



Relatório de Projeto de Internet das Coisas (IoT)

Projeto 01

Gregorio Pinheiro Cunha
Joaquim Chianca Dantas Beserra
Matheus Almeida Fontes



Projeto 01

Disciplina: IMD0902 - Internet das Coisas - 2024.1

Orientador: Prof. Heitor Medeiros Florencio

Universidade Federal do Rio Grande do Norte

Descrição do projeto

Projeto IoT de monitoramento de ambiente de biblioteca, com o objetivo de registrar a temperatura e a umidade do local, verificando se ambas estão dentro de valores adequados para correta preservação do acervo.

Resumo

Este relatório apresenta o desenvolvimento e implementação de um projeto de Internet das Coisas (IoT) voltado para o monitoramento ambiental de uma biblioteca. O objetivo principal foi garantir a preservação dos acervos de livros e documentos, controlando a temperatura e a umidade do ar, além de prevenir futuros prejuízos, como esquecer o ar-condicionado ligado por longos períodos. Para isso, foi utilizado sensor de umidade e temperatura (DHT11) conectado a um microcontrolador ESP32, que transmitiu os dados coletados para a plataforma Adafruit IO via Wi-Fi, utilizando o protocolo MQTT (Message Queuing Telemetry Transport).

A metodologia do projeto incluiu o levantamento de requisitos, a programação dos dispositivos, a prototipação e a configuração do sistema na plataforma Adafruit IO. O sistema foi projetado para realizar medições contínuas, possibilitando acompanhar em tempo real os valores de temperatura e umidade.

Os resultados indicaram que o sistema funcionou de maneira eficaz, proporcionando leituras precisas e consistentes, além de permitir uma visualização intuitiva dos dados em tempo real. O monitoramento contínuo possibilita a intervenção rápida para manter as condições ambientais ideais, contribuindo significativamente para a preservação dos acervos e prevenindo incidentes como o esquecimento do sistema de refrigeração ligado sem necessidade.

A discussão abordou a eficácia do sistema, a facilidade de manutenção e escalabilidade, o impacto positivo na preservação dos acervos e algumas limitações e possíveis melhorias futuras, como a integração com outros tipos de sensores e o controle remoto de dispositivos presentes no ambiente. Em conclusão, o projeto demonstrou ser uma solução eficaz e acessível para o monitoramento ambiental de bibliotecas, assegurando a integridade dos acervos e permitindo uma gestão ambiental eficiente.



1. Introdução

As bibliotecas são ambientes que abrigam acervos valiosos de livros, documentos e outros materiais impressos que são suscetíveis a danos causados por condições ambientais inadequadas, especialmente a temperatura e a umidade. A umidade alta pode promover o crescimento de fungos e mofo, enquanto temperaturas extremas podem acelerar a degradação do papel e das encadernações. Manter esses parâmetros dentro de limites seguros é essencial para preservar a integridade dos acervos e a saúde das pessoas que frequentam o ambiente.

Neste contexto, surge a necessidade de um sistema de monitoramento contínuo e eficaz das condições ambientais dentro da biblioteca, visando garantir a conservação adequada dos acervos. Este projeto propõe a implementação de um sistema de Internet das Coisas (IoT) para monitoramento de temperatura e umidade em tempo real, utilizando um sistema com sensores conectados à internet para coleta e análise dos dados.

Para a realização deste projeto, foram empregadas tecnologias específicas, incluindo um sensor de umidade e temperatura (DHT11), uma placa de desenvolvimento contendo microcontrolador com conectividade Wi-Fi (ESP32) e a plataforma Adafruit IO para gerenciamento e visualização dos dados na nuvem. A montagem do circuito foi feita em uma protoboard, e a programação foi desenvolvida na IDE Arduino, utilizando bibliotecas adequadas para a leitura dos sensores e a comunicação com a plataforma de dados.

Os objetivos principais deste sistema são garantir que a biblioteca mantenha as condições ambientais dentro dos parâmetros ideais para a preservação dos livros e documentos e monitoramento do consumo energético dos dispositivos elétricos e eletrônicos presentes, permitindo intervenções rápidas quando necessário. Com isso, espera-se prolongar a vida útil dos acervos e assegurar que a biblioteca continue a cumprir sua missão de conservar e disponibilizar o conhecimento para as futuras gerações.

