CENTRO DE ESTUDOS E SISTEMAS AVANÇADOS DO RECIFE

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Solar Analytics: Protótipo de monitoramento Solar com IoT e MQTT

CRISTINA MATSUNAGA
CARLOS EDUARDO
GABRIEL CHAVES
MARIA FERNANDA
LUCAS GABRIEL

Recife, Dezembro de 2025

1. Introdução (Contexto, motivação, objetivos).

O crescimento da demanda por fontes de energia renovável tem impulsionado o desenvolvimento de tecnologias capazes de monitorar e analisar, em tempo real, o desempenho de sistemas de geração energética, em especial os sistemas fotovoltaicos. O acompanhamento contínuo de métricas como tensão, corrente e potência é fundamental para garantir eficiência, confiabilidade e tomada de decisão baseada em dados.

Paralelamente, a evolução da Internet das Coisas (Internet of Things – IoT) possibilitou a integração de dispositivos embarcados, sistemas de comunicação e plataformas de armazenamento e visualização de dados. Tecnologias como microcontroladores, protocolos de comunicação leve e bancos de dados orientados a séries temporais permitem a criação de soluções de monitoramento distribuídas, escaláveis e de baixo custo.

Nesse contexto, este projeto tem como objetivo desenvolver um sistema didático de monitoramento de métricas simuladas de uma placa de energia solar, utilizando uma plataforma embarcada ESP32, comunicação via protocolo MQTT sobre rede Wi-Fi, armazenamento em banco de dados InfluxDB e visualização de informações por meio de dashboards no Grafana. As métricas são simuladas por potenciômetros, representando grandezas variáveis presentes em sistemas fotovoltaicos reais.

Como objetivos específicos, destacam-se: (i) implementar a coleta de dados por meio de dispositivos embarcados; (ii) realizar a transmissão eficiente dos dados utilizando o protocolo MQTT; (iii) processar e armazenar as informações em um banco de dados adequado para séries temporais; e (iv) disponibilizar visualizações gráficas que permitam a análise das métricas coletadas. Dessa forma, o projeto busca integrar conceitos de sistemas embarcados, redes, bancos de dados e visualização de dados, proporcionando uma visão prática e aplicada dos conteúdos abordados ao longo do curso.

2. Metodologia

A metodologia adotada neste projeto baseia-se na implementação de um sistema de monitoramento distribuído, utilizando conceitos de Internet das Coisas (IoT), comunicação assíncrona e armazenamento de dados em séries temporais. O sistema foi desenvolvido de forma modular, separando as responsabilidades de coleta, transmissão, processamento, armazenamento e visualização dos dados.

2.1. Diagrama do sistema.

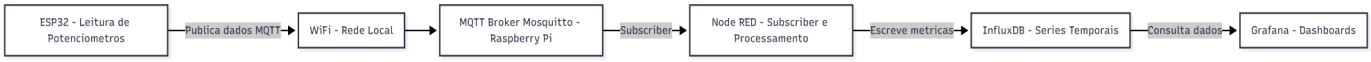
O sistema desenvolvido é organizado em três camadas principais: aquisição de dados, comunicação e processamento/visualização. Na camada de aquisição, uma placa ESP32 é responsável pela leitura de quatro potenciômetros que simulam métricas

provenientes de um sistema de geração de energia solar. Esses valores representam grandezas variáveis comumente analisadas em sistemas fotovoltaicos.

Na camada de comunicação, os dados coletados são transmitidos por meio de uma rede Wi-Fi, utilizando o protocolo MQTT. A Raspberry Pi atua como broker MQTT, sendo responsável por gerenciar os tópicos e distribuir as mensagens publicadas pelos dispositivos conectados.

Por fim, na camada de processamento e visualização, o Node-RED é utilizado como subscriber dos tópicos MQTT. Ele realiza o processamento, tratamento e encaminhamento dos dados para o banco de dados InfluxDB, que serve como base para a geração de dashboards no Grafana.

(O diagrama do sistema ilustra a comunicação entre a ESP32, o broker MQTT, o Node-RED, o InfluxDB e o Grafana.)



2.2. Lista de hardware/software.

Os componentes de hardware empregados no projeto são:

- Placa microcontrolada ESP32;
- Quatro potenciômetros para simulação das métricas da placa solar;
- Raspberry Pi configurada como broker MQTT;
- Computador para execução do Node-RED, InfluxDB e Grafana;
- Infraestrutura de rede Wi-Fi para comunicação entre os dispositivos.

