UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

CAIO MARCOS PLÍNIO ROVIERI MARTINS

ESTUDO E SIMULAÇÃO DE MEDIDAS DE ENERGIA PARA SISTEMAS ELÉTRICOS

Sorocaba
2024

CAIO MARCOS PLÍNIO ROVIERI MARTINS

ESTUDO E SIMULAÇÃO DE MEDIDAS DE ENERGIA PARA SISTEMAS ELÉTRICOS

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Ivando Severino Diniz

Sorocaba

2024

M386e

Martins, Caio Marcos Plínio Rovieri
Estudo e simulação de medidas de energia para sistemas elétricos / Caio Marcos Plínio Rovieri Martins. -- Sorocaba, 2024
61 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Ivando Severino Diniz

1. Energia elétrica. 2. Internet das coisas. 3. Microcontroladores. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

ESTUDO E SIMULAÇÃO DE MEDIDAS DE ENERGIA PARA SISTEMAS ELÉTRICOS

CAIO MARCOS PLÍNIO ROVIERI MARTINS

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DOS REQUISITOS PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Everson Martins

Coordenador

BANCA EXAMINADORA:

Prof. Dr. IVANDO SEVERINO DINIZ
Orientador/UNESP – Campus de Sorocaba

Eng. VINICÍUS F. DE SOUZA
UNESP – Campus de Sorocaba

Prof. Dr. EVERSON MARTINS
UNESP – Campus de Sorocaba

Sorocaba

11 de junho de 2024

De modo especial, aos meus pais e meus irmãos, que foram os grandes incentivadores ao longo da minha formação.

AGRADECIMENTOS

Agradeço primeiramente à minha família, pelo amor, carinho e respeito ao cuidarem de mim, como também, por terem investido e dedicado as suas vidas em prol da minha.

Ao meu orientador, Professor Doutor Ivando Severino Diniz, por todo apoio e suporte prestado ao longo do trabalho, como também, pela confiança e companheirismo.

Aos amigos da República Magnatas, por serem a minha família na cidade de Sorocaba e terem estado ao meu lado ao longo da minha formação.

À UNESP e seus membros por terem contribuído com a minha formação, instigando e incentivando a busca pelo conhecimento, como também, fornecendo o conhecimento e suporte necessário ao longo da graduação.

RESUMO

A rápida e crescente demanda por produtividade e eficiência em todos os setores da indústria, suscita a necessidade de contínuo aperfeiçoamento das tecnologias empregadas nos processos cada vez mais integradas com sistemas de automação, monitoramento e controle. Esta integração de dispositivos e sistemas no contexto da Internet das Coisas fornece ferramentas que possibilitam monitorar e produzir informações para analisar operações de diversas naturezas dentro de um processo produtivo de forma precisa, eficiente e relativamente barata. Com isto em vista, o trabalho consiste na descrição de um sistema para coleta de dados do comportamento elétrico de uma grua, é uma máquina central no canteiro de obras, responsável por transportar verticalmente grande quantidade de material e tem alto consumo de energia elétrica. Para realizar o monitoramento é proposto um sistema microcontrolado capaz de coletar, processar e enviar dados que descrevem a operação da máquina em termos de suas variáveis elétricas de tensão e corrente.

Palavras-chave: Internet das coisas; Microcontroladores; STM32; Monitoramento de energia.

ABSTRACT

The rapid and growing demand for productivity and efficiency in all industry sectors raises the need for continuous improvement of technologies used in processes that are increasingly integrated with automation, monitoring and control systems. This integration of devices and systems in the context of the Internet of Things provides tools that make it possible to monitor and produce information to analyze operations of different natures within a production process in a precise, efficient and relatively cheap way. With this in mind, the work consists of describing a system for collecting data on the electrical behavior of a crane. It is a central machine on the construction site, responsible for vertically transporting a large amount of material and has a high consumption of electrical energy. To carry out monitoring, a microcontrolled system is proposed capable of collecting, processing and sending data that describes the machine's operation in terms of its electrical voltage and current variables.