2. Arquitetura IoT

Na figura 01, abaixo, é possível observar a arquitetura IoT do presente projeto:

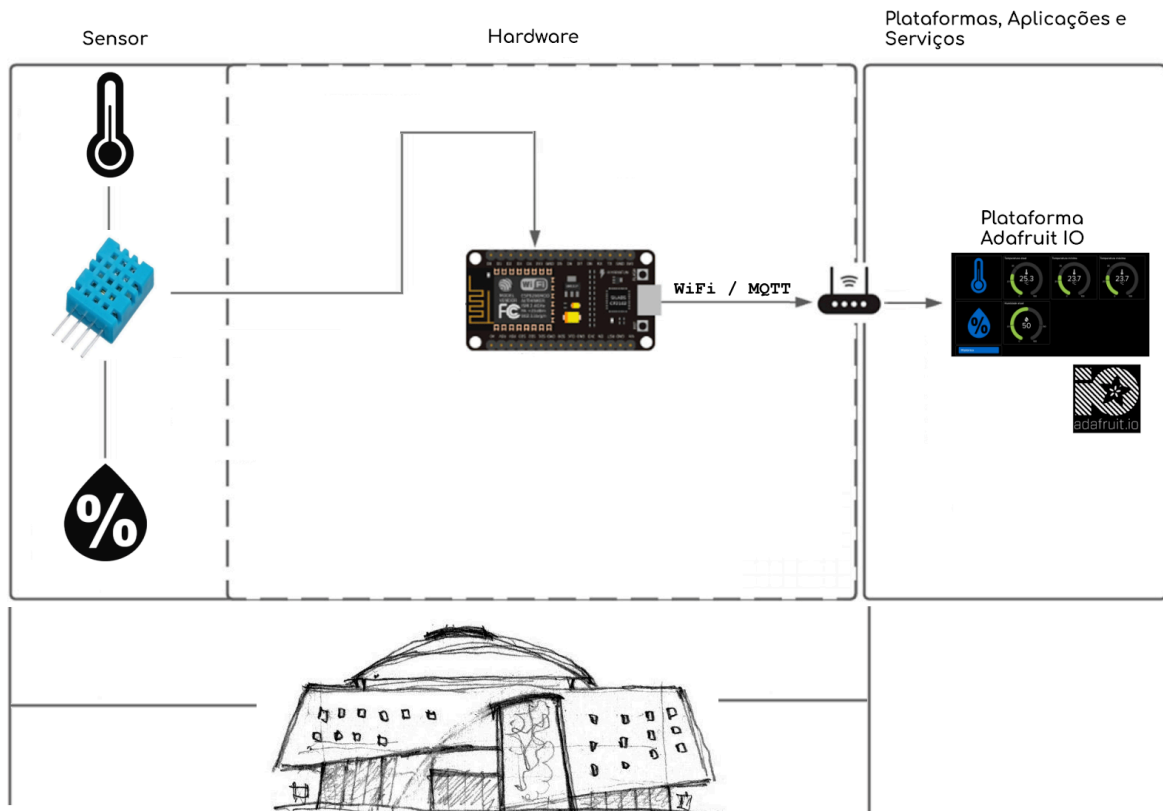


Figura 01 - Arquitetura IoT do projeto.

A arquitetura IoT do projeto está representada com 3 camadas. A primeira camada contempla o *sensor DHT11* de temperatura e umidade.

Na segunda camada, de *Hardware*, temos a placa de desenvolvimento com ESP32 recebendo os dados do sensor e se comunicando, através dos protocolos WiFi e MQTT (Message Queuing Telemetry Transport), com a plataforma Adafruit IO.

Por fim, na terceira camada, está presente a plataforma Adafruit IO, com os dashboards dos dados coletados pelo sensor.

Abaixo da primeira e segunda camadas está a representação da biblioteca, ambiente monitorado pelo projeto desenvolvido.



3. Metodologia

Segue, nas próximas subseções, a metodologia de desenvolvimento do projeto.

3.1. Dispositivos IoT

Coletamos os dados de temperatura e umidade através do sensor DHT11 conectado ao ESP32 e, após processamento no microcontrolador, enviamos os dados de temperatura atual, temperatura mínima, temperatura máxima e umidade para a plataforma Adafruit IO, fazendo uso dos protocolos WiFi e MQTT.

Abaixo, na Figura 02, é possível ver a montagem do circuito na plataforma Wokwi e, em seguida, na Figura 03, a montagem do circuito real. Na plataforma Wokwi utilizamos o sensor DHT22 para representar o sensor DHT11, por este não estar presente na plataforma.

