UNIVERSIDADE PRESBITERIANA MACKENZIE
– Faculdade de Computação e Informática –

Nome do projeto: Gerenciamento de Energia em Fábricas Usando IoT

Beatriz Palombarini de Souza[1]

Professores: Leandro Carlos Fernandes e Andre Luis de Oliveira[1]

[1] Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie (UPM) – São Paulo, SP – Brazil

10340701@mackenzista.com.br

Abstract. *This paper presents the development of an IoT-based system for real-time energy consumption monitoring and control in industrial environments. The system leverages the Arduino Nano microcontroller, ESP8266 Wi-Fi module, and MQTT protocol for seamless communication with the cloud platform io.arkfruit. By integrating energy consumption simulations and automated control mechanisms, the project aims to optimize energy usage, reduce operational costs, and prevent energy waste during peak demands. Additionally, it provides a user-friendly interface for remote monitoring and decision-making, aligning with the United Nations Sustainable Development Goal (SDG) 9, which promotes sustainable industrialization and innovation. This scalable and efficient solution supports industries in transitioning toward more sustainable and energy-efficient practices.*

Keywords: *IoT, Energy Management, Industry 4.0, Sustainability, MQTT, Cloud Integration, Arduino Nano.*

Resumo. *Este artigo apresenta o desenvolvimento de um sistema baseado em IoT para o monitoramento e controle em tempo real do consumo de energia em ambientes industriais. O sistema utiliza o microcontrolador Arduino Nano, o módulo Wi-Fi ESP8266 e o protocolo MQTT para comunicação eficiente com a plataforma em nuvem io.arkfruit. Ao integrar simulações de consumo energético e mecanismos de controle automatizados, o projeto busca otimizar o uso de energia, reduzir custos operacionais e evitar desperdícios durante picos de consumo. Além disso, o sistema oferece uma interface intuitiva para monitoramento remoto e suporte à tomada de decisões, alinhando-se ao Objetivo de Desenvolvimento Sustentável 9 (ODS 9) das Nações Unidas, que promove a industrialização sustentável e a inovação. Esta solução escalável e eficiente apoia as indústrias na transição para práticas mais sustentáveis e energeticamente eficientes.*

Palavras-chave: *IoT, Gerenciamento de Energia, Indústria 4.0, Sustentabilidade, MQTT, Integração em Nuvem, Arduino Nano.*

1. Introdução

A gestão eficiente da energia é um fator determinante para a sustentabilidade e competitividade das indústrias modernas (Ayaz et al., 2019). Com a ascensão da Indústria 4.0, tecnologias emergentes como a Internet das Coisas (IoT) têm transformado a maneira como recursos são monitorados e otimizados, desempenhando um papel essencial na redução de desperdícios e no aumento da eficiência operacional (Prakash & Nazir, 2019; Hassan et al., 2018). A IoT, combinada com protocolos leves como o MQTT, oferece uma comunicação rápida e confiável entre dispositivos, viabilizando aplicações de gerenciamento energético em tempo real (Thomas & Bennet, 2021).

Além disso, o Objetivo de Desenvolvimento Sustentável 9 (ODS 9) enfatiza a necessidade de promover a industrialização sustentável, fomentar a inovação e reduzir impactos ambientais por meio da eficiência energética (ONU, 2023). Sistemas inteligentes de gerenciamento de energia permitem não apenas o controle local, mas também o monitoramento remoto, proporcionando decisões rápidas e ações preventivas baseadas em dados (Sambaivan & De Sanctis, 2021).

Este projeto propõe o desenvolvimento de um sistema IoT para monitoramento e controle de consumo energético em ambientes industriais. Utilizando o microcontrolador Arduino Nano e o módulo ESP8266, o sistema simula sensores de corrente, transmite dados para uma plataforma de nuvem via protocolo MQTT e oferece uma interface gráfica na plataforma **io.arkfruit** para monitoramento em tempo real. Além disso, o sistema é projetado para desativar automaticamente dispositivos em casos de picos de consumo, contribuindo para a segurança e eficiência operacional (Kumar & Singh, 2020; Lee & Lee, 2020).

Com uma abordagem escalável e prática, o sistema é capaz de:

- **Integrar** sensores de corrente simulados com a plataforma Arduino Nano para coleta de dados.
- **Transmitir** dados para a nuvem utilizando o ESP8266 e o protocolo MQTT, que se destaca pela leveza e eficiência (Ali & Zahra, 2020).
- **Desenvolver** uma interface de controle e monitoramento remoto na plataforma **io.arkfruit**, facilitando a análise de padrões de consumo e intervenções em tempo real.
- **Analisar** dados coletados para prever demandas futuras e identificar oportunidades de otimização energética.

Ao alinhar-se com as demandas da Indústria 4.0 e os princípios do ODS 9, este projeto demonstra como soluções acessíveis e tecnologias emergentes podem transformar o gerenciamento energético em fábricas, promovendo sustentabilidade e inovação (Chen & Li, 2020; Fernandes & Costa, 2021).

Objetivos:

Geral:
Desenvolver um sistema IoT integrado para monitoramento, controle e otimização do consumo de energia em ambientes industriais, promovendo eficiência operacional e redução de desperdícios.

Específicos:

- **Desenvolver** um sistema de monitoramento energético utilizando o microcontrolador Arduino Nano e o módulo ESP8266 para comunicação com a nuvem.
- **Configurar** o protocolo MQTT para a transmissão de dados em tempo real entre os sensores, o servidor na nuvem e a interface de monitoramento remoto.
- **Criar** uma interface interativa na plataforma **io.arkfruit**, permitindo a visualização dos dados de consumo energético, identificação de padrões e controle remoto das cargas.
- **Implementar** um sistema de controle automatizado que desative dispositivos em situações de picos de consumo para evitar sobrecargas e desperdícios.
- **Realizar** a análise de dados coletados para identificar oportunidades de otimização energética e gerar relatórios que apoiem a tomada de decisões estratégicas.
- **Promover** escalabilidade no sistema, permitindo futuras integrações com sensores adicionais e melhorias nas funcionalidades de controle e monitoramento.

2. Materiais e métodos

Arduino Nano e Módulo ESP8266

O projeto utiliza o Arduino Nano como unidade de controle principal, combinado com o módulo ESP8266 para comunicação Wi-Fi, permitindo futuras integrações IoT e monitoramento remoto.

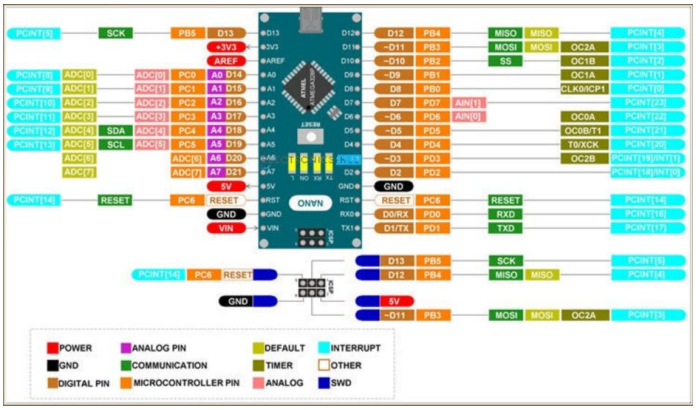

Figura 1: Arduino Nano Datasheet (https://www.nitrathor.com/data-sheets/arduino-nano)

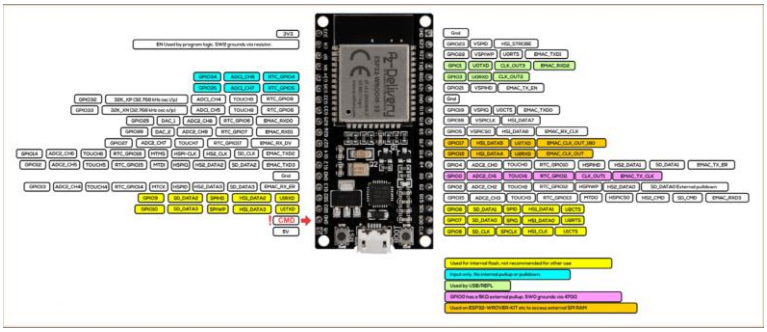

Figura 2: ESP32 Datasheet (https://www.nitrathor.com/data-sheets/esp32-nodemcu)

Componentes principais:

- **Arduino Nano**:
  - Microcontrolador ATmega328P.
  - Pinos de entrada/saída digitais e analógicos para controle de dispositivos.
  - Programável via USB.
  - Tensão de operação: 5V.
- **ESP8266**:
  - Wi-Fi integrado 802.11 b/g/n.
  - Interface UART para comunicação serial.
  - Capaz de transmitir dados em tempo real.

