BOCA
BOLETIM DE CONJUNTURA

UTILIZANDO A IOT NO SISTEMA DE TELEMETRIA E MONITORAMENTO EM TEMPO REAL PARA A GESTÃO DE DESLOCAMENTO TÉCNICO (GARAGEM-PERCURSO-DESTINO) NO TRANSPORTE COLETIVO

USING IOT IN A TELEMETRY AND REAL-TIME MONITORING SYSTEM FOR MANAGING TECHNICAL TRAVEL (GARAGE-ROUTE-DESTINATION) IN PUBLIC TRANSPORTATION

UTILIZACIÓN DE IOT EN UN SISTEMA DE TELEMETRÍA Y MONITORIZACIÓN EN TIEMPO REAL PARA LA GESTIÓN DE DESPLAZAMIENTOS TÉCNICOS (TALLER-RUTA-DESTINO) EN EL TRANSPORTE PÚBLICO

10.56238/bocav25n77-012

Paulo Henrique Polato
Graduando em Ciência da Computação

Caique Zaneti Kirilo
Professor Msc.

RESUMO
O artigo irá propor uma solução baseada em Internet das Coisas (IoT) para que haja o monitoramento do "KM morto" – o trajeto entre garagem e início da operação comercial. Utilizando microcontroladores (ESP32), geolocalização (GPS) e o protocolo MQTT para que haja a visualização de possíveis atrasos na partida e a eficiência operacional, que irá gerar uma proatividade na gestão antes mesmo do início da linha comercial.

Palavras-chave: Internet das Coisas (IoT). Telemetria em Tempo Real. Gestão de Transporte Coletivo.

ABSTRACT
This article proposes an Internet of Things (IoT)-based solution for monitoring the "dead mile" – the route between the garage and the start of commercial operation. Using microcontrollers (ESP32), geolocation (GPS), and the MQTT protocol, it will allow for the visualization of potential departure delays and operational efficiency, generating proactive management even before the start of commercial service.

Keywords: Internet of Things (IoT). Real-Time Telemetry. Public Transportation Management.

RESUMEN
Este artículo propone una solución basada en el Internet de las Cosas (IoT) para monitorizar el tramo intermedio entre la cochera y el inicio de la operación comercial. Mediante microcontroladores (ESP32), geolocalización (GPS) y el protocolo MQTT, permitirá visualizar posibles retrasos en las salidas y la eficiencia operativa, generando una gestión proactiva incluso antes del inicio del servicio comercial.

Palabras clave: Internet de las Cosas (IoT). Telemetría en Tiempo Real. Gestión del Transporte Público.

1 INTRODUÇÃO

Temos hoje a gestão do transporte publico focada na linha comercial, mas há um neglicenciamento do trajeto técnico.

Onde pode ocorrer a quebra do veículo, enfrentar um congestionamento saindo da garagem ou mesmo o desvio da rota, seja ela por interdição de ruas ou desvio de conduta do operador.

Sendo o problema somente percebido quando o passageiro já está esperando o ônibus no ponto de parada.

O artigo irá apresentar o desenvolvimento de um sistema de monitoramento focado especificamente na gestão do deslocamento técnico, suprindo a falta de dados em tempo real nessa janela crítica que impacta toda a trajetória de deslocamento.

2 METODOLOGIA

Camada de Coleta (Edge): Montagem do sensor com ESP32, modulo GP Neo – 6M e conexão GPRS/4G, com Implementação de lógica de Geofencing pra identificar as zonas de Garagem e Destino