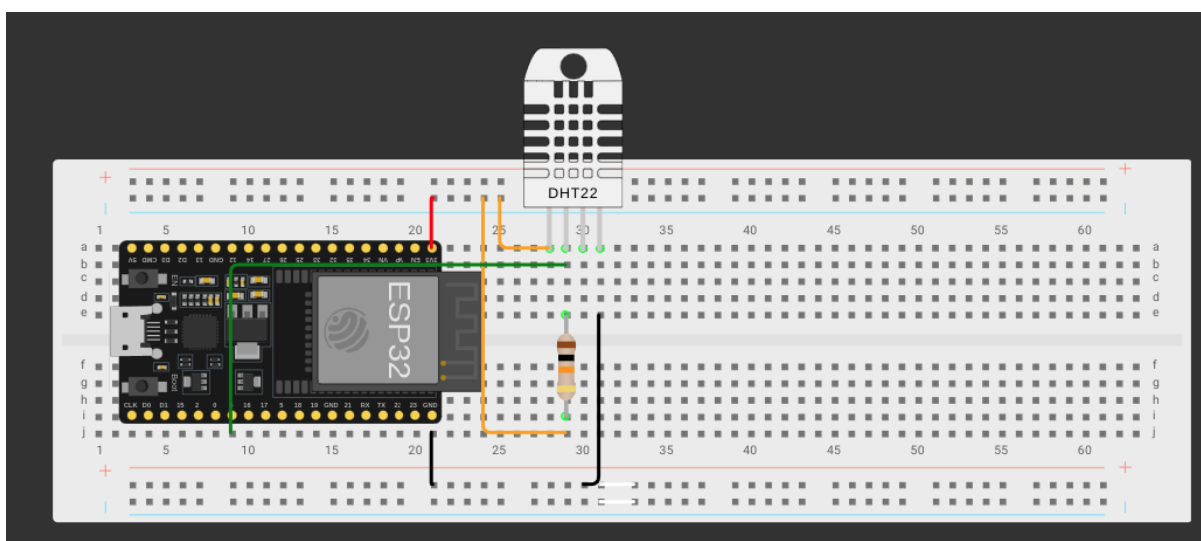


Figura 02 - Montagem do circuito na plataforma Wokwi.

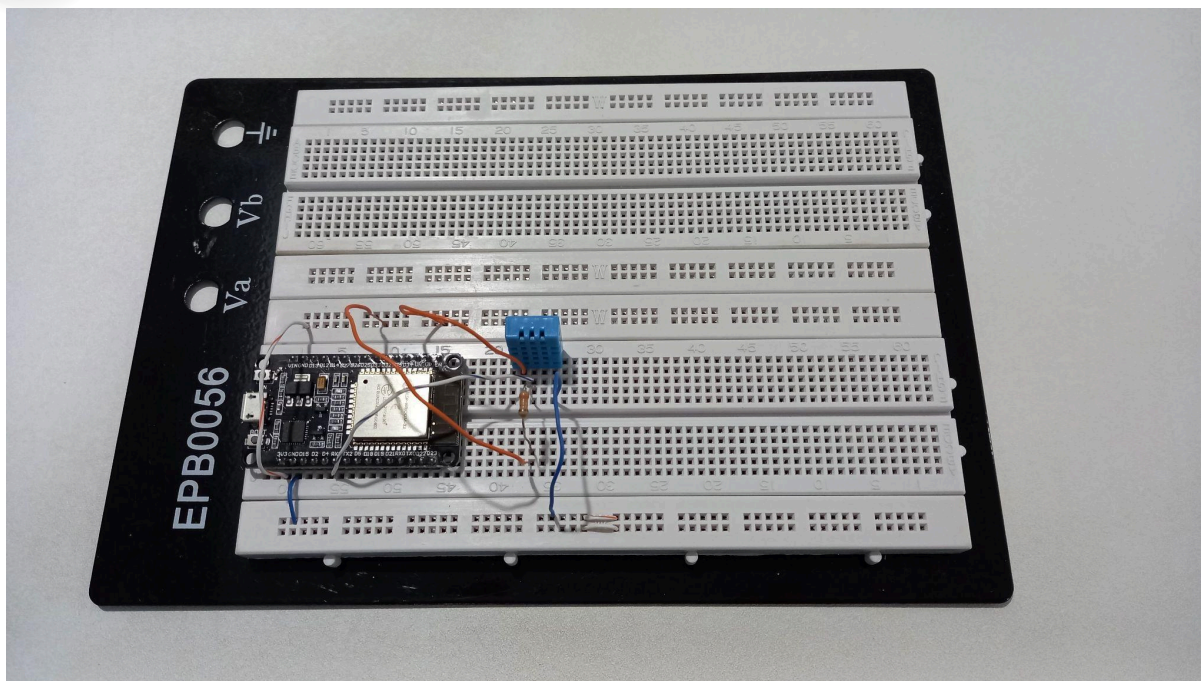


Figura 03 - Montagem do circuito real.

