SISTEMA IOT PARA MONITORAMENTO DE MOTORES NAVAIS

PROJETO FINAL – SISTEMAS EMBARCADOS COM IOT

CHARLES XAVIER, JOSÉ VITINEY, SOFIA MAYUMI

CONTEXTO DO PROJETO

- Cenário inspirado em operações em alto-mar (Open Sea)
- Monitoramento de motores navais em operação contínua
- Ambientes críticos e de difícil acesso
- Necessidade de coleta e análise remota de dados
- Aplicação de IoT para aumento da confiabilidade operacional

PROBLEMA

- Motores navais operam continuamente em ambientes críticos
- Falhas podem causar altos custos e riscos operacionais
- Monitoramento manual é limitado e pouco eficiente
- Falta de visibilidade em tempo real do estado do motor

OBJETIVO DO PROJETO

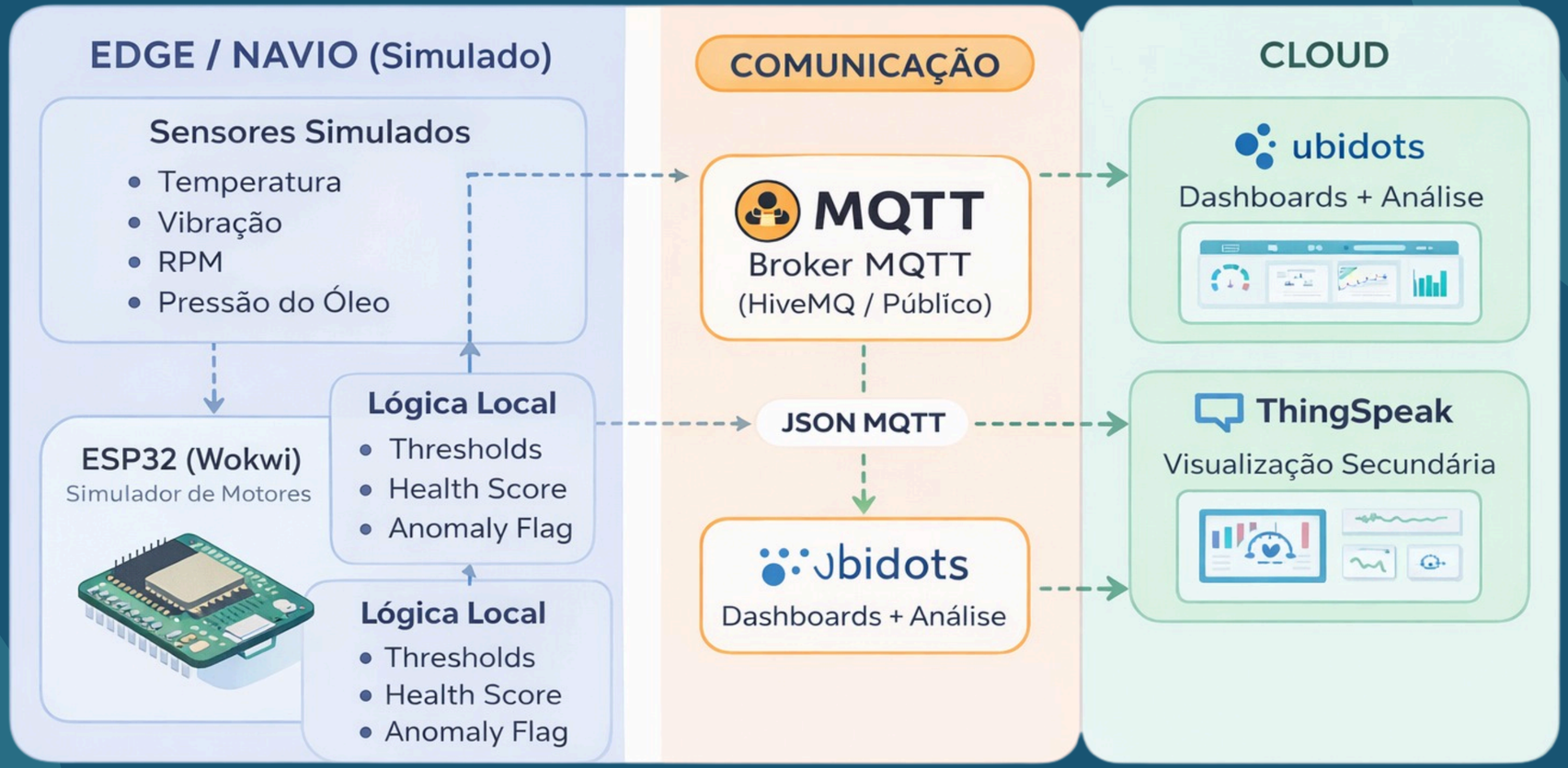
- Simular um sistema IoT para monitoramento de motores navais
- Coletar dados operacionais por sensores embarcados
- Realizar pré-processamento no edge device
- Enviar dados para a nuvem via MQTT
- Disponibilizar dados para análise e visualização