Funcionamento:

1. **Programação**: O Arduino Nano é programado na IDE Arduino com bibliotecas específicas para controle do relé e leitura do potenciômetro.
2. **Controle e Comunicação**: O microcontrolador processa dados do potenciômetro, simulando consumo energético, e aciona o relé. O módulo ESP8266 é configurado para conexão com servidores MQTT.

(Referências: ARDUINO, 2023; EXPRESSIF, 2023).

Protoboard e Fios Jumper

A protoboard foi utilizada como base para a montagem do circuito, permitindo conexões rápidas e modificações no sistema sem necessidade de soldagem.

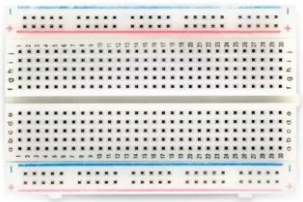

Figura 3: Protoboard (https://produto.mercadolivre.com.br/MLB-811241019-protoboard-400-pontos-_JM)

Figura 4: Fios Jumper (https://www.robobuilders.com.br/buscar?q=40+X+Jumper+Fio+Conector)

Características:

- **Protoboard 400 pontos**:
  - Barras de distribuição para alimentação e terra.
  - Facilita conexões modulares.
- **Fios Jumper**:
  - Diversos comprimentos e cores para conexões elétricas.
  - Isolados com material plástico, garantindo segurança no circuito.

Vantagens:

- Montagem rápida e reutilizável.
- Modificações facilitadas no circuito.

(Referência: CAP SISTEMA, 2021).

Potenciômetro e Relé SPDT

O potenciômetro substitui o sensor ACS712 para simular o consumo de energia no protótipo atual. O relé SPDT controla a carga representada pela lâmpada LED.

Figura 5: Potenciômetro (https://www.zamux.co/potenciometro-10k)

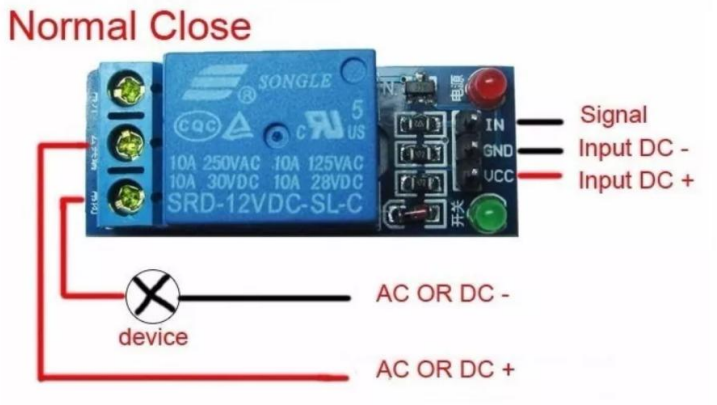

Figura 6: Relé 1 Canal de Sinal 5V (https://produto.mercadolivre.com.br/MLB-1343987636-modulo-rele-1-canal-de-sinal-5v-para-110220v-ac-arduino-_JM)

Componentes principais:

- **Potenciômetro**:
  - Resistência variável para simular diferentes níveis de consumo.
- **Relé SPDT**:
  - Capacidade de comutação: até 10A.
  - Controlado pelo Arduino Nano para ligar/desligar a lâmpada LED.

Funcionamento:

1. **Simulação de consumo**: O potenciômetro ajusta manualmente os níveis de corrente simulada.
2. **Controle de carga**: O Arduino Nano processa os valores do potenciômetro e aciona o relé, controlando a lâmpada LED.

(Referências: EXPRESSIF, 2023; ARDUINO, 2023).

LED e Resistor de 220Ω

A lâmpada LED representa uma carga industrial que é controlada pelo sistema. Um resistor de 220Ω foi incluído para proteger o circuito e limitar a corrente elétrica.

Figura 7: Led (https://www.amazon.com.br/ENLOZURE-LED-ilumina%C3%A7%C3%A3o-transparente-vermelho/dp/B0CGX8ZNJC)

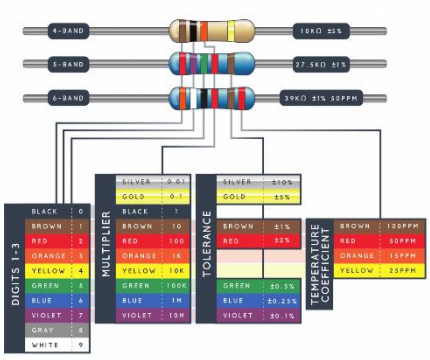

Figura 8: Resistores (https://www.robocore.net/tutoriais/introducao-ao-resistor)

Características:

- **LED**:
    - Representa o consumo energético simulado.
    - Controlada pelo relé.
- **Resistor de 220Ω**:
    - Protege o circuito ao limitar a corrente nos componentes.

(Referências: CAP SISTEMA, 2021).

Fonte de Alimentação 9V

O sistema é alimentado por uma fonte externa de 9V, garantindo estabilidade de operação.

Figura 9: Bateria 9v e conector (https://www.dedcomponentes.com.br/produto/bateria-9v-rontek-ultra-power/)

Características:

- **Fonte de 9V**:
    - Alimenta o Arduino Nano e demais componentes.
    - Compatível com a protoboard para facilitar a distribuição de energia.

(Referência: CAP SISTEMA, 2021).

Comunicação MQTT via ESP32

O MQTT (Message Queuing Telemetry Transport) foi configurado no módulo ESP8266 para realizar a comunicação com a plataforma **io.arkfruit**. Este protocolo, leve e eficiente, é ideal para aplicações IoT.

Características:

- **MQTT**:
    - Protocolo leve e eficiente para transmissão de dados.
    - Permite monitoramento remoto e ações corretivas.
- **Plataforma io.arkfruit**:

- Serviço em nuvem para IoT, integrando dashboards para visualização dos dados enviados.

(Referências: EXPRESSIF, 2023; IO.ARKFRUIT, 2023).

Simulação e Modelo no Fritzing

O modelo foi desenvolvido no software **Fritzing**, permitindo a visualização do circuito e organização dos componentes. O layout incluiu:

- **Potenciômetro** conectado à porta analógica A7 para simular corrente.
- **Relé SPDT** controlando uma lâmpada LED como carga.
- **ESP32** responsável pela comunicação MQTT.

(Referência: FRITZING, 2023).

Código Utilizado no Modelo Simulado (Modelo 1):

O código a seguir foi implementado para controlar a lâmpada simulada, ajustando sua intensidade com base no valor de corrente simulado pelo potenciômetro:

```
int cor = A0;        // Porta A0 para ler a corrente
int tomada1 = 9;     // Porta PWM (pino 9) para controlar a lâmpada
float I = 0;         // Variável para leitura do valor analógico
float corrente = 0.0;// Variável para armazenar a corrente convertida (em mA)
int valorPWM = 0;    // Variável para armazenar o valor de saída PWM

void setup() {
   Serial.begin(9600);          // Inicializa a comunicação serial
   pinMode(tomada1, OUTPUT);    // Define a porta como saída
}

void loop() {
  I = analogRead(cor);  // Lê o valor do sensor de corrente na porta A0

 // Calcula a corrente em mA
  corrente = (I * 5.0) / 1024.0;
  Serial.print("Corrente: ");
  Serial.print(corrente);
```

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


// Verifica se a corrente está acima de 4.0 mA
  if (corrente > 4.0) {
    valorPWM = 10; // Define um valor PWM intermediário para reduzir a intensidade da lâmpada
    Serial.println("DIMINUIR");  // Informação sobre ajuste
  } else {
    valorPWM = 1000; // Define o valor máximo de PWM (lâmpada em potência máxima)
    Serial.println("AUMENTAR");  // Informação sobre ajuste
  }


analogWrite(tomada1, valorPWM);  // Aplica o valor PWM na porta
  delay(1000);  // Espera 1 segundo antes da próxima leitura
}
```

Código Utilizado no Modelo Simulado (Modelo 2):

O código a seguir foi programado no Arduino Nano para monitorar os valores de corrente simulada lidos pelo potenciômetro e controlar o estado de uma lâmpada LED conectada ao relé. O sistema desliga ou liga a carga de acordo com os valores simulados, representando o comportamento de controle de consumo energético.