Este esboço inicial configura as comunicações seriais para capturar dados do GPS e inicializar o modem.

```
#include <TinyGPS++.h>
#include <HardwareSerial.h>

// --- CONFIGURAÇÕES DE GEOFENCING ---
const double LAT_GARAGEM = -23.5505; // Substitua pelas suas coordenadas
const double LNG_GARAGEM = -46.6333;
const double LAT_DESTINO = -23.5596;
const double LNG_DESTINO = -46.6581;
const float RAIO_CERCA   = 50.0;     // Raio em metros

// --- PINOS E SERIAIS ---
#define RX_GPS 16
#define TX_GPS 17
#define RX_GSM 26
#define TX_GSM 27
TinyGPSPlus gps;
HardwareSerial SerialGPS(1);
HardwareSerial SerialGSM(2);
```

```
void setup() {
  Serial.begin(115200);
  SerialGPS.begin(9600, SERIAL_8N1, RX_GPS, TX_GPS);
  SerialGSM.begin(115200, SERIAL_8N1, RX_GSM, TX_GSM);
    Serial.println("Rastreador Iniciado. Aguardando fix de satélites...");
}
void loop() {
  while (SerialGPS.available() > 0) {
    if (gps.encode(SerialGPS.read())) {
      verificarGeofence();
    }
  }
}
void verificarGeofence() {
  if (gps.location.isValid()) {
    double latAtual = gps.location.lat();
    double lngAtual = gps.location.lng();
    // Cálculo da distância para a Garagem
    double distGaragem = TinyGPSPlus::distanceBetween(latAtual, lngAtual, LAT_GARAGEM,
LNG_GARAGEM);
    // Cálculo da distância para o Destino
    double distDestino = TinyGPSPlus::distanceBetween(latAtual, lngAtual, LAT_DESTINO,
LNG_DESTINO);
    Serial.print("Lat: "); Serial.print(latAtual, 6);
    Serial.print(" | Lng: "); Serial.println(lngAtual, 6);
    if (distGaragem <= RAIO_CERCA) {
      Serial.println("STATUS: [DENTRO DA GARAGEM]");
      // Aqui podemos enviar um comando via GSM/4G
    }
    else if (distDestino <= RAIO_CERCA) {
      Serial.println("STATUS: [CHEGOU AO DESTINO]");
    }
    else {
      Serial.println("STATUS: [EM TRÂNSITO]");
    }
```

```
  }
}
```

- Camada de Transporte (Network): Uso do protocolo MQTT, por sua leveza e capacidade de lidar com instabilidade de redes em movimento

```
#include <TinyGPS++.h>
#include <TinyGsmClient.h>
#include <PubSubClient.h>
#include <HardwareSerial.h>
// --- CONFIGURAÇÕES GSM/GPRS ---
const char apn[]      = "://vivo.com.br"; // Altere conforme sua operadora
const char gprsUser[] = "";
const char gprsPass[] = "";
// --- CONFIGURAÇÕES MQTT ---
const char* mqttServer = "://hivemq.com";
const int mqttPort     = 1883;
const char* topicPath  = "rastreador/esp32/status";
// --- COORDENADAS GEOFENCING ---
const double LAT_GARAGEM = -23.5505;
const double LNG_GARAGEM = -46.6333;
const float RAIO_CERCA   = 50.0;
// --- HARDWARE E OBJETOS ---
#define TINY_GSM_MODEM_SIM800  // Ou SIM7600 se for 4G
HardwareSerial SerialGPS(1);
HardwareSerial SerialGSM(2);
TinyGsm modem(SerialGSM);
TinyGsmClient client(modem);
PubSubClient mqtt(client);
TinyGPSPlus gps;
unsigned long lastSend = 0;
void setup() {
  Serial.begin(115200);
  SerialGPS.begin(9600, SERIAL_8N1, 16, 17);
  SerialGSM.begin(115200, SERIAL_8N1, 26, 27);
  Serial.println("Iniciando Modem...");
  if (!modem.restart()) { Serial.println("Falha no modem"); return; }
```

```
  Serial.println("Conectando GPRS...");
  if (!modem.gprsConnect(apn, gprsUser, gprsPass)) { Serial.println("Falha GPRS"); return; }
    mqtt.setServer(mqttServer, mqttPort);
}
void reconectarMQTT() {
  while (!mqtt.connected()) {
    Serial.print("Conectando MQTT...");
    if (mqtt.connect("ESP32_Rastreador_01")) {
      Serial.println("Conectado!");
    } else {
      delay(5000);
    }
  }
}
void loop() {
  if (!mqtt.connected()) reconectarMQTT();
  mqtt.loop();
  while (SerialGPS.available() > 0) {
    if (gps.encode(SerialGPS.read())) {
      enviarDadosGPS();
    }
  }
}
void enviarDadosGPS() {
  // Envia a cada 10 segundos para não sobrecarregar o GPRS
  if (millis() - lastSend > 10000 && gps.location.isValid()) {
    lastSend = millis();
    double dist = TinyGPSPlus::distanceBetween(gps.location.lat(), gps.location.lng(),
LAT_GARAGEM, LNG_GARAGEM);
    String status = (dist <= RAIO_CERCA) ? "NA_GARAGEM" : "EM_TRANSITO";
    // Criando JSON simples para o payload
    String payload = "{\"lat\":" + String(gps.location.lat(), 6) +
                     ",\"lng\":" + String(gps.location.lng(), 6) +
                     ",\"status\":\"" + status + "\"}";
    mqtt.publish(topicPath, payload.c_str());
```

```
    Serial.println("Payload enviado: " + payload);
  }
}
```

Por que esta estrutura é ideal?