WOKWI

MQTT

ubidots

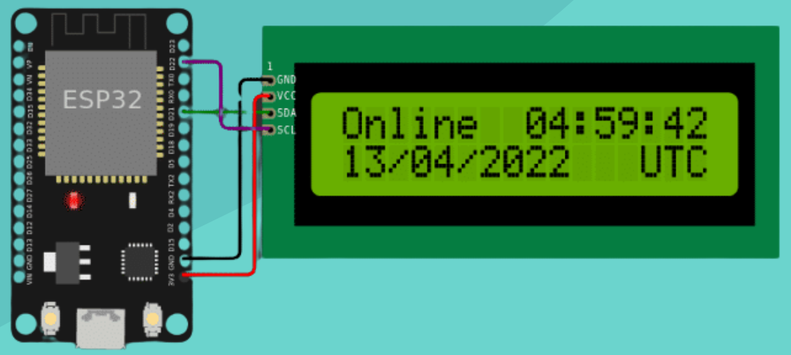
ARQUITETURA GERAL DO SISTEMA



ESP32

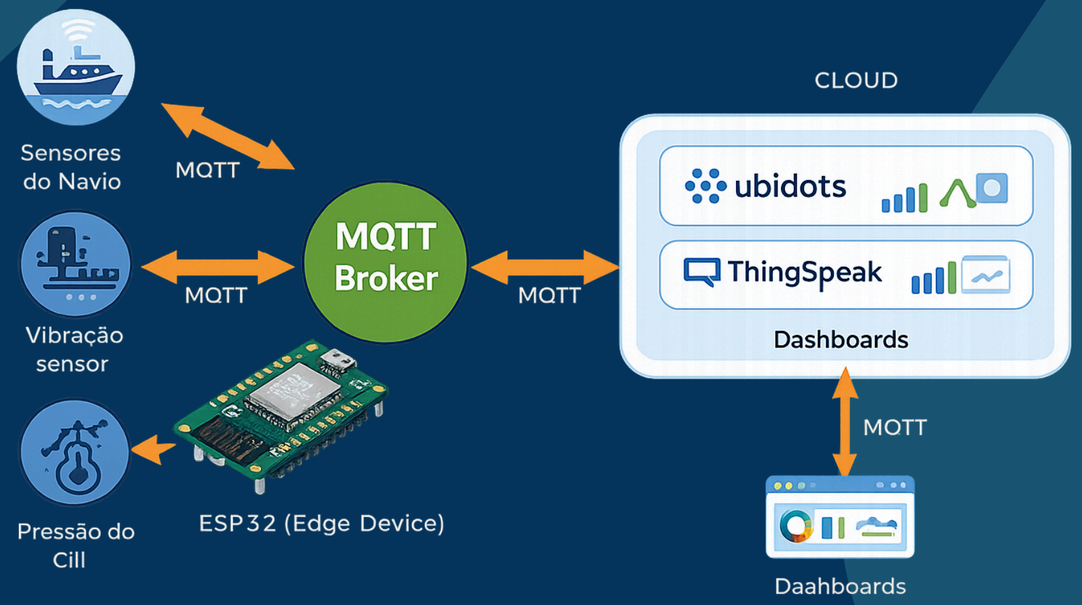
- ESP32 simulado no Wokwi
- Representa o device embarcado do navio
- Coleta dados dos sensores simulados
- Executa lógica local (health score e anomalias)
- Envia dados processados via MQTT