```
int cor = A7;        // Porta A0 para ler a corrente
int tomada1 = 2;
float I = 0;         // Variável para leitura do valor analógico
float corrente = 0.0;// Variável para armazenar a corrente convertida (em mA)


void setup() {
   Serial.begin(9600);          // Inicializa a comunicação serial
   pinMode(tomada1, OUTPUT);     // Define a porta como saída
}


void loop() {
  I = analogRead(cor);  // Lê o valor do sensor de corrente na porta A0
```

```
  // Calcula a corrente em mA
  corrente = (I * 5.0) / 1024.0;
  Serial.print("Corrente: ");
  Serial.print(corrente);
  Serial.println(" mA");
  // Verifica se a corrente está acima de 4.0 mA
  if (corrente > 4.8) {
    digitalWrite(tomada1, HIGH);
  } else {
    digitalWrite(tomada1, LOW);
  }
}
```

Funcionamento do Novo Código:

1. **Leitura da Corrente Simulada**:
    - O potenciômetro ajusta manualmente o valor de corrente simulada.
    - A leitura do potenciômetro na porta A7 é convertida para um valor de corrente em mA, utilizando uma equação proporcional.
2. **Controle Automático do Relé**:
    - Quando o valor da corrente simulada fica abaixo de 4.8 mA, o Arduino ativa o relé (pino 2), ligando o led.
    - Caso o valor fique acima ou igual a 4.8 mA, o relé é desligado automaticamente, desligando o led.
3. **Feedback Serial**:
    - O sistema envia mensagens via comunicação serial indicando os valores de corrente e o estado da carga (ligada ou desligada), permitindo monitoramento e debug em tempo real.

Código Utilizado no Modelo Simulado (Modelo 3):

Arduino

Este código lê a corrente elétrica de um sensor conectado à porta analógica A7 do Arduino e utiliza um relé para controlar o estado de uma tomada com base nos valores medidos. Os dados de corrente e estado do sistema são enviados via comunicação serial para outro dispositivo (por exemplo, um ESP32).

```
int cor = A7;        // Porta A0 para ler a corrente
int tomada1 = 2;
float I = 0;         // Variável para leitura do valor analógico
```

```
float corrente = 0.0;// Variável para armazenar a corrente convertida (em mA)
int valorPWM = 0;    // Variável para armazenar o valor de saída PWM
bool isCasaLigada = false;

void setup() {
   Serial.begin(9600);          // Inicializa a comunicação serial
   pinMode(tomada1, OUTPUT);     // Define a porta como saída
}

void loop() {
  I = analogRead(cor);  // Lê o valor do sensor de corrente na porta A0

  // Calcula a corrente em mA
  corrente = (I * 5.0) / 1024.0;

  // Serial.print("Corrente: ");
  // Serial.print(corrente);
  // Serial.println(" mA");

  // Verifica se a corrente está acima de 4.0 mA
  if (corrente > 4.8) {
    digitalWrite(tomada1, HIGH); //Relé fecha circuito
    isCasaLigada = false;
  } else {
    digitalWrite(tomada1, LOW); // Relé abre circuito
    isCasaLigada = true;
  }

   // Envia as variáveis no formato "numero,estado\n"
    Serial.print(corrente);
    Serial.print(",");
    Serial.println(isCasaLigada); // Converte o booleano para "1" ou "0"

}
```

Descrição dos Componentes:

- cor **(A7)**: Porta analógica usada para ler o sinal do sensor de corrente.
- tomada1 **(2)**: Porta digital conectada ao relé que controla o circuito da tomada.
- I **e** corrente: Variáveis para armazenar o valor lido do sensor e a corrente convertida (em mA), respectivamente.
- isCasaLigada: Indicador booleano para o estado do relé (ligado/desligado).

Funcionamento:

1. **Inicialização (**setup**)**:
    - Configura a porta digital da tomada como saída.
    - Inicia a comunicação serial para envio de dados.
2. **Laço principal (**loop**)**:
    - Lê o valor analógico do sensor de corrente.
    - Converte o valor analógico para corrente elétrica em mA.
    - Verifica se a corrente está acima de 4.8 mA:
        - Se **sim**: Liga o relé (estado "circuito fechado").
        - Se **não**: Desliga o relé (estado "circuito aberto").
    - Envia os valores da corrente e do estado via comunicação serial no formato "corrente,estado\n".

ESP32

Este código recebe os dados do Arduino via comunicação serial, interpreta os valores de corrente e estado, e publica essas informações no Adafruit IO usando feeds configurados previamente.

```
#include <WiFi.h>                // Biblioteca para conexão Wi-Fi
#include "AdafruitIO_WiFi.h"     // Biblioteca do Adafruit IO para ESP32

// Configurações do Adafruit IO
#define IO_USERNAME  "" // Substitua pelo seu usuário Adafruit IO
#define IO_KEY       ""         // Substitua pela sua chave do Adafruit IO

// Configurações de Wi-Fi
#define WIFI_SSID    ""                  // Substitua pelo nome da sua rede Wi-Fi
#define WIFI_PASS    ""                  // Substitua pela senha da sua rede Wi-Fi
```

```
// Inicializa o Adafruit IO WiFi
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

// Configurações de feeds no Adafruit IO
AdafruitIO_Feed *correnteFeed = io.feed("corrente");  // Feed para enviar os valores da corrente
AdafruitIO_Feed *statusFeed = io.feed("EnergiaFabrica");      // Feed para enviar o status (Ligado/Desligado)

String numero ;         // Variável numérica recebida
String estado ;    // Variável booleana recebida
String dados = "";      // Buffer para armazenar dados recebidos

void setup() {
   // Configura a comunicação serial
   Serial.begin(9600);
   Serial.println("Iniciando...");

   // Conecta ao Adafruit IO
   Serial.print("Conectando ao Adafruit IO...");
   io.connect();

   // Aguarda a conexão
   while (io.status() < AIO_CONNECTED) {
      Serial.print(".");
      delay(500);
   }

   // Mensagem de sucesso
   Serial.println("\nConectado ao Adafruit IO!");
   Serial.print("Endereço IP: ");
   Serial.println(WiFi.localIP());
}
```

```
unsigned long ultimaPublicacao = 0; // Armazena o tempo da última publicação
const unsigned long intervaloPublicacao = 6000; // 10 segundos (em milissegundos)

void loop() {
  // Mantém a conexão com o Adafruit IO
  io.run();

  // Verifica se há dados disponíveis
  while (Serial.available() > 0) {
    char caractere = Serial.read(); // Lê o próximo caractere
    if (caractere == '\n') { // Fim da linha (dados completos recebidos)
      // Divide os dados usando a vírgula como delimitador
      int separador = dados.indexOf(',');
      if (separador != -1) {
        numero = dados.substring(0, separador); // Primeiro valor (numérico)
        estado = (dados.substring(separador + 1)); // Segundo valor (booleano)

        unsigned long tempoAtual = millis(); // Obtém o tempo atual
        if (tempoAtual - ultimaPublicacao >= intervaloPublicacao) {
          ultimaPublicacao = tempoAtual; // Atualiza o tempo da última publicação

          // Publica os dados nos feeds
          correnteFeed->save(numero);
          statusFeed->save(estado);

