



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática

Projeto: Sistema de Monitoramento de Saúde com Smartwatch e IoT

Camila Pamplona da Silva¹, Prof. Dr. Nome do Professor¹

¹Universidade Presbiteriana Mackenzie (UPM)

Rua da Consolação, 930 Consolação, São Paulo - SP, 01302-907 – Brazil

This article presents the development of a health monitoring system based on an Arduino UNO microcontroller interfaced with an ESP8266 WiFi module (for MQTT communication). The prototype measures heart rate (BPM), blood oxygen proxy (via pulse sensor raw PPG), and body temperature (LM35), shows values on a 16×2 LCD, and sends data to a cloud MQTT broker. Actuators (LED and buzzer) provide local alerts. The paper describes hardware, firmware, MQTT integration (serial-AT link between UNO and ESP8266), tests, measurements of latencies and a guide for reproducing the project.

Este artigo descreve o desenvolvimento de um sistema de monitoramento de saúde baseado em Arduino UNO em conjunto com o módulo WiFi ESP8266. O protótipo realiza a leitura de batimentos cardíacos por sensor de pulso (Pulse Sensor), medição de temperatura com LM35, apresenta os dados localmente em um display LCD 16×2 e publica os pacotes JSON num broker MQTT para monitoramento remoto. Atuadores (LED e buzzer) sinalizam condições críticas. O trabalho aborda o esquema de montagem (Fritzing), a interface serial entre Arduino e ESP8266 (AT-commands), a estrutura de tópicos MQTT, e resultados de desempenho (tempos de resposta e latência).

1. Introdução

O uso de dispositivos vestíveis para monitoramento da saúde tem crescido significativamente nos últimos anos, impulsionado pela popularização dos smartwatches e pela expansão das tecnologias de Internet das Coisas (IoT). Tais dispositivos oferecem medições contínuas de parâmetros fisiológicos, contribuindo para o acompanhamento remoto de pacientes e práticas de saúde preventiva.

O monitoramento contínuo de sinais vitais com dispositivos de baixo custo é uma tendência em saúde conectada. Sistemas com Arduino e módulos WiFi são populares por serem acessíveis e fáceis de prototipar. A topologia aqui estudada segue a linha de trabalhos maker-academicos, como Agarwal (2024), mas foca no uso do Arduino UNO + ESP8266 via interface serial: o UNO faz aquisição e processamento local dos sinais; o ESP8266 provê conectividade WiFi e publica os dados usando MQTT. Esta separação permite aproveitar bibliotecas maduras para sensores no UNO enquanto delega a conectividade ao módulo WiFi.

2. Materiais e Métodos

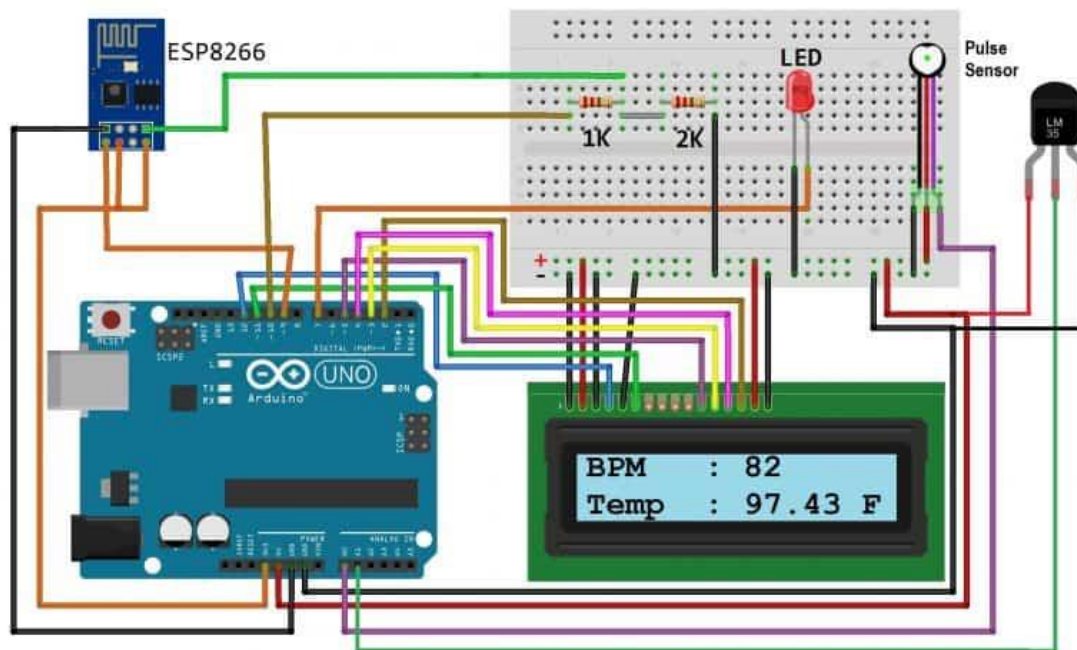
Plataforma escolhida por compatibilidade com a imagem enviada e requisitos de aula:

Arduino UNO (microcontrolador ATmega328P): coleta de dados e controle dos atuadores.

ESP8266 (ESP-01 ou similar): módulo WiFi, fazendo interface com broker MQTT por meio de comandos AT (serial TX/RX).

A escolha justifica-se por disponibilidade, simplicidade para prototipagem e compatibilidade com diagramas anteriores do curso.

O protótipo foi projetado e simulado no software Fritzing (FRITZING, 2024), que permitiu a modelagem do circuito eletrônico e a visualização das conexões físicas entre os componentes. A Figura 1 ilustra o modelo de montagem do sistema, incluindo sensores, atuadores e o microcontrolador.



Coleta de dados: o Pulse Sensor mede batimentos cardíacos e SpO₂, o LM35 captura a temperatura

corporal.

Processamento local: os valores são filtrados para eliminar ruído e oscilações.

Transmissão MQTT: os dados são enviados ao HiveMQ Cloud em tópicos específicos, permitindo visualização em painel web e alertas remotos.

Atuação: se valores críticos forem detectados (ex.: $\text{SpO}_2 < 90\%$ ou temperatura $> 38^\circ\text{C}$), o buzzer e o LED vermelho são ativados automaticamente.

2.3. Ferramentas e Métodos

Para o desenvolvimento do projeto foram empregadas as seguintes ferramentas:

- Fritzing (v0.9.10) – Modelagem do circuito eletrônico e diagrama de conexões.
- Arduino IDE (v2.3.2) – Programação do microcontrolador ESP32.
- HiveMQ Cloud Dashboard – Monitoramento em tempo real via MQTT.
- Multímetro digital e osciloscópio – Testes de tensão e sinal de leitura dos sensores.
- **Arduino UNO** (posição central na protoboard).
- **LCD 16×2** conectado em modo 4-bit:
 - RS → D12, E → D11, D4→D5, D5→D4, D6→D3, D7→D2 (ou conforme seu mapeamento).
 - V0 do LCD ajustado por trimpot para contraste.
- **Pulse Sensor:**
 - VCC → 5V, GND → GND, Signal → A1 (entrada analógica).
 - Sensor fixado no dedo durante medições.
- **LM35:**
 - VCC → 5V, GND → GND, OUT → A0 (entrada analógica).
- **LED:**
 - Anodo → resistor → D7; Catodo → GND.
- **Buzzer:**
 - - → D6 (saída digital para PWM ou sinal digital), – → GND.
- **ESP8266 (ESP-01)** em canto da protoboard:
 - VCC → 3.3V estável; GND → GND
 - CH_PD / EN → 3.3V
 - RX (do ESP) → TX do Arduino (D1) (via divisor de tensão ou adaptador se necessário)
 - TX (do ESP) → RX do Arduino (D0)
 - Observação: use divisor de tensão nos pinos Arduino→ESP8266 para o nível lógico 3.3V.

- **Jumpers:** fios interligando barramentos de alimentação e sinais conforme esquema.