WOKWi



COMUNICAÇÃO MQTT

- Protocolo MQTT para comunicação IoT
- Comunicação leve e eficiente
- Uso de broker público (HiveMQ)
- Envio de dados do edge para a cloud
- Mensagens estruturadas em formato JSON



CÓDIGO

```
1   #include <WiFi.h>
2   #include <PubSubClient.h>
3   #include <string.h>
4
```

- Bibliotecas para Wi-Fi, MQTT e manipulação de strings
- Base para comunicação e integração IoT

```
// ========================
// CONFIG WIFI (Wokwi)
// ========================
const char* WIFI_SSID = "Wokwi-GUEST";
const char* WIFI_PASS = "";

// ========================
// UBIDOTS MQTT (Industrial)
// ========================
const char* MQTT_HOST = "industrial.api.ubidots.com";
const int   MQTT_PORT = 1883;

// ⚠️ RECOMENDADO: revogar este token e gerar outro no Ubidots.
// Depois, não versionar no GitHub (use secrets.h).
const char* UBIDOTS_TOKEN = ;

// No Ubidots: USER = TOKEN
const char* MQTT_USER = UBIDOTS_TOKEN;

// Em muitos casos pode ser vazio; se falhar, coloque UBIDOTS_TOKE
const char* MQTT_PASS = "";

// Devices no Ubidots (um por motor)
const char* DEVICE_LABEL_ENGINE_A = "ship01-engine-a";
const char* DEVICE_LABEL_ENGINE_B = "ship01-engine-b";

// Tópicos Ubidots
String topicA = String("/v1.6/devices/") + DEVICE_LABEL_ENGINE_A;
String topicB = String("/v1.6/devices/") + DEVICE_LABEL_ENGINE_B;
```

- Configuração de Wi-Fi no ambiente Wokwi
- Parâmetros de conexão com o Ubidots
- Definição de dispositivos e tópicos MQTT

```
// ========================
// REGRAS (edge intelligence)
// ========================
int calcHealthScore(float tempC, float vibRms, int rpm, float oilBar)
  int score = 100;

  if (tempC > 85) score -= (int)((tempC - 85) * 2);
  if (tempC > 95) score -= (int)((tempC - 95) * 4);

  if (vibRms > 3.0) score -= (int)((vibRms - 3.0) * 15);
  if (vibRms > 6.0) score -= 20;

  if (rpm < 1400) score -= (1400 - rpm) / 10;
  if (rpm > 1900) score -= (rpm - 1900) / 10;

  if (oilBar < 2.8) score -= (int)((2.8 - oilBar) * 30);
  if (oilBar < 2.2) score -= 25;

  if (score < 0) score = 0;
  if (score > 100) score = 100;
  return score;
```

- Avaliação da condição do motor
- Uso de limites operacionais
- Processamento realizado localmente no ESP32

```
int calcAnomalyFlag(float tempC, float vibRms, int rpm, float oilBar)
  bool tempCrit = tempC > 98;
  bool vibCrit  = vibRms > 6.5;
  bool oilCrit  = oilBar < 2.2;
  bool rpmCrit  = (rpm < 1350) || (rpm > 2000);
  return (tempCrit || vibCrit || oilCrit || rpmCrit) ? 1 : 0;
```