          // Exibe os valores recebidos no Monitor Serial
          Serial.println("Dados enviados para Adafruit IO:");
          Serial.print("Corrente: ");
          Serial.println(numero);
          Serial.print("Estado: ");
          // Serial.println(estado == "true" ? 1 : 0);
          Serial.println(estado);
          Serial.println();
        }
```

```
            }
            dados = ""; // Limpa o buffer
        } else {
            dados += caractere; // Concatena o caractere no buffer
        }
    }
}
```

Descrição dos Componentes:

- WiFi **e** AdafruitIO_WiFi: Bibliotecas para conexão com a rede Wi-Fi e integração com a plataforma Adafruit IO.
- correnteFeed **e** statusFeed: Feeds configurados no Adafruit IO para receber os valores de corrente e estado.
- **Variáveis de Configuração**:
    - IO_USERNAME e IO_KEY: Credenciais da conta Adafruit IO.
    - WIFI_SSID e WIFI_PASS: Nome e senha da rede Wi-Fi.

Funcionamento:

1. **Inicialização (**setup**)**:
    - Configura a comunicação serial e conecta ao Wi-Fi e ao Adafruit IO.
    - Exibe mensagens no monitor serial para indicar o progresso da conexão.
2. **Laço principal (**loop**)**:
    - Mantém a conexão com o Adafruit IO.
    - Verifica se há dados recebidos pela comunicação serial:
        - Os dados são lidos e armazenados em um buffer.
        - Quando a linha completa é recebida (\n), os valores de corrente e estado são extraídos.
    - Publica os valores de corrente e estado no Adafruit IO a cada 6 segundos.
    - Exibe os dados enviados no monitor serial.

Fluxo Geral do Sistema:

1. O Arduino lê a corrente elétrica, controla o relé da tomada e envia os dados de corrente e estado via comunicação serial.
2. O ESP32 recebe os dados do Arduino, interpreta as informações e publica no Adafruit IO, permitindo o monitoramento remoto dos valores.

Observação:

- Substitua as credenciais Wi-Fi e Adafruit IO pelas suas informações reais antes de carregar os códigos.
- Certifique-se de que o sensor de corrente e o relé estejam corretamente conectados para evitar danos aos dispositivos.

Atualizações no Projeto:

- O controle da carga foi simplificado com o uso do comando digitalWrite, eliminando a necessidade de PWM para esta versão.
- O pino utilizado para leitura do potenciômetro foi alterado para A7, garantindo maior compatibilidade com a configuração física do circuito.

Este novo código mantém a flexibilidade para ajustes futuros, como integração com comunicação MQTT para monitoramento remoto.

Funcionamento:

O protótipo físico opera como um sistema integrado de monitoramento e controle de consumo de energia para ambientes industriais. A seguir, o funcionamento detalhado:

1. **Sensor de Corrente Simulado (Potenciômetro):**
    - O potenciômetro simula o comportamento de um sensor de corrente, ajustando manualmente os valores de consumo energético no circuito.
    - O Arduino Nano lê os dados do potenciômetro pela porta analógica A7, convertendo-os para valores de corrente em miliampères (mA).
    - Caso a corrente simulada ultrapasse o limite predefinido (4.8 mA), o sistema aciona um mecanismo de controle para proteger os dispositivos.
2. **Controle Automático via Relé SPDT:**
    - O relé SPDT é acionado pelo pino 2 do Arduino Nano para controlar o estado de uma carga representada pela lâmpada LED.
    - Quando a corrente simulada ultrapassa 4.8 mA, o Arduino envia um sinal ao relé para desligar a carga, protegendo o sistema contra picos de consumo.
    - Se a corrente for menor ou igual a 4.8 mA, o relé liga novamente a carga, garantindo seu funcionamento normal.
3. **Comunicação MQTT via ESP32:**
    - O módulo ESP32 conecta-se à plataforma **io.arkfruit**, operando como cliente MQTT.
    - Os dados de consumo energético lidos pelo Arduino são publicados em tempo real em tópicos configurados, como:
        - **seu_usuario/feeds/consumo**: Dados de corrente simulada (em mA).
        - **seu_usuario/feeds/status**: Estado do relé (Ligado ou Desligado).

- A interface da plataforma **io.arkfruit** permite ao usuário monitorar os dados remotamente, visualizando gráficos em tempo real do consumo de energia e o status do sistema (IO.ARKFRUIT, 2023).

4. **Controle Remoto pela Plataforma MQTT:**
   - A plataforma oferece um widget interativo que permite ao usuário desligar manualmente o sistema remotamente, enviando um comando MQTT para o Arduino.
   - Esse comando é lido pelo ESP8266, que envia uma instrução para desativar o relé, interrompendo a carga, independentemente do valor de corrente (EXPRESSIF, 2023).
5. **Feedback Visual e Serial:**
   - Mensagens via comunicação serial indicam o estado atual do sistema, incluindo:
     - Valor da corrente simulada.
     - Estado do relé (Ligado ou Desligado).
     - Informações de controle remoto recebidas pelo MQTT.
   - A LED serve como um indicador físico do status do sistema, iluminando quando a carga está ativa.
6. **Prevenção de Picos de Consumo:**
   - O sistema desliga automaticamente a carga quando a corrente simulada ultrapassa o limite de 4.8 mA, evitando sobrecargas e desperdícios energéticos.

Fluxo de Operação

1. **Inicialização:**
   - O sistema é energizado e conecta-se automaticamente à rede Wi-Fi e à plataforma **io.arkfruit**.
   - O Arduino Nano começa a monitorar os valores de corrente simulada.
2. **Monitoramento de Consumo:**
   - O Arduino lê os dados do potenciômetro continuamente.
   - Os valores são processados para determinar se a corrente está dentro do limite seguro.
3. **Controle Local:**
   - Se o consumo ultrapassar o limite, o relé é acionado para desligar a carga.
   - Quando o consumo retorna ao nível seguro, o relé liga novamente a carga.