Keywords: Internet of things; Microcontrollers; Stm32; Energy monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

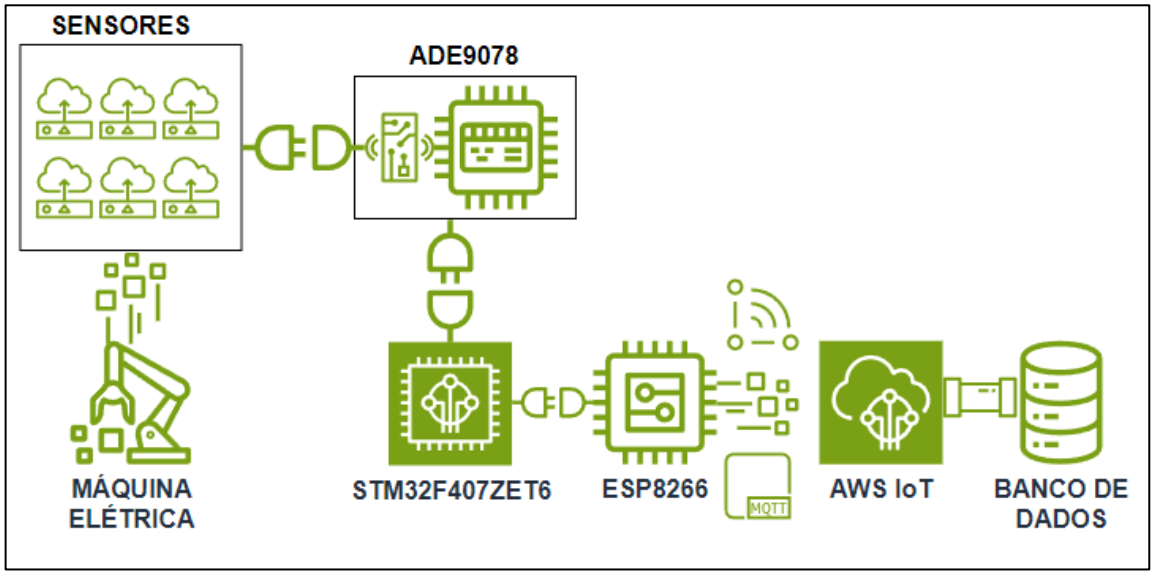
1 INTRODUÇÃO

> A Indústria 4.0, ou a quarta Revolução Industrial, é comumente considerada como a automação e digitalização da indústria e dos sistemas de produção. As tecnologias IoT e de nuvem tornaram-se críticas facilitadores desse esforço e fornecem a capacidade de integrar e automatizar máquinas para se tornarem mais inteligente e adaptável. (1, tradução própria, p.4). Podemos usar hardware e software IoT para atingir os objetivos da Indústria 4.0, fornecendo uma solução robusta e um conjunto de tecnologias de força industrial que permitem a instrumentação e medição de equipamento e seu ambiente. Tenha em mente que a indústria é muitas vezes conduzida em condições ambientais extremas. Condições e a abordagem mais barata muitas vezes não é a abordagem correta. Um trade-off entre custo e a confiabilidade deve ser considerada, pois se você precisar substituir um componente com muita frequência, o valor poderá ser perdido em esforço, tempo ou perda de dados enquanto espera a mudança ocorrer (1, p. 10, tradução própria).

O foco deste trabalho é descrever o emprego das tecnologias de informação e de sistemas embarcados com o intuito de coletar dados da operação desta máquina em termos suas grandezas elétricas. A coleta e envio de dados é projetada para que sensores de corrente e tensão conectados à entrada de energia, no quadro de disjuntores na base da grua, façam leituras dessas variáveis, transmitam para os microcontroladores conectados a nuvem.

O projeto propõe um sistema que integra as etapas de aquisição, processamento e envio de dados composto pelos dispositivos de borda o ADE9078, circuito integrado monitor de energia responsável pela leitura e processamento dos dados vindos dos sensores, STM32F4ZET6, microcontrolador central que configurado para receber os dados e permite, formata e controla o fluxo de dados para o dispositivo de módulo Wi-Fi, e ESP8266, o microcontrolador utilizado para conexão Wi-Fi e envio de informações via MQTT que integra do o sistema de borda com a nuvem.

Figura 1 - Esquema do sistema de monitoramento



Fonte: Autoria própria

Este fluxo de dados continua na aplicação em nuvem da AWS a qual possui diversas ferramentas dedicadas a sistemas de Internet das Coisas para receber, transformar, organizar e armazenar os dados um banco de dados específicos para aplicações como esta.

