RELATÓRIO CARDIO IA

ARMAZENAMENTO E PROCESSAMENTO LOCAL (EDGE COMPUTING):

Nome: Diego Nunes Veiga

RM:560658

Turma: 2TIAOR

1. OBJETIVO

O objetivo desta segunda parte do projeto **CardioIA** é demonstrar a aplicação prática dos conceitos de **Fog e Cloud Computing** em um sistema embarcado desenvolvido na plataforma **Wokwi**, utilizando o microcontrolador **ESP32**.
A proposta consiste em dar continuidade à arquitetura iniciada na Parte 1 (Edge Computing), agora implementando a camada de **comunicação e visualização em nuvem**, com foco no protocolo **MQTT (Message Queuing Telemetry Transport)** e na exibição dos dados em dashboards como o **Node-RED**.

O projeto tem como finalidade simular um sistema vestível de monitoramento cardíaco, capaz de coletar dados fisiológicos — como temperatura corporal e oxigenação do sangue — e transmiti-los para uma camada de nuvem.
Por meio dessa estrutura, é possível compreender como o **ESP32** atua como nó intermediário, enviando informações processadas na borda para serviços externos responsáveis por análise e visualização, demonstrando o papel da **computação em névoa e nuvem** na Internet das Coisas aplicada à saúde digital.

2. EXPLICAÇÃO DO DESENVOLVIMENTO DO SOFTWARE

O software da Parte 2 foi desenvolvido na IDE Arduino, utilizando a biblioteca PubSubClient para a implementação da comunicação MQTT.
A aplicação foi executada e testada na simulação Wokwi, replicando as funcionalidades básicas de um ambiente conectado. Embora não tenha sido possível realizar a integração prática com o Node-RED ou o envio real de dados para um broker MQTT, todo o código foi estruturado de forma compatível com esses serviços, de modo que a implementação física possa ser facilmente completada.

Durante a inicialização, o programa configura os parâmetros de rede Wi-Fi e define as credenciais do broker MQTT. O ESP32, ao ser executado, realiza a tentativa de conexão à rede e, em seguida, à instância do broker. Uma vez estabelecida a comunicação, o microcontrolador passa a publicar periodicamente mensagens MQTT em tópicos

específicos, contendo os valores simulados dos sensores de temperatura, umidade e oxigenação.

O código possui três componentes principais:

- **Conexão Wi-Fi**, responsável por estabelecer o acesso à rede;
- **Conexão MQTT**, encarregada de gerenciar a sessão com o broker, autenticar e publicar mensagens;
- **Leitura e publicação de dados**, que realiza a coleta dos sensores (DHT22 e potenciômetro) e organiza as informações em formato JSON.

A lógica central da aplicação é baseada em um **loop contínuo**, no qual o sistema:

1. Garante a manutenção da conexão com o broker (reconectando quando necessário);
2. Lê os valores dos sensores;
3. Monta uma mensagem JSON com os parâmetros medidos;
4. Publica as mensagens em três tópicos distintos, por exemplo:
    - cardioia/temperatura
    - cardioia/umidade
    - cardioia/oxigenacao
5. Aguarda um pequeno intervalo antes da próxima leitura, simulando um comportamento periódico de aquisição e transmissão de dados.

A estrutura geral da mensagem publicada segue o padrão JSON simplificado, conforme o exemplo abaixo:

{"temperatura": 36.8, "umidade": 58.7, "oxigenacao": 97.2}

Para o tratamento de falhas, o código inclui uma função de reconexão automática, que identifica a perda de comunicação com o broker e tenta restabelecê-la, evitando a interrupção do fluxo de dados. Todas as mensagens e eventos de conexão são exibidos no Monitor Serial, servindo como uma forma de depuração e validação do funcionamento do protocolo.

A implementação foi projetada para operar com qualquer broker MQTT público (como HiveMQ Cloud ou Mosquitto), bastando configurar o endereço, a porta e as credenciais de acesso. Mesmo no ambiente Wokwi, o código cumpre a função de simular o envio contínuo de dados, preparando o sistema para integração futura com aplicações reais de nuvem.