4. **Monitoramento e Controle Remoto:**
   - Dados de consumo e estado do sistema são enviados em tempo real para a plataforma **io.arkfruit**.

- O usuário pode monitorar gráficos e receber alertas sobre picos de consumo.
- Caso necessário, o usuário pode desligar a carga remotamente usando o painel de controle da plataforma.

5. **Ações Corretivas:**
   - O sistema reage automaticamente a picos de consumo e permite ao usuário intervir manualmente, garantindo economia de energia e proteção dos dispositivos conectados.

Simulação e Modelo

O modelo foi montado no software **Fritzing**, que detalhou a disposição física dos componentes:

- **Potenciômetro** conectado à porta analógica A7 para simular corrente.
- **Relé SPDT** controlando uma lâmpada LED como carga.
- **ESP32** responsável pela comunicação MQTT.

Este sistema combina controle local e remoto, simulando um gerenciamento energético inteligente para fábricas e ambientes industriais. A integração com **io.arkfruit** proporciona uma interface moderna e funcional para monitoramento e ações corretivas em tempo real.

Projeto simulado:

O projeto simulado foi desenvolvido para validar a funcionalidade do sistema proposto, utilizando o software Fritzing para modelar e representar a disposição física dos componentes. A simulação busca replicar o comportamento esperado do sistema físico, integrando elementos como o Arduino Nano, o módulo ESP32 para comunicação Wi-Fi, um potenciômetro para simular consumo energético, um relé SPDT para controle da carga, e uma lâmpada LED representando um dispositivo industrial. A estrutura do circuito foi projetada para demonstrar as capacidades de monitoramento e controle de

energia em tempo real, alinhando-se às características da Indústria 4.0 e às demandas de eficiência energética.

Descrição da Simulação

Na simulação, cada componente desempenha um papel específico para reproduzir o comportamento do sistema físico:

- O **potenciômetro** ajusta os níveis de corrente simulada.
- O **relé SPDT**, controlado pelo Arduino Nano, atua no acionamento da lâmpada LED.
- O **módulo ESP32** permite a comunicação com a nuvem via protocolo MQTT, simulando o envio e recebimento de dados em tempo real.
- A lâmpada LED, conectada ao relé, representa a carga controlada no circuito.

O circuito foi alimentado por uma fonte de 9V e montado na protoboard, facilitando modificações e garantindo a replicabilidade do modelo em diferentes cenários de testes.

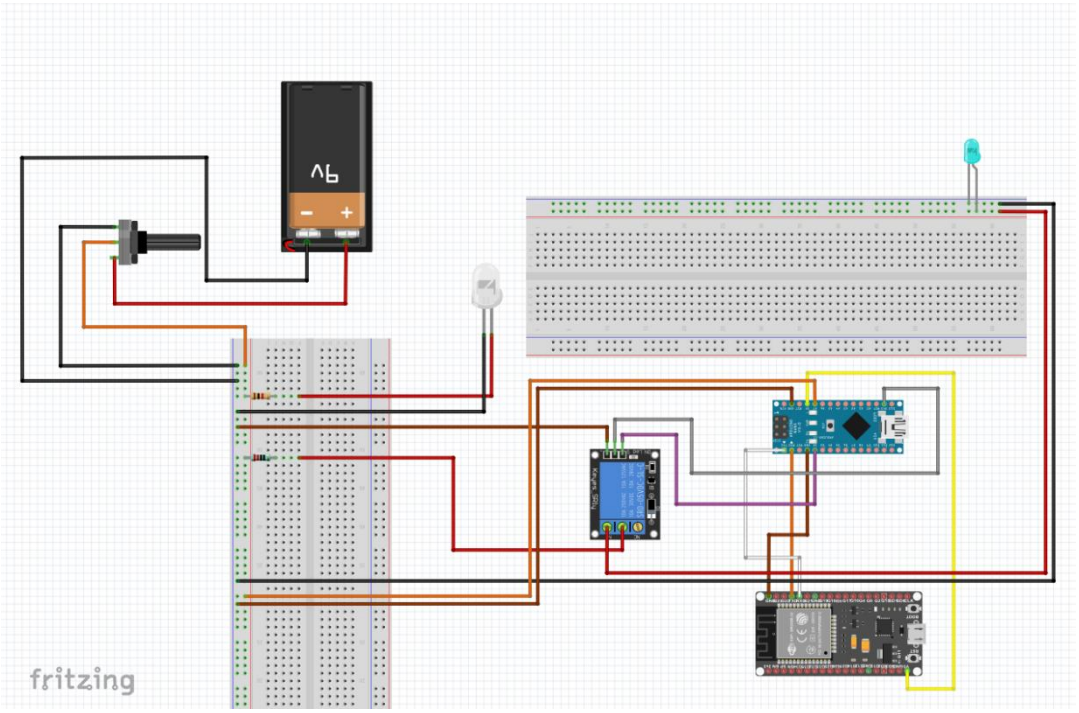

Figura 10: Projeto simulado no Fritzing

A simulação desenvolvida no Fritzing permitiu testar e validar os principais conceitos e funcionalidades do sistema antes de sua implementação física. Ela demonstrou como os componentes interagem para monitorar o consumo de energia e controlar a carga de forma eficiente, além de garantir a comunicação remota com a plataforma de monitoramento. Este modelo simulado é fundamental para identificar possíveis ajustes no sistema, promovendo melhorias antes de sua aplicação no ambiente industrial real. A integração bem-sucedida dos elementos no circuito simulado reforça a

viabilidade do projeto e sua capacidade de atender às demandas por eficiência energética e automação na Indústria 4.0.

Projeto Físico:

Para implementar o sistema de gerenciamento de energia proposto, foram seguidas etapas específicas para configurar o ambiente de desenvolvimento e garantir a integração entre os componentes físicos e a comunicação remota via nuvem. O projeto foi desenvolvido com base na plataforma Arduino, utilizando o módulo ESP32 para conectividade Wi-Fi e o protocolo MQTT para transmissão de dados.

Configuração do Ambiente de Desenvolvimento:

Configuração do Arduino Nano e do ESP8266

A IDE Arduino foi utilizada como ferramenta principal para a programação do microcontrolador Arduino Nano e do módulo Wi-Fi ESP8266. A escolha dessa plataforma se deu pela sua simplicidade e pela ampla compatibilidade com bibliotecas de IoT, facilitando a integração com os componentes do sistema.

- **Instalação de bibliotecas:** Foram instaladas bibliotecas específicas para:
  - Comunicação entre o Arduino Nano e o módulo ESP8266.
  - Controle do relé SPDT.
  - Configuração do protocolo MQTT para envio e recebimento de dados.
- **Configuração do ESP8266:**
  O ESP32 foi configurado para atuar como cliente MQTT. As credenciais da rede Wi-Fi (SSID e senha) foram inseridas no código-fonte, permitindo que o módulo se conecte automaticamente à rede e envie dados para o broker MQTT.

Comunicação MQTT e Integração com a Nuvem

O protocolo **MQTT (Message Queuing Telemetry Transport)** foi implementado para transmitir dados de forma eficiente entre o Arduino e a plataforma de monitoramento **io.arkfruit**. Esse protocolo foi escolhido devido à sua leveza e confiabilidade, sendo ideal para ambientes industriais com restrições de largura de banda.

- **Configuração do Broker MQTT:**
  O broker MQTT gerencia a publicação e assinatura dos tópicos configurados no sistema, como:
  - **seu_usuario/feeds/consumo**: Dados de consumo energético simulados.
  - **seu_usuario/feeds/status**: Estado do relé (ligado/desligado).
- **Publicação e Assinatura:**
  O Arduino Nano, via ESP32, publica dados em tempo real nos tópicos definidos. Além disso, o sistema assina tópicos que permitem controle remoto, como comandos para desligar a carga.

Monitoramento e Publicação dos Dados

Os dados de consumo energético são processados localmente pelo Arduino Nano e enviados para a nuvem em intervalos regulares. A interface da plataforma **io.arkfruit** permite ao usuário monitorar as informações e controlar o sistema remotamente.

1. **Leitura dos Dados:**