A função setup é apresentada a seguir. Nela é feita a inicialização da Serial e do sensor DHT11, a configuração do pino de entrada dos sinais do DHT11 e do pino de saída para ligar o LED da placa indicando conexão com a plataforma Adafruit IO e a conexão com essa plataforma.



```
46 void setup() {  
47     Serial.begin(115200);  
48  
49     pinMode(DHTPin, INPUT);  
50     pinMode(2, OUTPUT);  
51  
52     dht.begin();  
53  
54     // wait for serial monitor to open  
55     while(! Serial);  
56  
57     // connect to io.adafruit.com  
58     Serial.println("Conectando com Adafruit IO ");  
59     io.connect();  
60  
61     // wait for a connection  
62     while(io.status() < AIO_CONNECTED) {  
63         Serial.print(".");  
64         delay(500);  
65     }  
66  
67     // we are connected  
68     Serial.println();  
69     Serial.println(io.statusText());  
70     digitalWrite(2, HIGH);  
71 }
```

Figura 04 - Código da função *setup*.

A função *loop* apenas chama as funções *io.run* e *publish*.

```
73 void loop() {  
74     io.run();  
75     publish();  
76 }
```

Figura 05 - Código da função *loop*.

A função *publish* chama as funções para atualizar a temperatura e a umidade, faz a conexão das variáveis com os feeds da plataforma Adafruit IO, chama a função para imprimir os dados na saída serial e aguarda pelo tempo estabelecido, neste caso de 1 minuto.



```
78 void publish() {
79     updateTemperature();
80     updateHumidity();
81
82     temperatureFeed->save(temperature);
83     maxTemperatureFeed->save(maxTemperature);
84     minTemperatureFeed->save(minTemperature);
85
86     humidityFeed->save(humidity);
87     // maxHumidityFeed->save(maxHumidity);
88     // minHumidityFeed->save(minHumidity);
89
90     printData();
91
92     delay(publishingInterval);
93 }
94
```


Figura 06 - Código da função *publish*.

```
95 void updateTemperature() {
96     temperature = dht.readTemperature();
97     if(maxTemperature < temperature) {
98         maxTemperature = temperature;
99     }
100     if(minTemperature > temperature) {
101         minTemperature = temperature;
102     }
103 }
```

Figura 07 - Código da função *updateTemperature*.

```
105 void updateHumidity() {
106     humidity = dht.readHumidity();
107 }
108
```

Figura 08 - Código da função *updateHumidity*.



```
115 void printData() {
116     Serial.print("Temperatura atual: ");
117     Serial.println(temperature);
118     Serial.print("Temperatura mínima: ");
119     Serial.println(minTemperature);
120     Serial.print("Temperatura máxima: ");
121     Serial.println(maxTemperature);
122
123     Serial.print("Humidade: ");
124     Serial.println(humidity);
125 }
```

Figura 09 - Código da função *printData*.

O código completo pode ser encontrado em: <https://github.com/gregopc/IloT>

3.2. Conectividade

Utilizamos a comunicação WiFi de 2,4 GHz do ESP32 para conectá-lo à internet e o protocolo MQTT (Message Queuing Telemetry Transport) para mandar os dados, via internet, para a plataforma da Adafruit IO.

3.3. Plataformas, Aplicações e Serviços

Utilizamos a plataforma da Adafruit IO para conectar o ESP32, possibilitando o monitoramento e análise dos dados coletados e apresentados em dashboards.

Além de medições realizadas em ambiente domiciliar, coletamos, a cada 1 minuto, os dados de temperatura e umidade do ambiente de biblioteca por aproximadamente 1 hora, entre 8h00 e 9h00 da manhã.

Montamos duas dashboards, uma contendo os registros de temperatura e umidade atuais, temperatura mínima e temperatura máxima registradas, e outra contendo os históricos de temperatura e de umidade. Essas dashboards podem ser vistas nas Figuras 04 e 05, respectivamente.