A elaboração do sistema de monitoramento descrito é apresentada em 4 partes. A primeira parte é dedicada a trazer os conceitos das tecnologias de internet das coisas, IoT, e dispositivos envolvidos na solução, como os sensores e microcontroladores.

A metodologia reuni a descrição de todos os dispositivos, como são ligados, como se combinam e se comunicam entre si, utilizando as ferramentas de software para a programação. O fluxo de dados é descrito em cada parte desde a captação pelos sensores até seu envio para a nuvem.

A parte seguinte, traz os resultados e das integrações dos dispositivos sensores, medidores e microcontroladores que estabelecem o fluxo de dados seguidos na descrição do sistema de monitoramento.

A parte final traz a conclusão do trabalho e trabalhos futuros que podem ser desenvolvidos de forma a adicionar recursos, funcionalidades e segurança ao sistema.

2 CONCEITOS FUNDAMENTAIS

A seguir os conceitos utilizados no trabalho para construir uma solução separados em tópicos abordando os principais componentes e tecnologias dentro do contexto da aplicação da Internet da Coisas e monitoramento da energia de ativos elétricos.

Iniciando com o dispositivo central da solução, o microcontrolador, capaz de integrar a aquisição de dados, realizar processamento destes dados e transmiti-los para um destino específico. Na sequência, são apresentados os periféricos responsáveis pela coleta de dados, os sensores, os quais realizam leituras de variáveis físicas relacionadas à operação que se deseja monitorar, portanto são a interface dos microcontroladores com o mundo real. Após conceituar a parte de hardware do sistema, o tópico seguinte explora o conceito de Internet of Things (IoT) apresentando a ideia de integração dos sensores, microcontroladores com aplicação em nuvem que recebe os dados coletados, processados e transmitidos pelos dispositivos de borda.

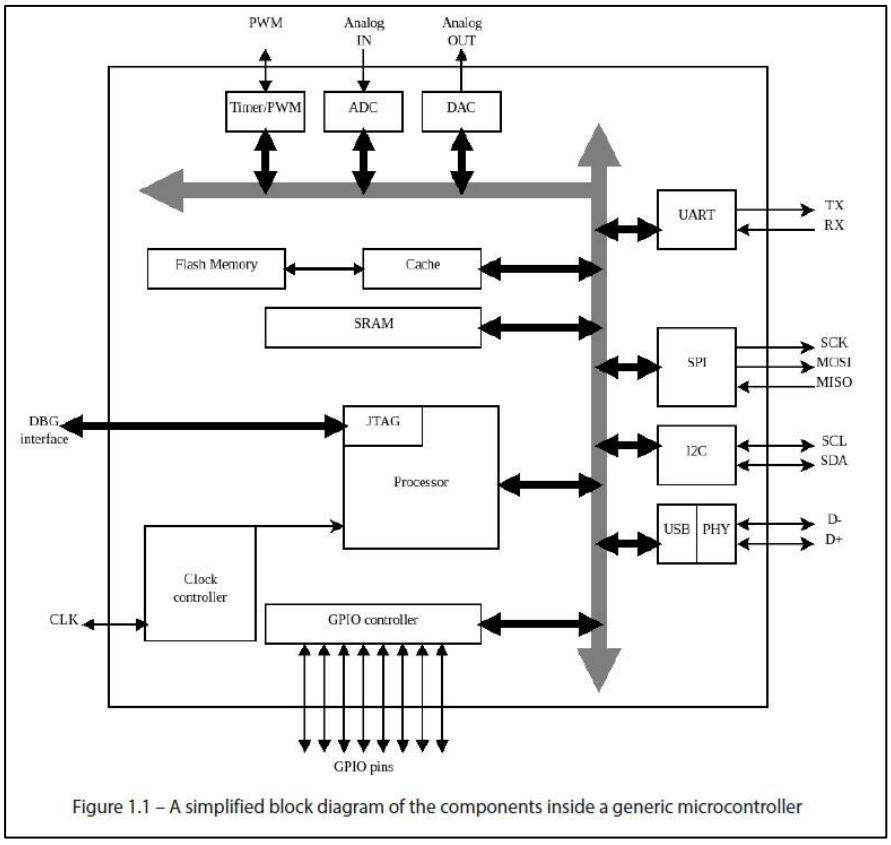
2.1 Microcontroladores

A arquitetura de um sistema embarcado é centrada em torno de seu microcontrolador, às vezes também referida como unidade microcontroladora (MCU). Este é normalmente um único circuito integrado contendo o processador, memória RAM, memória flash, receptores e transmissores seriais e outros componentes principais. O mercado oferece muitas opções diferentes entre arquiteturas, fornecedores, faixas de preço, recursos e recursos integrados. Geralmente são projetados para serem baratos, de baixo consumo de recursos, de baixo consumo de energia e independentes sistemas, em um único circuito integrado, razão pela qual são frequentemente chamados como System-on-Chip (SoC).