3. SUBALGORITMOS E EXPLICAÇÕES

O sistema foi dividido em subalgoritmos que garantem modularidade e clareza no código.
A seguir, estão descritas as principais funções e seus papéis dentro da aplicação:

- **Configuração Wi-Fi:**
  Inicializa o módulo Wi-Fi do ESP32 e tenta se conectar à rede especificada. Caso a conexão falhe, o dispositivo realiza novas tentativas até obter sucesso.

- **Conexão MQTT:**
  Função responsável por configurar o cliente MQTT, definir o servidor e gerenciar a reconexão automática.
  Utiliza a função client.connect() para autenticação e client.loop() para manter o vínculo ativo com o broker.

- **Leitura dos Sensores:**
  Realiza a leitura dos dados provenientes do sensor **DHT22** (temperatura e umidade) e do **potenciômetro**, simulando um segundo sinal vital (oxigenação ou batimentos).
  Esses valores são processados e armazenados em variáveis numéricas para posterior envio.

- **Formatação das Mensagens JSON:**
  Organiza os dados coletados em uma estrutura JSON, com chaves correspondentes a cada parâmetro. Essa abordagem facilita o consumo dos dados por sistemas externos, como o Node-RED.

- **Publicação MQTT:**
  Envia as mensagens formatadas para os tópicos predefinidos utilizando o comando client.publish(topic, message). Esse processo ocorre em intervalos regulares, simulando uma transmissão contínua de dados médicos.

- **Reconexão Automática:**
  Subalgoritmo que verifica periodicamente o estado da conexão MQTT e executa novamente o processo de login caso a comunicação com o broker seja interrompida.
  Exibe mensagens no monitor serial indicando sucesso ou falha na reconexão.

Esses subalgoritmos tornam o código escalável e adaptável, permitindo que novas métricas sejam incluídas e publicadas facilmente em diferentes tópicos, sem alterar a estrutura principal do programa.

4. RESULTADOS OBTIDOS

Durante os testes no ambiente de simulação Wokwi, foi possível validar a inicialização do Wi-Fi, o estabelecimento da conexão MQTT simulada e a publicação periódica das

mensagens nos tópicos definidos. As leituras do sensor DHT22 e do potenciômetro foram realizadas corretamente, e as mensagens JSON foram exibidas no monitor serial conforme o esperado.

Embora o sistema não tenha sido integrado ao Node-RED nem a um broker real, o fluxo teórico foi validado, confirmando a funcionalidade da lógica de publicação e a compatibilidade do código com o protocolo MQTT.

O projeto, portanto, atingiu o objetivo de demonstrar a camada de comunicação e transmissão em nuvem, consolidando o entendimento sobre como o ESP32 pode atuar como publicador MQTT em aplicações reais de IoT médico.

5. CONCLUSÃO

O projeto CardioIA – Parte 2 representa o avanço da arquitetura iniciada na primeira fase, incorporando o conceito de computação em névoa e nuvem à solução embarcada.
Mesmo sem a integração prática ao Node-RED, o desenvolvimento realizado demonstra o domínio dos conceitos de publicação de dados MQTT, formatação JSON, reconexão automática e transmissão periódica de informações sensoriais.

A implementação mostra que o ESP32 é capaz de atuar como um nó inteligente dentro de uma arquitetura distribuída, enviando informações de forma eficiente, segura e escalável.
Em um cenário físico, esse mesmo código poderia ser conectado a um broker MQTT real, permitindo a visualização dos dados em dashboards como Node-RED ou Grafana Cloud, com alertas automáticos e análise de histórico.

Com isso, o sistema desenvolvido cumpre os objetivos propostos na Fase 3 do projeto CardioIA, evidenciando a transição entre a borda (Edge) e a **nuvem (Cloud)**, pilares fundamentais da **Internet das Coisas aplicada à saúde digital**.