TinyGsmClient: Faz a "ponte" entre o modem serial e o protocolo TCP/IP, permitindo que o MQTT funcione como se estivesse em um Wi-Fi.

JSON Payload: O formato { "lat": x, "lng": y } é o padrão para integração com dashboards (Node-RED, TagoIO ou ThingsBoard).

Economia de Dados: O MQTT usa cabeçalhos de apenas 2 bytes, o que é essencial para economizar o plano de dados do chip M2M/GPRS.

Reconexão Automática: A função reconectar MQTT() garante que o sistema volte a operar após túneis ou áreas de sombra de sinal celular.

- Camada de Processamento:

  (Cloud/Backend): Utilizando Servidor em Node.js ou Python que consome os dados e calcula a Estimativa de Chegada (ETA) cruzando dados de GPS com APIs de trânsito.

```
const mqtt = require('mqtt');
const axios = require('axios');

// --- CONFIGURAÇÕES ---
const G_MAPS_API_KEY = 'SUA_CHAVE_GOOGLE_MAPS';
const DESTINO_FINAL = '-23.5596,-46.6581'; // Latitude,Longitude do Destino
const MQTT_BROKER = 'mqtt://://hivemq.com';
const TOPICO = 'rastreador/esp32/status';

const client = mqtt.connect(MQTT_BROKER);

client.on('connect', () => {
    console.log('Conectado ao Broker MQTT');
    client.subscribe(TOPICO);
});

client.on('message', async (topic, message) => {
    try {
        const dados = JSON.parse(message.toString());
```

```
        const { lat, lng, status } = dados;

        console.log(`Recebido - Lat: ${lat}, Lng: ${lng}, Status: ${status}`);

        if (status === 'EM_TRANSITO') {
            const eta = await calcularETA(lat, lng);
            console.log(`>>> Estimativa de Chegada: ${eta.durationText} (${eta.distanceText})`);

            // Aqui podemos salvar no Banco de Dados ou enviar para um Dashboard
        }
    } catch (error) {
        console.error('Erro ao processar mensagem:', error.message);
    }
});

// função calcularETA para evitar erros de leitura
async function calcularETA(lat, lng) {
    const url = `https://googleapis.com{lat},${lng}&destinations=${DESTINO_FINAL}&departure_time=now&traffic_model=best_guess&key=${G_MAPS_API_KEY}`;

    try {
        const response = await axios.get(url);
        // Acesso ao array do Google Maps
        const element = response.data.rows[0].elements[0];

        if (element.status === 'OK') {
            return {
                durationText: element.duration_in_traffic ? element.duration_in_traffic.text : element.duration.text,
                distanceText: element.distance.text
            };
        }
    } catch (err) {
        console.error('Erro na conexão com a API');
```

```
    }
    return { durationText: 'Indisponível', distanceText: 'N/A' };
}
```

Como funciona o fluxo completo:

O ESP32 envia as coordenadas via MQTT (Payload JSON).

O seu servidor Node.js (que pode estar em uma VPS ou local) recebe o tópico.

O servidor extrai a localização e pergunta à API do Google: "Quanto tempo leva daqui até o destino agora?".

O resultado (ETA) é exibido ou armazenado.

3 DESENVOLVIMENTO

1. Garagem: O dispositivo realiza o “ Check-out” automático:

Utilizando o ESP32 se conecta ao Wi-Fi da garagem(mas estável) realizando um “handshake” com o servidor.

O sistema utiliza Geofencing(cerca virtual) Onde as coordenadas de GPS saem do polígono desenhado ao redor da garagem, assim o estado do veículo no banco de dados muda de ESTACIONADO para EM_ DESLOCAMENTO_ TECNICO.