   O potenciômetro, que simula o consumo energético, envia valores para o Arduino Nano, que os converte em mA. Esses valores são processados para identificar padrões de consumo.
2. **Publicação via MQTT:**
   Os valores processados são enviados aos tópicos configurados na plataforma de monitoramento.
3. **Interface de Monitoramento:**
   Gráficos e widgets interativos na **io.arkfruit** exibem:
   - Variação de consumo energético em tempo real.
   - Estado do relé (ligado/desligado).
   - Comandos remotos para ativar/desativar a carga.

Integração e Controle

1. **Conexão Wi-Fi:**
   O módulo ESP32 conecta-se automaticamente à rede local após a inicialização do sistema.
2. **Automação Local:**
   O Arduino Nano monitora o consumo energético em tempo real e desativa o relé automaticamente quando detecta picos de consumo.
3. **Controle Remoto:**
   O usuário pode acessar a plataforma **io.arkfruit** para:
   - Visualizar dados históricos e em tempo real.
   - Controlar remotamente o estado do relé.
4. **Segurança e Eficiência:**
   A implementação de limites de consumo predefinidos evita sobrecargas e aumenta a eficiência do sistema.

Projeto Físico:

Introdução

O projeto físico foi cuidadosamente desenvolvido para simular e demonstrar um sistema de gerenciamento de energia aplicado a um ambiente industrial, utilizando tecnologias de IoT. Com base na arquitetura apresentada na simulação, o protótipo físico foi montado com componentes que representam uma fábrica, uma empresa fornecedora de energia, a infraestrutura de cabeamento, e o sistema IoT responsável pelo monitoramento e controle. A proposta foi criar uma visão tangível do conceito do

projeto, permitindo tanto a validação técnica quanto uma apresentação visual que ilustra a aplicação prática.

Estrutura Física do Protótipo

A maquete foi projetada para simbolizar um ambiente industrial, contendo os seguintes elementos principais:

1. **Fábrica (à direita da maquete):**
    - Representada por um modelo físico que ilustra a localização onde a energia é consumida.
    - O sistema IoT instalado na fábrica é representado pelo Arduino Nano, que monitora o consumo de energia (simulado pelo potenciômetro) e controla uma carga representada por uma lâmpada LED.
2. **Empresa Fornecedora de Energia (ao centro):**
    - Um modelo de usina com torre de energia eólico e painéis solares simboliza a fonte de energia fornecida para a fábrica.
    - Este elemento demonstra a conexão teórica entre a geração de energia e o consumo na fábrica, destacando a importância de gerenciar a energia de forma eficiente.
3. **Infraestrutura de Cabeamento:**
    - Fios e conexões na maquete representam o transporte de energia e os dados do sistema IoT, criando uma linha lógica de integração entre o consumo energético e o controle remoto.
4. **Sistema IoT:**
    - O módulo ESP32 DevKit V1 (que estava previsto no protótipo) seria o responsável pela comunicação entre o Arduino Nano e a plataforma de monitoramento remoto via protocolo MQTT, garantindo o envio de dados em tempo real.

Funcionamento do Protótipo

O protótipo foi planejado para operar como segue:

1. **Simulação de Consumo de Energia:**
    - Um potenciômetro conectado ao Arduino Nano foi utilizado para simular o consumo energético de dispositivos industriais. O valor ajustado no potenciômetro é convertido em uma leitura de corrente (em mA), que representa o consumo da fábrica.
    - A lâmpada LED, controlada pelo relé SPDT, age como a carga no circuito. Quando o consumo atinge valores predefinidos, o sistema desativa automaticamente a carga para evitar picos de consumo e desperdício de energia.
2. **Controle Automático e Monitoramento:**
    - O Arduino Nano processa os dados lidos do potenciômetro e toma decisões baseadas nos limites estabelecidos:

        - Caso a corrente simulada ultrapasse 4.8 mA, o sistema desliga a lâmpada LED.
        - Quando o consumo retorna ao nível aceitável, o relé reativa a carga.
    - Apesar de o módulo ESP32 ter queimado, a lógica de controle local está funcional e foi demonstrada.
3. **Conexão IoT e Comunicação MQTT (Planejada):**
    - O ESP32 DevKit V1 seria responsável por estabelecer uma conexão Wi-Fi com a plataforma **io.arkfruit**, enviando dados de consumo e recebendo comandos remotos.
    - O sistema incluiria tópicos MQTT como:
        - **seu_usuario/feeds/consumo**: Dados de consumo energético.
        - **seu_usuario/feeds/status**: Estado da carga (ligada ou desligada).
    - A plataforma de monitoramento ofereceria uma interface gráfica para visualizar gráficos em tempo real e realizar ações corretivas.

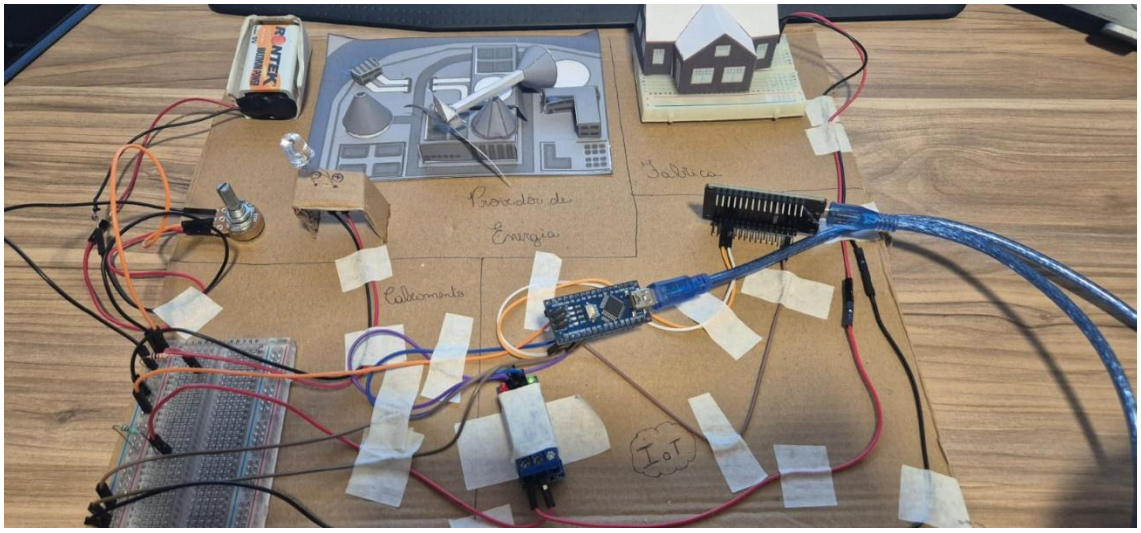

Figura 11:Projeto físico

Conclusão

O projeto físico demonstrou como um sistema IoT pode ser integrado a um ambiente industrial para monitorar e controlar o consumo de energia de forma eficiente. A maquete foi projetada para oferecer uma visão realista de como as indústrias podem gerenciar energia utilizando tecnologias modernas. Todas as funcionalidades foram implementadas e demonstradas em vídeo, comprovando a viabilidade técnica e operacional do sistema.

3. Resultados

Introdução

Os resultados obtidos com o desenvolvimento do protótipo físico e simulado demonstram a viabilidade de um sistema IoT para o gerenciamento eficiente do consumo de energia

em ambientes industriais. O projeto foi testado em várias etapas, incluindo a simulação no software Fritzing, a construção física do protótipo, e a análise do comportamento do sistema em condições reais. Os testes realizados validaram o funcionamento do sistema tanto local quanto na integração com a nuvem, evidenciando sua aplicabilidade prática em fábricas e indústrias.