Os endereços onde a RAM e a memória Flash são mapeadas dependem do modelo específico e são geralmente fornecidos no *datasheet*. Um microcontrolador pode executar código em sua linguagem de máquina nativa; isto é, uma sequência de instruções transmitidas em um arquivo binário específico da arquitetura em que está continuando. Por padrão, os compiladores fornecem um arquivo executável genérico como saída da compilação e operações de montagem, que precisam ser convertidas em um formato que possa ser executado pelo destino.

A parte do Processador é projetada para executar as instruções que foram armazenadas em seu próprio formato binário diretamente da RAM, bem como de sua memória flash interna. Isso geralmente é mapeado começando da posição zero na memória ou outro endereço conhecido especificado no microcontrolador manual. A CPU pode buscar e executar código da RAM mais rapidamente, mas o firmware final é armazenado na memória flash, que geralmente é maior que a RAM em quase todos os microcontroladores e permite reter os dados durante ciclos de energia e reinicializações. A figura 2 traz um diagrama de blocos simplificado com os principais componentes encontrados na maioria dos microcontroladores.

Figura 2 - Diagrama de componentes de microcontrolador genérico

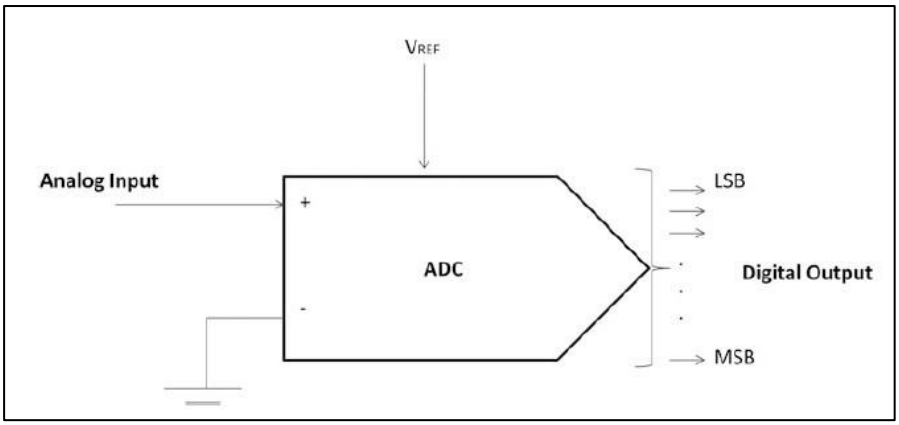


Fonte: (3)

> Compilar um ambiente operacional de software para um microcontrolador embarcado e carregá-lo a memória flash requer uma máquina host, que é um conjunto específico de ferramentas de hardware e software. Alguns conhecimentos sobre as características do dispositivo alvo também são necessários para instruir o compilador a organizar os símbolos dentro da imagem executável. Por muitas razões válidas, C é a linguagem mais popular em software embarcado, embora não seja a única opção disponível.

> Linguagens de nível superior, como Rust e C++, pode produzir código incorporado quando combinado com um tempo de execução incorporado específico, ou até mesmo em alguns casos, removendo totalmente o suporte de tempo de execução da linguagem (2, tradução própria).
>
> A conexão do circuito digital ao dispositivo sensor só pode ser feita se os sensores são inerentemente digitais. No entanto, quando sinais analógicos estão envolvidos no projeto, a interface se torna muito mais complexa. A figura 3 ilustra o funcionamento de um conversor analógico-digital. Neste caso, é necessária uma maneira de traduzir sinais analógicos em formato digital - um ADC (Analogic-Digital Converter); conversor digital para analógico ou DAC (Digital-Analogic Converter) executa a operação oposta (2, p.17, tradução própria).