Tendo como dados coletados, horário de saída, nível de bateria do sistema e diagnóstico inicial do veículo.

2. Percurso: O algoritmo valida se o veículo mantém a velocidade média e a rota planejada.

O dispositivo irá alternar para a rede GPRS/4G, para que haja a economia de dados irá utilizar o protocolo MQTT, enviando pacotes leves (JSON) com lat, long, speed e times tamp.

Se houver uma “área de sombra” (sem sinal), o código deve implementar uma Fila de Mensagens (Buffer) na merória RAM ou cartão SD do ESP32, assim quando o sinal voltar os dados acumulado serão enviados com os horários retroativos.

O backend irá processar as coordenadas e estimar o tempo de chegada ao destino inicial, comparando com a tabela de horários definidas.

3. Destino: O “Ckeck-in” encerra o log de deslocamento técnico e liberar o veículo para o sistema de bilhetagem/operação.

Ao entrar no raio de alcance (Geofence) do ponto de destino final, o sistema irá reconhecer a conclusão do trajeto técnico.

O status muda para EM_OPERACAO_COMERCIAL, sendo o log do “KM MORTO” encerrado e calculando o tempo total gasto e a distância percorrida.

Esses dados são consolidados em Dashboard para o CCO/Gestor, mostrando se aquele veículo cumpriu o deslocamento técnico dentro da meta de tempo e distância estipulada

4 CONCLUSÃO

- Redução da incerteza operacional no terminal de saída.
- Histórico preciso de tempo e consumo no trajeto técnico.
- Alertas automáticos de atraso para os CCO/Gestor.

AGRADECIMENTOS

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida e me dar forças sempre na minha caminhada.

A minha esposa Priscila, que sempre me deu toda força e apoio para chegar aqui. A ela dedico a muita eterna gratidão.

Aos meus pais Ivete e José (in memoriam) por sempre acreditar em mim e pelo esforço para minha criação.

Aos meus filhos Pamela e Pablo, por serem o combustível diário para a busca de minha evolução, como pessoa e como profissional.

Aos meus sogros Jaidete e Nelson, pelo zelo e apoio constante em minha vida.

Ao Professor Msc. Caique Zaneti Kirilo pelo apoio e orientação dentro desta jornada de evolução acadêmica.

REFERÊNCIAS

ASHTON, Kevin. That "Internet of Things" thing. RFID Journal, [s. l.], v. 22, n. 7, p. 97-114, 2009.

ESPRESSIF SYSTEMS. ESP32 series datasheet. v. 3.6. [S. l.], 2023. Disponível em: espressif.com.

GOOGLE. Distance Matrix API documentation. [S. l.], 2024. Disponível em: google.com.

HARTKE, Mikal. TinyGPS++: a new, object-oriented GPS library for Arduino. 2023. Disponível em: github.com.

HILLIARD, Robert. MQTT essentials: a lightweight IoT protocol. [S. l.]: O'Reilly Media, 2018.

MAZIERO, Carlos A. Sistemas operacionais: conceitos e mecanismos. Curitiba: UFPR, 2019. Disponível em: ufpr.br.

OASIS. MQTT Version 3.1.1. OASIS Standard, 2014. Disponível em: oasis-open.org.

SHILDENSTOCK, Volodymyr. TinyGSM: a small Arduino library for GSM modules. 2023. Disponível em: github.com

U-BLOX. NEO-6: u-blox 6 GPS modules data sheet. [S. l.], 2011. Disponível em: u-blox.com.

KUROSE, James F.: ROSS, Keith W, Redes de Computadores e a Internet uma abordagem top-dow 7. ed .São Paulo.:Pearson 2017

HIVEMQ. MQTT Essentials: The Ultimate Guide to MQTT, 2015. Disponível em: hivemq.com.

FIGUEIREDO, L . et al. Towards the Development of Intelligent Transportation Systems, IEEE Intelligentet Transportation Systems, 2001.

SARKER, V.K. et al. Real- Time Endpoint Monitoring in IoT – Based Fleet Management Systems. IEEE Sensores Journal, 2019

POSTGIS PROJECT PostGIS Saptial Database Managemnt.

GUTTING, Ralf Hartmut. An Introduction to Spatial Database Systems. VLDB Journal, 1994.