Em relação ao software, foram utilizadas as seguintes ferramentas e tecnologias:

- Ambiente de desenvolvimento para ESP32 (Arduino IDE ou equivalente);
- Broker MQTT instalado na Raspberry Pi;
- Node-RED para assinatura dos tópicos MQTT, processamento e encaminhamento dos dados;
- Banco de dados InfluxDB, adequado para armazenamento de dados em séries temporais;
- Grafana para criação de painéis e visualização gráfica das métricas coletadas.

2.3. Fluxo de comunicação (MQTT + Wi-Fi + Node-RED).

O fluxo de comunicação inicia-se na ESP32, que realiza a leitura periódica dos valores analógicos dos potenciômetros. Esses valores são convertidos em dados digitais e encapsulados em mensagens MQTT, as quais são publicadas em tópicos previamente definidos, utilizando a rede Wi-Fi como meio de transmissão.

A Raspberry Pi, atuando como broker MQTT, recebe as mensagens publicadas e as distribui aos clientes inscritos nos respectivos tópicos. O Node-RED funciona como subscriber desses tópicos, sendo responsável por receber os dados enviados pela ESP32.

No Node-RED, os dados passam por etapas de tratamento, incluindo formatação, identificação temporal e organização das métricas, de modo a torná-las compatíveis com o modelo de armazenamento do InfluxDB. Após o processamento, as informações são enviadas e armazenadas no banco de dados de séries temporais.

O Grafana acessa diretamente o InfluxDB para consultar os dados armazenados e gerar dashboards interativos, possibilitando a análise visual e em tempo real das métricas simuladas do sistema de energia solar.

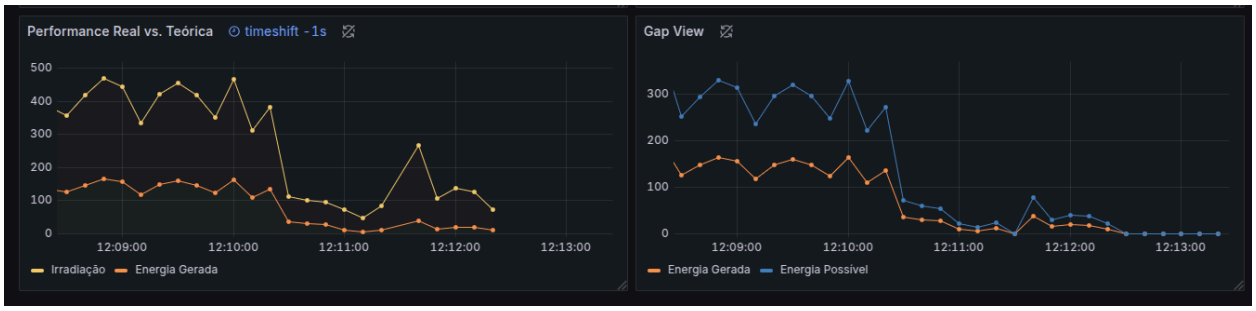
3. Resultados

Esta seção apresenta os resultados obtidos a partir da implementação e execução do sistema de monitoramento proposto, evidenciando o correto funcionamento da coleta, transmissão, processamento, armazenamento e visualização das métricas simuladas da placa de energia solar.

3.1. Dashboard de Visualização

Os dados coletados pela ESP32 e armazenados no banco de dados InfluxDB foram utilizados para a construção de dashboards no Grafana. Os painéis exibem gráficos em tempo real referentes às métricas simuladas pelos quatro potenciômetros, permitindo o acompanhamento contínuo do comportamento das variáveis.

Os dashboards desenvolvidos possibilitam a visualização de tendências, variações e estabilidade dos sinais ao longo do tempo, fornecendo uma representação clara e intuitiva das informações monitoradas. A Figura 2 e 3 apresentam o dashboard gerado pelo Grafana com os dados coletados durante a execução do sistema.



3.2. Dados coletados (gráficos/tabelas).

Os gráficos gerados demonstram a evolução temporal das variáveis monitoradas, evidenciando a capacidade do sistema em registrar alterações instantâneas e persistentes nos valores coletados.