Figura 3 - Diagrama Analógico-Digital-Conversor genérico



Fonte: (2)

Todos os ADCs sofrem de erros de não linearidade causados por suas imperfeições físicas, permitindo que sua saída se desvie de uma função linear (ou alguma outra função, em o caso de um ADC deliberadamente não linear) de sua entrada. Esses erros às vezes podem ser mitigados pela calibração ou evitado por testes. Parâmetros importantes são não linearidade integral (INL) e não linearidade diferencial (DNL). Estes reduzem a faixa dinâmica dos sinais analógicos que podem ser digitalizados pelo ADC, reduzindo também a resolução efetiva da ADC.

O custo de um ADC é proporcional aos seguintes parâmetros - precisão, número de bits e estabilidade. Mas mesmo o ADC mais caro pode comprometer precisão quando ruído excessivo interfere no sinal de entrada, seja esse sinal está em milivolts ou muito maior.

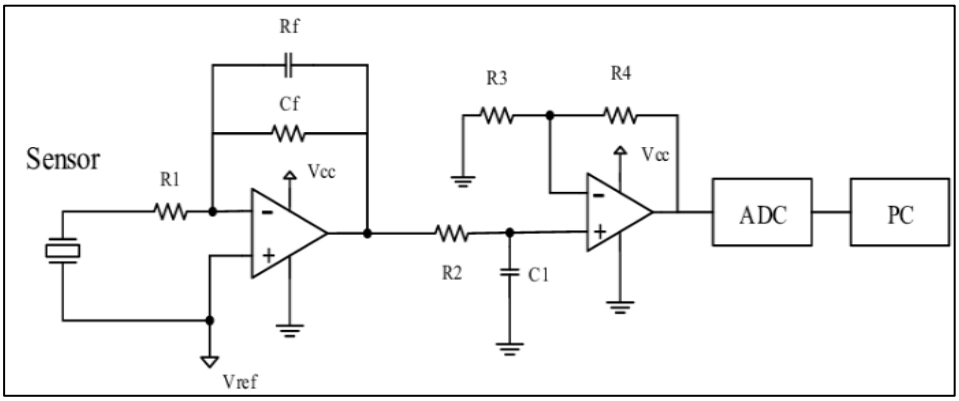
2.2 Sensores

Transdutores e sensores são usados para converter um fenômeno físico em um sinal elétrico (tensão ou corrente) que será então convertido em um sinal digital usado para o próximo estágio, como um computador, sistema digital ou placa de memória (2, tradução própria, p.3).

O sensor ideal converte apenas uma única quantidade física, química ou geométrica em um sinal elétrico de uma forma altamente linear e é insensível a outras quantidades. Na prática, isto é muitas vezes difícil de conseguir; em particular, a temperatura influencia o valor medido (3, tradução própria, p.2).

A primeira operação no sistema DAQ é o condicionamento do sinal. O sinal do sensor deve primeiramente ser enviado ao sistema eletrônico para transformar o aumento ou nível decrescente de amplitude. No entanto, para ser útil à interface dispositivos, o sinal pode ser enviado em algumas formas de condicionamento. Em geral, todos dispositivos de interface são projetados para permitir a interface de sinais de detecção em uma tensão faixa de 0 a 5 V que será digitalizada no conversor A/D. Em geral, os dispositivos de condicionamento são projetados para serem flexíveis, a fim de alterar essa faixa (2, p. 14, tradução própria).

Figura 4 - Circuito genérico de condicionamento de sinal



Fonte:(4)

“O processo de condicionamento envolve uma combinação de processos mais simples que pode ser usado em todos os tipos de sinais - conversão de uma resistência em tensão, divisão, amplificando e mudando uma tensão, como ilustra a figura 4 acima” (2, p. 14, tradução própria).