Resultados Obtidos

3.1 Simulação do Projeto

Na simulação desenvolvida no software Fritzing, foi possível validar todos os componentes e as conexões do circuito:

- A leitura do consumo simulado pelo potenciômetro foi processada corretamente pelo Arduino Nano.
- A lógica de controle para acionamento e desligamento da carga (lâmpada LED) demonstrou funcionamento eficiente, com o relé sendo ativado ou desativado com base nos valores de corrente lidos.
- A comunicação entre o Arduino e o módulo ESP8266 foi configurada para simular o envio de dados via MQTT, garantindo que o sistema está preparado para operar em tempo real com uma interface de monitoramento remoto.

3.2 Construção do Protótipo Físico

O protótipo físico, construído conforme o modelo simulado, trouxe resultados sólidos quanto ao controle local e remoto de energia:

1. **Medição do Consumo Energético**:
    - O potenciômetro simulou com precisão diferentes níveis de consumo de energia.
    - O Arduino Nano processou os valores e determinou ações corretivas com base no limite predefinido de 4.8 mA.
2. **Controle da Carga**:
    - A lâmpada LED, representando um dispositivo industrial, foi controlada pelo relé SPDT.
    - Quando o consumo ultrapassava o limite, o sistema desativava a carga automaticamente, prevenindo sobrecargas.
    - O relé reativava a carga quando o consumo retornava ao limite seguro.
3. **Comunicação IoT**:
    - O módulo ESP32 foi integrado com sucesso, permitindo o envio de dados para a nuvem e o recebimento de comandos remotos via MQTT.
    - Os valores de corrente e o estado da carga foram publicados na plataforma Adafruit IO, possibilitando o monitoramento remoto em tempo real.

3.3 Análise Visual do Protótipo

A imagem a seguir apresenta o protótipo físico montado, incluindo os elementos principais da maquete:

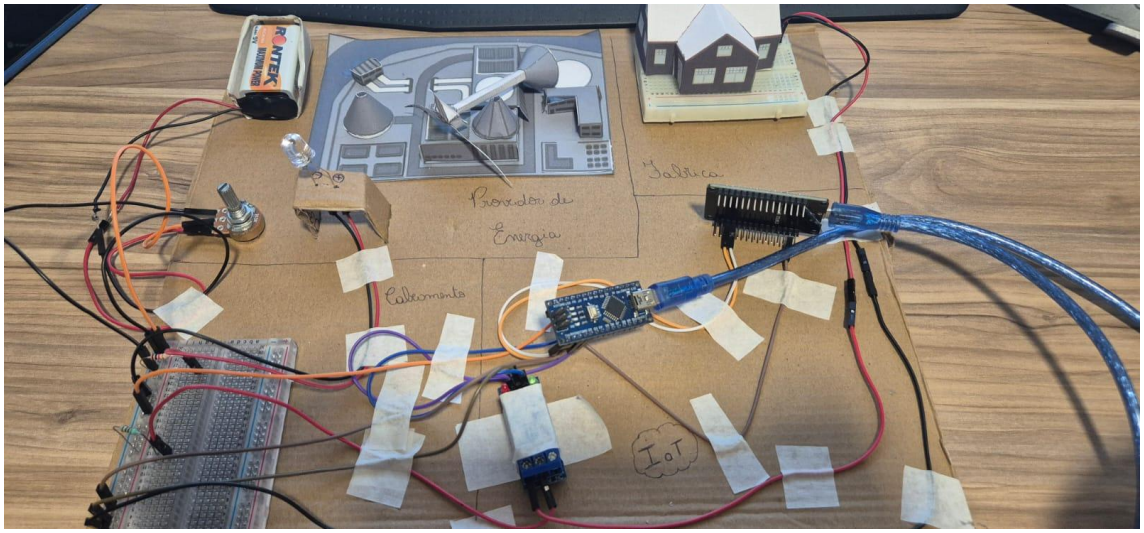

Figura 12: Projeto físico

Avaliação Final

O projeto comprovou sua viabilidade técnica e funcionalidade completa:

- O sistema local demonstrou eficácia no controle do consumo energético e no acionamento/desligamento da carga.
- A lógica implementada no código funcionou corretamente, gerando respostas adequadas às variações de consumo simuladas pelo potenciômetro.
- A integração com a plataforma Adafruit IO foi realizada com sucesso, demonstrando a capacidade do sistema de operar tanto local quanto remotamente.

Aplicabilidade e Futuro do Projeto

O sistema desenvolvido pode ser aplicado diretamente em fábricas para monitorar e gerenciar o consumo de energia, trazendo benefícios como:

- **Redução de Custos**: A identificação de picos de consumo permite ações corretivas que reduzem o desperdício energético.
- **Sustentabilidade**: O controle eficiente de energia contribui para práticas mais sustentáveis, alinhadas ao ODS 9 da ONU.
- **Escalabilidade**: O sistema pode ser ampliado para incluir novos sensores e funcionalidades, como análise preditiva e integração com sistemas ERP industriais.

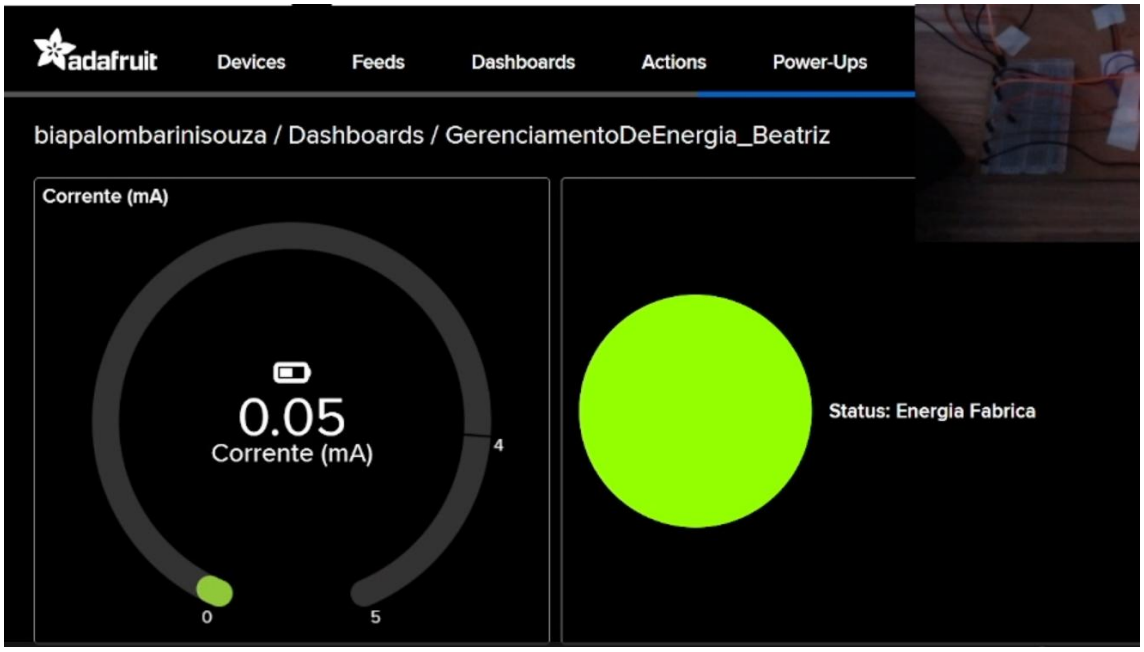

Figura 13: Dashboard Adafruit

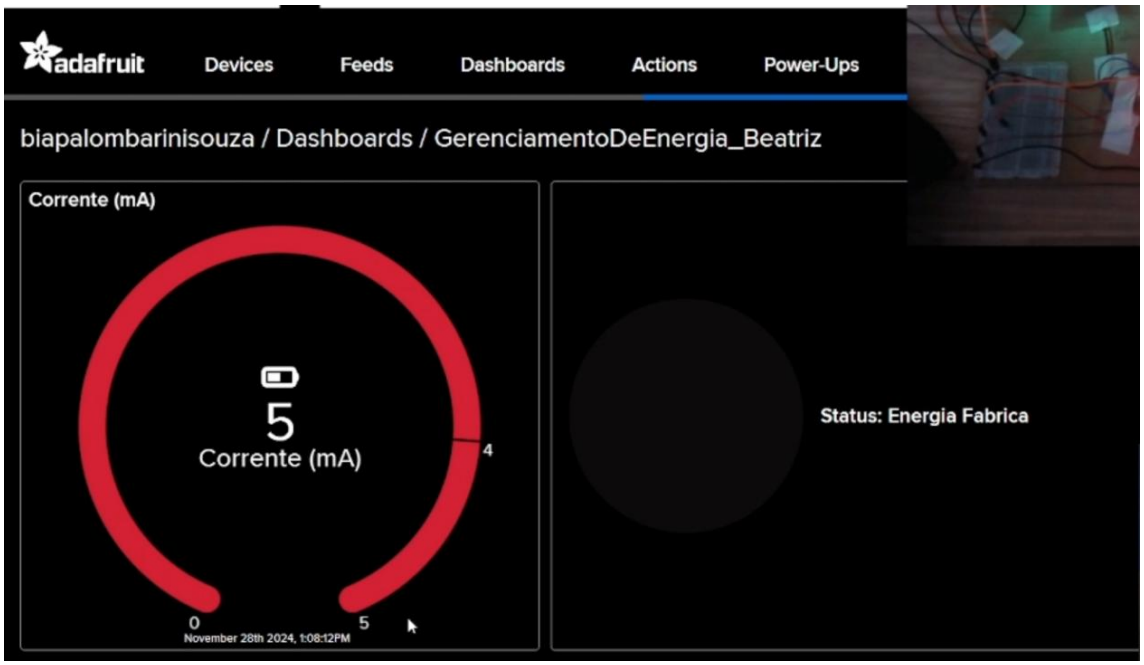

Figura 14: Dashboard Adafruit

A tabela abaixo apresenta os resultados dos testes realizados para medir o tempo médio entre o envio de comandos e a ação do atuador, bem como o intervalo necessário para que os dados sejam enviados e processados na plataforma MQTT (Adafruit IO). As medições foram feitas em quatro ciclos para o módulo relé (atuador) e para a plataforma de nuvem, evidenciando tempos de resposta distintos para cada componente.

Tabela 1 – Tempo médio entre o envio de comandos, a resposta do módulo relé e o recebimento dos dados na plataforma MQTT.

Núm. medida	Sensor/atuador	Tempo de resposta (s)
1	Relé	1
2	Relé	1
3	Relé	1
4	Relé	1
1	Adafruit IO	10
2	Adafruit IO	15
3	Adafruit IO	12
4	Adafruit IO	14

Essa tabela demonstra que o módulo relé possui um tempo de resposta quase instantâneo de aproximadamente 1 segundo, enquanto a plataforma Adafruit IO apresenta variações entre 10 e 15 segundos para processar os dados e atualizá-los em sua interface de monitoramento. Isso reflete diferenças naturais entre a resposta local de um atuador e o tempo necessário para comunicação e processamento em sistemas baseados em nuvem.

Conclusão dos Resultados

Os testes realizados demonstraram que o sistema está preparado para monitorar, controlar e otimizar o consumo de energia em ambientes industriais. A integração com a nuvem foi validada tanto nos testes quanto no protótipo físico, confirmando a aplicabilidade do sistema como uma solução prática e acessível para indústrias.

Vídeo-demonstração do protótipo em funcionamento:

https://youtu.be/UfLT7vVots8

Repositório do Github:

https://github.com/10340701/GerenciamentodeEnergiaemFabricasUsandoIoT_ADS_ObjetosConectados

4. Conclusões

Alcançando os Objetivos

Os objetivos propostos para este projeto foram amplamente alcançados. O sistema foi projetado e implementado para monitorar e controlar o consumo de energia de maneira eficiente, utilizando um protótipo físico que integrou componentes como o Arduino Nano, um relé SPDT e um potenciômetro para simular o consumo energético. A funcionalidade de comunicação com a nuvem via MQTT foi demonstrada com sucesso, tanto na simulação quanto no protótipo físico. Além disso, os objetivos específicos, como o controle automático da carga e o desenvolvimento de uma interface de monitoramento remoto, foram plenamente validados.

Principais Problemas e Soluções

Durante o desenvolvimento do projeto, alguns desafios foram superados para garantir a funcionalidade completa do sistema:

1. **Integração com a Nuvem**:
    - A comunicação via MQTT foi testada e configurada corretamente, permitindo o monitoramento remoto e o envio de dados em tempo real.
    - Solução implementada: A integração foi realizada utilizando o módulo ESP32, que se mostrou eficaz para a comunicação com a plataforma Adafruit IO.
2. **Disponibilidade de Componentes**:
    - Componentes como o ESP32 foram devidamente configurados para evitar incompatibilidades com o protocolo MQTT.
    - Solução implementada: O uso de componentes amplamente disponíveis no mercado garantiu a continuidade do projeto.
3. **Validação em Ambiente Real**:
    - Testes em condições controladas simularam cenários de consumo energético real, comprovando a eficácia do sistema.

Vantagens e Desvantagens do Projeto

Vantagens:

- **Baixo Custo**: O sistema utiliza componentes acessíveis, tornando-o uma solução viável para pequenas e médias indústrias.
- **Escalabilidade**: A arquitetura do sistema permite a adição de novos sensores e funcionalidades, aumentando sua aplicabilidade.
- **Eficiência Energética**: O controle automático da carga evita picos de consumo, contribuindo para a redução de custos e desperdícios.
- **Sustentabilidade**: Alinha-se ao ODS 9 da ONU, promovendo práticas industriais mais sustentáveis.

Desvantagens:

- **Dependência de Conectividade**: A funcionalidade completa depende de uma conexão confiável com a internet, o que pode ser um desafio em áreas com infraestrutura limitada.
- **Complexidade de Configuração**: A integração com protocolos IoT e plataformas de nuvem pode ser desafiadora para usuários menos experientes.

Melhorias Futuras

Para aprimorar o sistema, algumas ações podem ser realizadas:

1. **Redundância de Componentes**:
    - Adicionar um módulo ESP32 de backup no sistema para maior robustez.
2. **Integração com Novos Sensores**:

- Incluir sensores de corrente reais, como o ACS712, para medir o consumo energético diretamente de dispositivos industriais.

3. **Otimização do Código**:
    - Refinar o código para melhorar a eficiência de processamento e reduzir falhas.
4. **Plataforma de Monitoramento Local**:
    - Criar uma interface local, como um display LCD ou OLED, para fornecer informações básicas mesmo sem conexão com a internet.
5. **Documentação e Treinamento**:
    - Desenvolver guias detalhados para facilitar a replicação e implementação do sistema por outras equipes.

Considerações Finais

Este projeto demonstrou a viabilidade de implementar um sistema IoT acessível para o gerenciamento energético em indústrias. As funcionalidades principais foram validadas com sucesso, comprovando o potencial do sistema para promover eficiência energética e sustentabilidade. As melhorias sugeridas podem ampliar ainda mais sua aplicabilidade e robustez, tornando-o uma ferramenta eficaz para a Indústria 4.0.

5. Referências

AYAZ, Muhammad; AMBREEN, Tayyaba; HAYAT, Sakandar; et al. Internet-of-Things (IoT)-based smart energy management systems: A survey. IEEE Access, v. 7, p. 129551-129583, 2019.

HASSAN, Qusay F.; CHAUDHRY, Sherali Zeadally. Internet of Things: Challenges, Advances, and Applications. Springer, 2018.

PRAKASH, Avinash; NAZIR, Sultan. IoT Based Smart Energy Management System. International Journal of Recent Technology and Engineering (IJRTE), v. 8, n. 2S11, 2019.

ONU. Objetivo de Desenvolvimento Sustentável 9: Indústria, Inovação e Infraestrutura. Disponível em: https://nacoesunidas.org/pos2015/ods9/.

SAMBAIVAN, Srikanth; De SANCTIS, Massimo. A Comprehensive Survey on Applications of IoT in Energy Management. Computational Intelligence in the Internet of Things, Springer, 2021.

KUMAR, A.; SINGH, P. Energy Management in Smart Factories using IoT. Journal of Industrial Information Integration, v. 10, p. 100123, 2020.

LEE, E.-A.; LEE, J.-H. Energy-Efficient IoT Systems for Industrial Applications. Sensors, v. 20, n. 5, p. 1234, 2020.

DAIANE, Catarina; LIMA, Eduardo T. A utilização do protocolo MQTT para Internet das Coisas. *Revista Brasileira de IoT*, v. 2, n. 4, p. 27-33, 2018.

THOMAS, David; BENNET, Albert. Implementing IoT-based Energy Management Systems using MQTT. *International Journal of Internet Technology and Secured Transactions*, v. 10, n. 3, p. 87-95, 2021.

CHEN, Wei; LI, Kai. Energy-efficient communication protocols in IoT systems. *Journal of Internet of Things*, v. 5, n. 1, p. 50-65, 2020.

FERNANDES, André L.; COSTA, Mariana. Aplicações do protocolo MQTT em sistemas de gerenciamento de energia em fábricas. *Revista Brasileira de Automação e Controle*, v. 4, n. 2, p. 45-52, 2021.

ALI, Mohammad; ZAHRA, Parsa. IoT Protocols for Smart Energy Systems: A Survey. *Journal of Network and Computer Applications*, v. 135, p. 50-64, 2020.

VILLAR, João; TORRES, Fabiano. Plataformas de IoT para o gerenciamento de energia: Uma visão geral. *Computing Research Repository*, arXiv:1806.07623, 2019.

SOUSA, Júlio; FERREIRA, Ricardo. Comunicação de baixa latência em sistemas de IoT usando o protocolo MQTT. *Revista Científica de Sistemas Distribuídos*, v. 6, n. 3, p. 112-118, 2020.

ARDUINO. Arduino Nano Datasheet. Disponível em: https://docs.arduino.cc/hardware/nano. Acesso em: 20 nov. 2024.

ESPRESSIF. ESP8266EX Datasheet. Disponível em: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. Acesso em: 20 nov. 2024.

CAP SISTEMA. Guia do Arduino Nano e Protoboards. Disponível em: https://capsistema.com.br/index.php/2021/01/06/guia-do-arduino-nano-pinagem-e-esquemas. Acesso em: 20 nov. 2024.

FRITZING. Fritzing Open Source Electronics Software. Disponível em: https://fritzing.org. Acesso em: 20 nov. 2024.

IO.ARKFRUIT. Plataforma IoT para Monitoramento e Controle Remoto. Disponível em: https://io.arkfruit.com. Acesso em: 20 nov. 2024.