- Identificação de condições críticas
- Geração de flag de anomalia

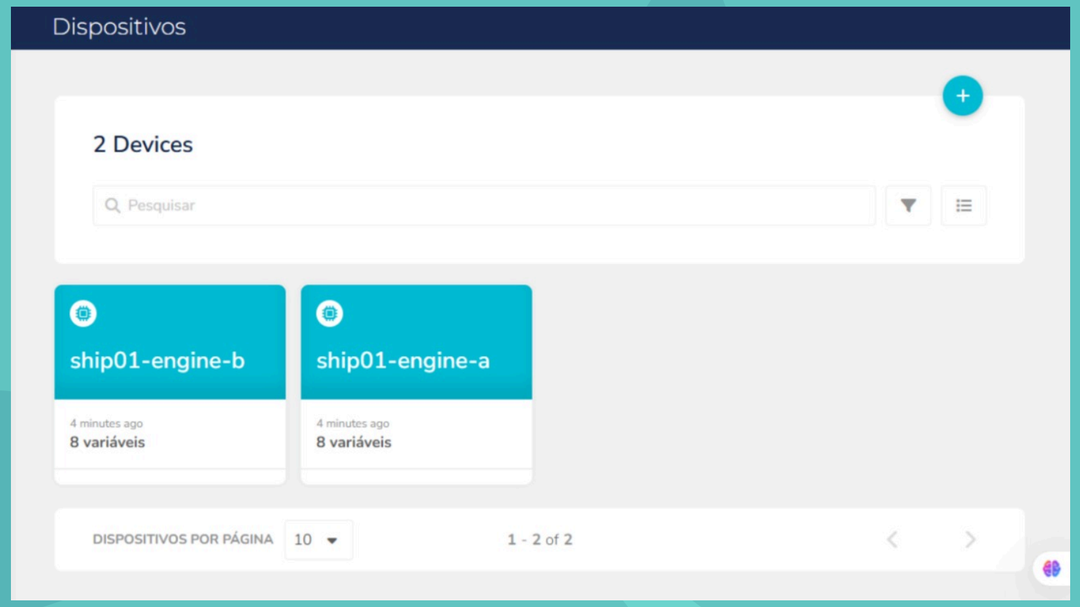
```
mqtt.publish(topicA.c_str(), payloadA.c_str());
mqtt.publish(topicB.c_str(), payloadB.c_str());

Serial.println(payloadA);
Serial.println(payloadB);
```

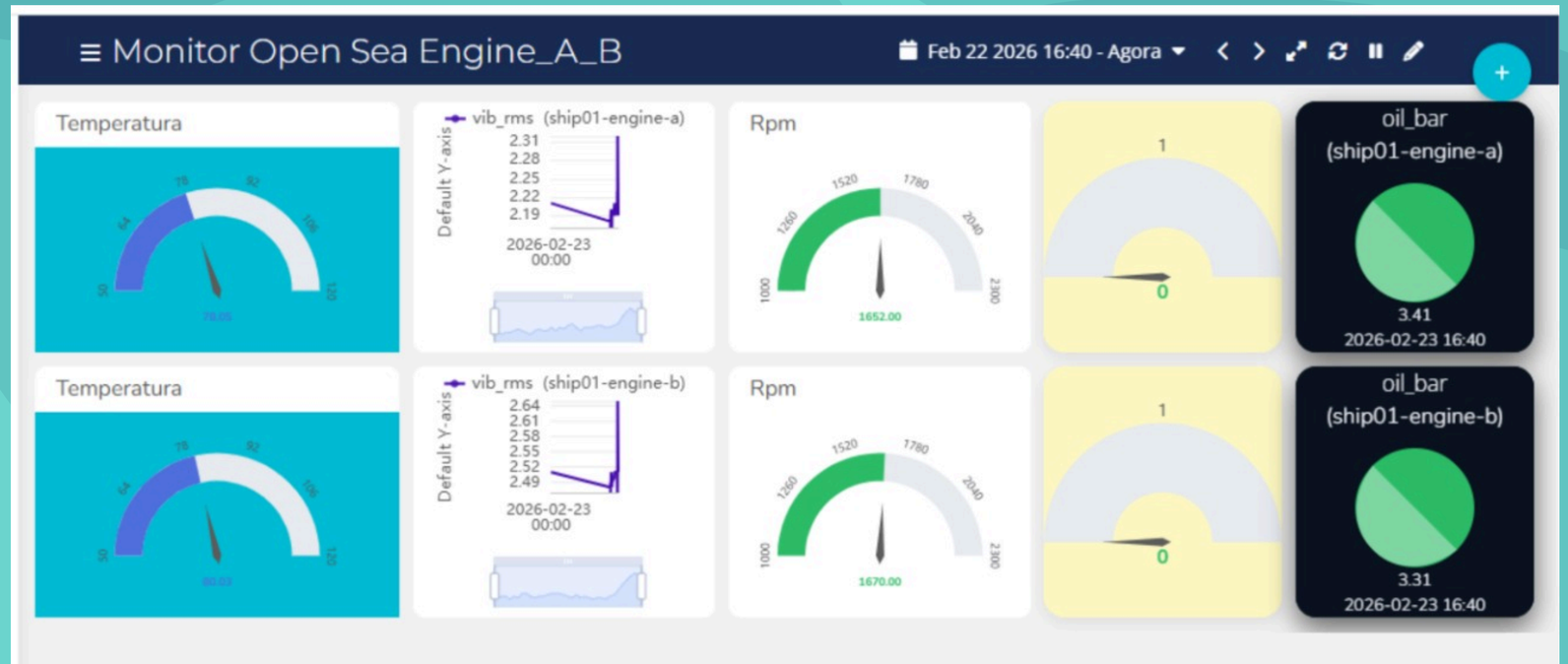
- Envio de dados processados
- Comunicação assíncrona e eficiente
- Integração com dashboards na cloud