5. Conclusão (Desafios, aprendizados, melhorias futuras).

A partir do desenvolvimento deste projeto, foi possível implementar com sucesso um sistema de monitoramento de métricas simuladas de uma placa de energia solar, integrando conceitos de sistemas embarcados, redes de comunicação, processamento de dados e visualização de informações. O sistema atendeu aos objetivos propostos, demonstrando a viabilidade do uso de tecnologias de Internet das Coisas (IoT) para coleta e análise de dados em tempo real.

Entre os principais desafios enfrentados, destacam-se a configuração da comunicação entre os dispositivos via protocolo MQTT, a integração entre o Node-RED e o banco de dados InfluxDB, bem como a padronização e tratamento dos dados recebidos antes do armazenamento. A superação desses desafios contribuiu para o aprofundamento do conhecimento prático sobre comunicação assíncrona, arquitetura orientada a eventos e bancos de dados voltados para séries temporais.

Como aprendizados, ressalta-se a importância da modularização do sistema, que facilitou o desenvolvimento, a depuração e futuras expansões da solução. A utilização do Node-RED mostrou-se eficaz para o processamento e encaminhamento dos dados, permitindo a rápida integração entre os diferentes componentes do sistema. Além disso, o uso do Grafana possibilitou a criação de dashboards claros e intuitivos, favorecendo a interpretação das métricas coletadas.

Por fim, como melhorias futuras, o sistema pode ser expandido para utilizar sensores reais em substituição aos potenciômetros, possibilitando a aplicação em cenários reais de monitoramento fotovoltaico. Outras evoluções incluem a implementação de mecanismos de segurança na comunicação MQTT, o armazenamento de um maior volume de dados para análises históricas mais aprofundadas e a inclusão de alertas automáticos para detecção de comportamentos anômalos. Dessa forma, o projeto demonstra potencial de aplicação prática e serve como base para estudos e desenvolvimentos futuros na área de IoT e energia renovável.