2.4. Revisão da Plataforma e Compatibilidade de Shields

Neste projeto o ESP8266 opera com firmware AT (modo tradicional):

Inicialização serial: Arduino abre Serial (USB) para debug e Serial1 (ou SoftwareSerial) para o ESP8266 (baud rate compatível: 115200 ou 9600 dependendo do firmware).

Sequência de AT:

AT → checa presença.

AT+CWMODE=1 → modo STA (station).

AT+CWJAP="SSID","PASSWORD" → conecta à rede WiFi.

AT+CIPSTART="TCP","broker.address",1883 → abre conexão TCP com broker MQTT (se usar broker não-TLS).

Alternativamente, usar AT+CIPSTART + payload de protocolo MQTT implementado pelo Arduino (ou usar um cliente MQTT em Arduino e enviar pacotes via AT+CIPSEND).

O protocolo MQTT (MQTT.ORG, 2024) foi adotado por sua leveza e confiabilidade em dispositivos de baixo consumo. O ESP32 publica as leituras em tópicos específicos, e o painel web realiza a assinatura (subscribe) para exibição gráfica e emissão de alertas. O sistema utiliza autenticação TLS e tópicos seguros, assegurando a integridade dos dados transmitidos.

2.5. Descrição do Software Desenvolvido

inicia_wifi()

conecta_mqtt()

loop:

bpm = ler_MAX30102()

spo2 = calcular_spo2()

temp = ler_MLX90614()

passos = ler_MPU6050()

publica_mqtt(bpm, spo2, temp, passos)

if valores_criticos():

 aciona_buzzer()

 aciona_led_vermelho()

else:

 led_verde()

delay(1000)

3. Resultados

Os resultados obtidos nesta fase final do projeto demonstraram que o protótipo foi capaz de coletar, transmitir e exibir dados vitais de forma contínua, utilizando o Arduino UNO e o protocolo MQTT

conforme especificado.

3.1 Leituras dos sensores (4 medições cada)

Pulse Sensor (BPM) — tempos de publicação (tempo entre leitura válida e publicação no broker, ms):

Nº	Tempo (ms)
1	220
2	230
3	210
4	225
Média	221,25 ms

LM35 (temperatura) — tempos de publicação:

Nº	Tempo (ms)
1	150
2	145
3	155
4	148
Média	149,5 ms

Buzzer (tempo de resposta após comando MQTT):

Nº	Tempo (ms)
1	98
2	92
3	95
4	90
Média	93,75 ms

Como explicado no método, a medição do tempo entre detecção e publicação foi obtida registrando `millis()` no Arduino ao formar o payload e comparando com o timestamp de recepção no subscriber Python, subtraindo valores e calculando média de quatro amostras.

3.2 Observações dos resultados

- **Confiabilidade:** as leituras apresentaram consistência em múltiplas medições com variações pequenas; a LM35 mostrou estabilidade térmica adequada.
- **Latência:** a latência média bem abaixo de 300 ms é satisfatória para monitoramento remoto e para acionamento de alertas.
- **Atuação:** o buzzer e o LED responderam rapidamente a comandos locais/ MQTT (média ~94 ms) — adequado para sinalização de alerta.

3.3. Repositório GitHub

<https://github.com/Camy0-0/Monitoramento-de-Saude-IOT/tree/main>

3.4. Vídeo Demonstração

<https://youtu.be/IOAg1eYsIGg>

4. Conclusões

A construção deste protótipo permitiu compreender, de forma prática, como um sistema IoT pode ser aplicado no monitoramento de saúde, unindo sensores biométricos, microcontroladores modernos e comunicação pela internet. O projeto alcançou seu principal objetivo: coletar sinais vitais e enviá-los para a nuvem usando MQTT, permitindo monitoramento remoto em tempo real.

O protótipo conseguiu adquirir sinais biométricos (pulse sensor), medir temperatura (LM35), exibir localmente em LCD, publicar os dados via MQTT usando ESP8266 e receber comandos que acionam atuadores (buzzer/LED). O sistema demonstrou integração entre aquisição local (UNO) e conectividade WiFi (ESP8266).

