```

A.2 Arquivo *thingProperties.cpp*

```
1 #include "thingProperties.h"
2
3 CloudTemperature temperatura;
4 CloudPercentage  umidade;
5 CloudPercentage  qualidade_do_ar;
6 String alerta_qualidade;
7 CloudSwitch dht_status;
8 CloudSwitch mq_status;
9
```

```
10 WiFiConnectionHandler ArduinoIoTPreferredConnection("SSID_REDE",
     "SENHA_REDE");
11
12 void initProperties() {
13   ArduinoCloud.setBoardId("DEVICE_ID_XXXX");
14   ArduinoCloud.setSecretDeviceKey("DEVICE_KEY_XXXX");
15
16   ArduinoCloud.addProperty(temperatura, READ, ON_CHANGE,
       onTemperaturaChange);
17   ArduinoCloud.addProperty(umidade, READ, ON_CHANGE,
       onUmidadeChange);
18   ArduinoCloud.addProperty(qualidade_do_ar, READ, ON_CHANGE,
       onQualidadedoarChange);
19   ArduinoCloud.addProperty(alerta_qualidade, READ, ON_CHANGE,
       onAlertaqualidadeChange);
20
21   ArduinoCloud.addProperty(dht_status, READ);
22   ArduinoCloud.addProperty(mq_status, READ);
23 }
24
25 void onTemperaturaChange() {}
26 void onUmidadeChange() {}
27 void onQualidadedoarChange() {}
28 void onAlertaqualidadeChange() {}
```

A.3 Arquivo *thingProperties.h*

```
1 #ifndef THINGPROPERTIES_H
2 #define THINGPROPERTIES_H
3
4 #include
5 #include
6
7 extern CloudTemperature temperatura;
8 extern CloudPercentage  umidade;
9 extern CloudPercentage  qualidade_do_ar;
10 extern String alerta_qualidade;
11 extern CloudSwitch dht_status;
12 extern CloudSwitch mq_status;
13
14 const char DEVICE_LOGIN_NAME[] = "DEVICE_ID_XXXX";
15 const char DEVICE_KEY[] = "DEVICE_KEY_XXXX";
16
17 const char SSID[] = "SSID_REDE";
18 const char PASS[] = "SENHA_REDE";
19
20 void onTemperaturaChange();
21 void onUmidadeChange();
22 void onQualidadedoarChange();
23 void onAlertaqualidadeChange();
24
25 void initProperties();
26
27 extern WiFiConnectionHandler ArduinoIoTPreferredConnection;
28
```

```
29 #endif
```