CLOUD-UBIDOTS

- Ubidots como plataforma principal de análise
- Recebimento de dados via MQTT
- Armazenamento e visualização em dashboards
- ThingSpeak utilizado como visualização complementar

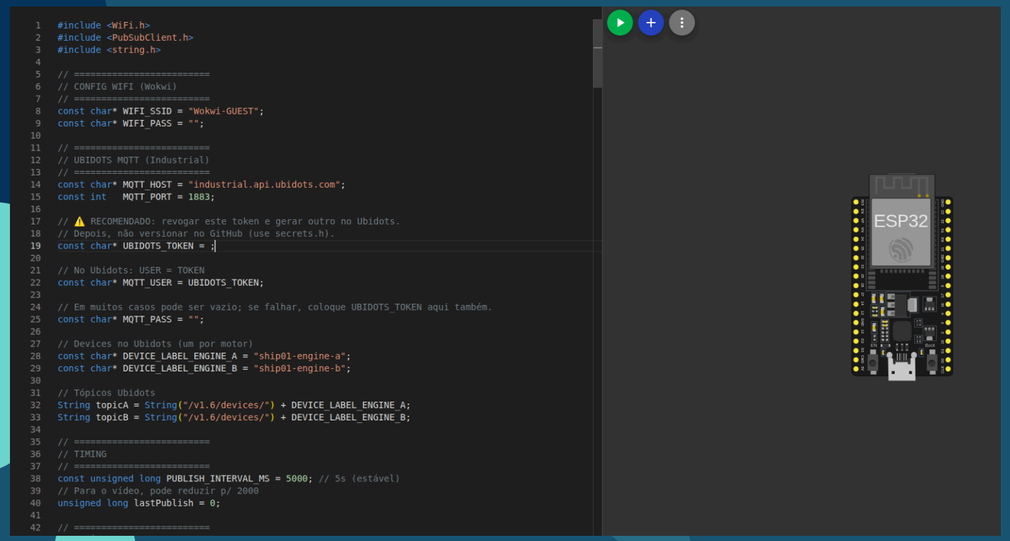
DASHBOARDS


Dispositivos
2 Devices
Pesquisar
ship01-engine-b
4 minutes ago
8 variáveis
ship01-engine-a
4 minutes ago
8 variáveis
DISPOSITIVOS POR PÁGINA
10
1 - 2 of 2


Monitor Open Sea Engine_A_B
Feb 22 2026 16:40 - Agora
Temperatura
vib_rms (ship01-engine-a)
Default Y-axis
2.31
2.28
2.25
2.22
2.19
2026-02-23
00:00
Rpm
1000
1260
1520
1780
2040
2300
1652.00
1
0
oil_bar
(ship01-engine-a)
3.41
2026-02-23 16:40
Temperatura
vib_rms (ship01-engine-b)
Default Y-axis
2.64
2.61
2.58
2.55
2.52
2.49
2026-02-23
00:00
Rpm
1000
1260
1520
1780
2040
2300
1670.00
1
0
oil_bar
(ship01-engine-b)
3.31
2026-02-23 16:40

TESTES REALIZADOS

- Simulação completa no Wokwi
- Testes de conexão Wi-Fi
- Testes de comunicação MQTT
- Validação dos cenários: normal, stress e falha
- Monitoramento via Serial Monitor



LIMITAÇÕES DO PROJETO

- Projeto baseado em ambiente de simulação
- Sensores simulados (não físicos)
- Segurança implementada em nível básico
- Sem integração com sistemas reais de bordo
- Limitação no número de variáveis nos dashboards devido ao plano gratuito do Ubidots

CONCLUSÃO

- Sistema IoT completo e funcional
- Integração entre edge, comunicação e cloud
- Aplicação de edge intelligence
- Projeto alinhado a um cenário real
- Atende aos requisitos do projeto final