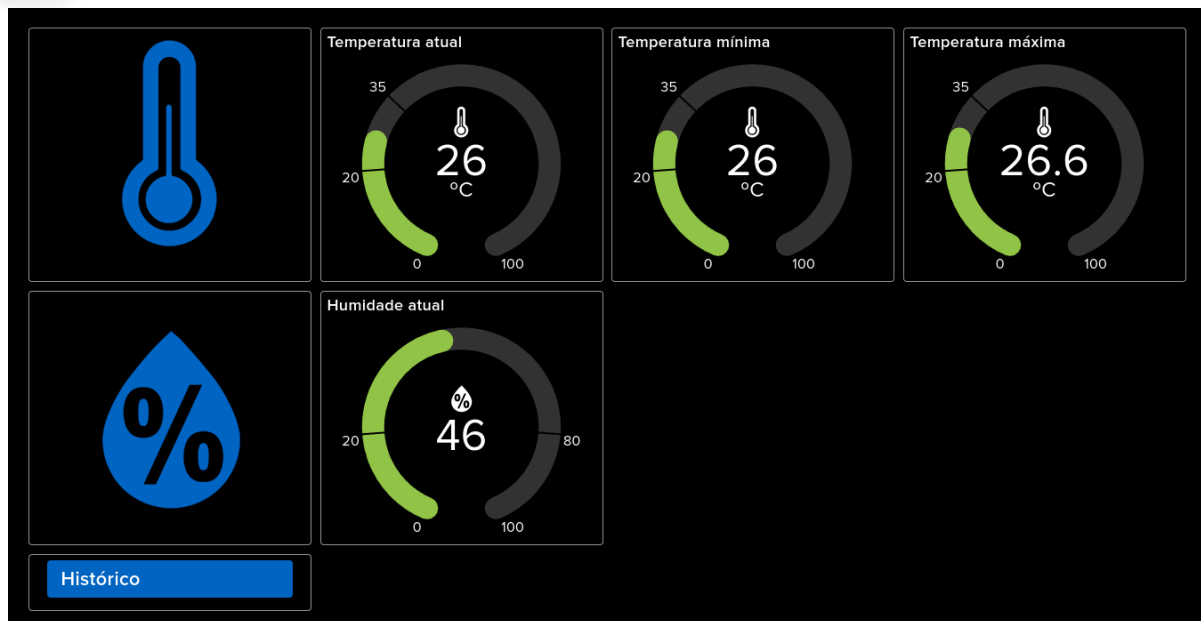


Figura 10 - Dashboard do projeto na plataforma Adafruit IO.



Figura 11 - Dashboard apresentando histórico na plataforma Adafruit IO.



4. Resultados e discussões

O projeto desenvolvido atendeu as expectativas dos requisitos propostos, permitindo o monitoramento, em tempo real, das grandezas de temperatura e umidade do ambiente, fundamental para a correta preservação do acervo, ainda possibilitando, de modo indireto, verificar o uso do sistema de refrigeração, ajudando a garantir maior eficiência energética do ambiente.

Contudo, para uma solução final, a ser instalada no ambiente de biblioteca, precisaríamos de conexão duradoura à fonte de energia, seja bateria ou rede elétrica, e encapsulamento do circuito. Também seria interessante adotar outra estratégia para a aplicação web, tendo em vista que a plataforma da Adafruit IO possui limitações na quantidade de feeds, dashboards e dados enviados. Inclusive elaboramos, inicialmente, dashboard com temperaturas e umidades atuais, mínimas e máximas, mas acabamos retirando o envio dos valores mínimos e máximos de umidade por ter excedido o limite de dados da plataforma para usuários sem assinatura paga.

Outra limitação que observamos foi em relação a preservação dos dados de temperaturas mínimas e máximas que, por serem armazenados apenas em variáveis do programa, eram perdidos ao se reiniciar o ESP32. Uma solução para essa limitação seria armazenar esses valores na memória flash interna do microcontrolador e recuperá-los assim que ocorresse uma reinicialização.

Para aprimoramentos posteriores, seria interessante realizar a conexão a um servidor NTP (Network Time Protocol) para obter informações de data e hora e utilizar o SPIFFS (Serial Peripheral Interface Flash File System) para armazenar os valores de temperatura, umidade e demais valores derivados desses, na memória interna do ESP32. Também poderiam ser adicionados outros sensores, tais quais de presença e de luminosidade e atuadores para, por exemplo, permitir o controle do sistema de iluminação e de refrigeração.



5. Referências

[1] CACPNRJ. Interface DHT11 DHT22 com ESP32 e valores de exibição usando servidor Web - Cap Sistema. Disponível em:
<<https://capsistema.com.br/index.php/2020/12/03/interface-dht11-dht22-com-esp32-e-valores-de-exibicao-usando-servidor-web/>>.