6. Apêndice

6.1 Códigos completos (ESP32).

O código completo pode ser acessado através do nosso github

```
#include <WiFi.h>
#include <PubSubClient.h>

// ======== Wi-Fi ========
const char* ssid     = "outra_dimensao";
const char* password = "kika1234";

// ======== MQTT (broker de teste público) ========
const char* mqtt_server = "10.213.91.245";   // depois você trocar
pelo IP da rasp
const int   mqtt_port   = 1883;
const char* mqtt_topic  = "solar/metrics"; // só escolher " qualquer
" nome

WiFiClient espClient;
PubSubClient mqttClient(espClient);

// ======== Pinos dos potenciômetros ========
const int PIN_IRR   = 34; // Irradiação (W/m²)
const int PIN_EFF   = 35; // Eficiência (%)
const int PIN_AREA  = 32; // Área (m²)
```

```
const int PIN_COST  = 33; // Custo (R$)

// Faixas para mapeamento
const float IRR_MIN      = 0.0;
const float IRR_MAX      = 1000.0;   // W/m²

const float EFF_MIN_PCT  = 10.0;
const float EFF_MAX_PCT  = 25.0;     // %

const float AREA_MIN     = 0.5;
const float AREA_MAX     = 2.0;      // m²

const float COST_MIN     = 500.0;
const float COST_MAX     = 5000.0;   // R$

const int SMOOTH_N = 5; // média simples

// ======== Funções auxiliares ========
float mapf(float x, float in_min, float in_max, float out_min, float
out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
out_min;
}

int readADC(int pin) {
  long acc = 0;
  for (int i = 0; i < SMOOTH_N; i++) {
    acc += analogRead(pin);
    delay(2);
  }
  return (int)(acc / SMOOTH_N);
}

void connectWifi() {
  Serial.println();
  Serial.print("Conectando ao Wi-Fi: ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
```

```
  WiFi.begin(ssid, password);

  int tentativas = 0;
  const int TENTATIVAS_MAX = 20;

  while (WiFi.status() != WL_CONNECTED && tentativas <
TENTATIVAS_MAX) {
    delay(500);
    Serial.print(".");
    tentativas++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\n✅ Conectado ao Wi-Fi!");
    Serial.print("IP da ESP32: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("\n❌ Falha ao conectar no Wi-Fi.");
    Serial.println("Verifique SSID, senha e tipo de rede (2.4
GHz).");
  }
}

void connectMQTT() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("⚠ Wi-Fi não conectado, não dá pra conectar no
MQTT.");
    return;
  }

  while (!mqttClient.connected()) {
    Serial.print("Conectando ao broker MQTT... ");
    // clientId aleatório pra evitar conflito
    String clientId = "ESP32Solar-" + String(random(0xffff), HEX);

    if (mqttClient.connect(clientId.c_str())) {
      Serial.println("conectado!");
      // se precisasse assinar tópico, seria aqui
(mqttClient.subscribe(...))
```

```
    } else {
      Serial.print("falha, rc=");
      Serial.print(mqttClient.state());
      Serial.println(" tentando de novo em 2s...");
      delay(2000);
    }
  }
}

// ======== Setup ========
void setup() {
  Serial.begin(115200);
  delay(1000);

  // Configura ADC
  analogSetAttenuation(ADC_11db);
  pinMode(PIN_IRR,  INPUT);
  pinMode(PIN_EFF,  INPUT);
  pinMode(PIN_AREA, INPUT);
  pinMode(PIN_COST, INPUT);

  // Wi-Fi
  connectWifi();

  // MQTT
  mqttClient.setServer(mqtt_server, mqtt_port);
  randomSeed(micros());  // pra gerar clientId aleatório

  if (WiFi.status() == WL_CONNECTED) {
    connectMQTT();
  }

  Serial.println("\n{\"status\":\"boot\",\"msg\":\"ESP32 Solar Sim
iniciado\"}");
}

// ======== Loop principal ========
void loop() {
  // mantém conexão MQTT viva
```

```
  if (WiFi.status() == WL_CONNECTED && !mqttClient.connected()) {
    connectMQTT();
  }
  mqttClient.loop();

  // Leituras brutas
  int rawIrr  = readADC(PIN_IRR);
  int rawEff  = readADC(PIN_EFF);
  int rawArea = readADC(PIN_AREA);
  int rawCost = readADC(PIN_COST);

  // Conversão para faixas "reais"
  float irrWm2  = mapf(rawIrr,  0.0, 4095.0, IRR_MIN, IRR_MAX);
  float effPct  = mapf(rawEff,  0.0, 4095.0, EFF_MIN_PCT,
EFF_MAX_PCT);
  float areaM2  = mapf(rawArea, 0.0, 4095.0, AREA_MIN, AREA_MAX);
  float costBRL = mapf(rawCost, 0.0, 4095.0, COST_MIN, COST_MAX);

  // Potência estimada: P = Irradiação * Área * Eficiência
  float effDec = effPct / 100.0;
  float powerW = irrWm2 * areaM2 * effDec;

  // ===== Saída organizada no Serial =====
  Serial.println("\n================== MÉTRICAS DO PAINEL SOLAR
==================");
  Serial.print("Irradiação:       "); Serial.print(irrWm2, 1);
Serial.println(" W/m²");
  Serial.print("Eficiência:       "); Serial.print(effPct, 1);
Serial.println(" %");
  Serial.print("Área do Painel:   "); Serial.print(areaM2, 2);
Serial.println(" m²");
  Serial.print("Custo (estimado): R$ "); Serial.print(costBRL, 2);
Serial.println();
  Serial.print("Potência Gerada:  "); Serial.print(powerW, 1);
Serial.println(" W");

Serial.println("----------------------------------------------------
---------");
```

```
  // ===== Envio via MQTT em JSON =====
  if (mqttClient.connected()) {
    char payload[256];
    snprintf(
      payload,
      sizeof(payload),
      "{\"irr_wm2\":%.1f,\"eff_pct\":%.1f,\"area_m2\":%.2f,"
      "\"cost_brl\":%.2f,\"power_w\":%.1f}",
      irrWm2, effPct, areaM2, costBRL, powerW
    );

    bool ok = mqttClient.publish(mqtt_topic, payload);
    Serial.print("MQTT publish: ");
    Serial.println(ok ? "OK" : "ERRO");
  } else {
    Serial.println("⚠ Não conectado ao MQTT, não foi possível
publicar.");
  }

  delay(2000); // atualiza a cada 2s
}
```

Link do github com toda a documentação técnica:
https://github.com/LucasGdBS/SolarAnalytics/tree/main