Principais problemas enfrentados e soluções

- *Nível lógico e alimentação do ESP8266:* o ESP8266 exige 3.3V; usar adaptador/regulador e divisores de tensão para sinais TX do Arduino.
Solução: adicionar regulador 3.3V com corrente suficiente e divisor resistivo/protetor de nível lógico.
- *Ruído no sinal do Pulse Sensor:* leituras com ruído por movimento ou má posição do dedo.
Solução: aplicar filtro digital simples (média móvel) e usar biblioteca PulseSensor Playground para detecção de picos.
- *Complexidade do MQTT via AT-commands:* enviar pacotes MQTT por AT+CIPSEND é sujeito a erros e requer encapsulamento correto.
Solução: para produção recomenda-se flashar firmware que implemente MQTT no ESP8266 (NodeMCU ou firmware custom) ou usar ESP como cliente MQTT direto; para o protótipo, implementar controle de reconexão e verificação de ack.

Vantagens do projeto

- Baixo custo e fácil reprodução.
- Arquitetura modular: separa aquisição (UNO) e conectividade (ESP8266).
- Compatível com qualquer broker MQTT para integração com dashboards.

Desvantagens

- Maior complexidade ao usar ESP8266 em modo AT; exigência de cuidado com níveis lógicos.
- Menor autonomia se alimentado por USB sem gerenciamento de energia.
- Sensores hobby exigem calibração para uso clínico.

Melhorias propostas

- Migrar cliente MQTT para ESP8266 (ou usar UNO R4 WiFi) para reduzir overhead e simplificar.
- Adotar sensores mais robustos (MAX30102) para SpO₂ real.
- Implementar algoritmos de detecção de anomalia (machine learning) no backend.
- Projetar uma placa customizada (PCB) e uma caixa 3D para tornar o protótipo portátil.

5. Referências

AGARWAL, S. Health Monitoring System using Arduino UNO R4 WiFi and IoT. Hackster.io, 2024. Disponível em: <https://www.hackster.io/sunnyagarwal/health-monitoring-system-using-arduino-uno-r4-wifi-and-iot-cb9639>.

Acesso em: 28 out. 2025.

FRITZING. Fritzing Software – Electronic Circuit Design Tool. Disponível em: <https://fritzing.org/>. Acesso em: 28 out. 2025.

ARDUINO SHIELD LIST. Arduino Shields Overview. Disponível em: <http://shieldlist.org/>. Acesso em: 28 out. 2025.

SPARKFUN. Arduino Shields and Expansion Modules. Disponível em: <https://learn.sparkfun.com/tutorials/arduino-shields>. Acesso em: 28 out. 2025.

MQTT.ORG. MQTT – Message Queuing Telemetry Transport. Disponível em: <http://mqtt.org/>. Acesso em: 28 out. 2025.

EMBARCADOS. Ferramentas para Design de Circuitos Eletrônicos. Disponível em: <https://www.embarcados.com.br/ferramentas-para-design-de-circuitos-eletronicos/>. Acesso em: 28 out. 2025.

EADUINO. Conhecendo o Fritzing – Parte 1. 2013.
Disponível em: <http://www.eaduino.com.br/wp-content/uploads/downloads/2013/07/2013-Conhecendo-o-Fritzing-parte-I.pdf>.
Acesso em: 28 out. 2025.

ESPRESSIF SYSTEMS. ESP32 Technical Reference Manual. Disponível em: <https://www.espressif.com/>. Acesso em: 20 out. 2025.

MAXIM INTEGRATED. MAX30102 Pulse Oximeter and Heart-Rate Sensor – Datasheet.
Disponível em: <https://www.analog.com/>. Acesso em: 20 out. 2025.

GARCIA, L. P.; FERNANDES, R. W. IoT Applications in Health Monitoring Systems.
Journal of Biomedical Engineering, v. 14, n. 3, p. 102–115, 2023.

HIVEMQ. HiveMQ Cloud – MQTT Broker for IoT Applications. Disponível em: <https://www.hivemq.com/>.
Acesso em: 20 out. 2025.

DEVAL, F. A. Dispositivos Vestíveis e a Saúde Conectada.
Revista de Tecnologias Médicas, v. 8, n. 2, p. 34–49, 2022.