2.3 Internet of Things - IoT

> A revolução da Indústria 4.0 pode ser resumida em uma palavra - ‘Conectividade’. Conectividade permitirá a produção inteligente com a proliferação de IIoT (Industrial Internet of Things), nuvem e big data. Dispositivos inteligentes podem coletar vários dados sobre localização interna, posição externa, informações de status, padrões de uso dos clientes, etc. Eles têm a capacidade não apenas de coleta de informações, mas também o compartilhamento das informações entre os pares pretendidos. Isso será benéfico na construção de um processo de fabricação eficiente em indústrias ambientes e também auxiliando nas manutenções preventivas planejadas na maquinaria. O outro benefício está na identificação de erros na esteira de produção o mais rápido possível, pois é um fator importante para reduzir custos de produção e de manutenção. A Indústria 4.0 também está focando em problemas de otimização na indústria usando dispositivos inteligentes para utilizar serviços orientados a dados. Indústria 4.0 e IIoT são usados para compartilhamento de tarefas complexas, tomada de decisões com base em informações coletadas dados e acesso remoto a máquinas. Conectividade massiva das coisas e dados a capacidade de coleta/compartilhamento desses promove a segurança como um requisito importante para os conceitos de IIoT e Indústria 4.0 (5, p. 3, tradução própria).

A conectividade faz a possibilidade de integração de dispositivos com a nuvem o que contribui de forma determinante na capacidade de monitorar, controlar e analisar o processo produtivo das indústrias. Essa conectividade com a nuvem amplia as possibilidades da Indústria 4.0 pois este novo paradigma fornece muitas ferramentas e recursos de como coletar, transferir, transformar e analisar grandes quantidade de dados vindos inúmeros dispositivos todos conectados.

> A computação em nuvem é um paradigma de computação disruptiva e, como tal, exigiu grandes mudanças nas muitas áreas da ciência da computação e engenharia da computação, incluindo armazenamento de dados, arquitetura de computadores, rede, gerenciamento de recursos, agendamento e, por último, mas não menos importante, segurança do computador. Em 2011, o NIST, o Instituto Nacional de Padrões e Tecnologia dos EUA, definiu a computação em nuvem como “um modelo para viabilizar o acesso de rede onipresente, conveniente e sob demanda a um conjunto compartilhado de

> recursos configuráveis de computação (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços". Os usuários de serviços de computação pagam conforme consomem computação, armazenamento e comunicação recursos. Embora a computação utilitária geralmente exija uma infraestrutura semelhante à nuvem, o foco da computação em nuvem está no modelo de negócios para fornecer serviços de computação. A computação em nuvem é caracterizada por cinco atributos - autoatendimento sob demanda, amplo acesso à rede, agrupamento de recursos, elasticidade rápida e serviço medido. DBaaS – banco de dados como serviço é um serviço mais recente além dos três modelos de entrega de serviços em nuvem - SaaS – Software como Serviço, PaaS – Plataforma como um Serviço e IaaS – Infraestrutura como Serviço (6, p. 2, tradução própria).

2.4 Protocolo MQTT

> O protocolo MQTT (Message Queuing Telemetry Transport) é um protocolo de conectividade de máquina-a-máquina (M2M) e IoT. MQTT é um protocolo de mensagens leve que funciona com um servidor (broker) baseado mecanismo de publish-subscribe e é executado sobre TCP/IP (Protocolo de Controle e Transmissão/Protocolo de Internet) (7, p. 30, tradução própria).
>
> O broker funciona como um servidor que recebe as mensagens publicadas pelos clientes e distribui para os assinantes de acordo com os tópicos de interesse. Esse tipo de comunicação demonstra-se muito eficaz pois possibilita o desacoplamento direto entre quem publica a mensagem e quem assina para receber o conteúdo (8, p. 24).

- Tópico do MQTT - O termo "tópico" refere-se a palavras-chave que o agente MQTT usa para filtrar mensagens para os clientes MQTT. Os tópicos são organizados de maneira hierárquica, semelhante a um diretório de arquivos ou pastas.

- Publicação de MQTT - Os clientes MQTT publicam mensagens contendo o tópico e os dados em formato de bytes. O cliente determina o formato de dados, como dados de texto, dados binários, arquivos XML ou JSON.

- Assinatura de MQTT - Os clientes MQTT enviam uma mensagem SUBSCRIBE (ASSINAR) ao agente MQTT para receber mensagens sobre tópicos de interesse. Esta mensagem contém um identificador exclusivo e uma lista de assinaturas.

3 METODOLOGIA

Nesta seção são abordas as etapas necessárias para o desenvolvimento e os componentes envolvidos na solução proposta de monitoramento de máquina elétrica como aplicação do conceito de Internet das Coisas (Internet of Things).

O sistema tem o objetivo de monitorar a máquina através dos valores de tensão e corrente consumidos durante todo o tempo, utilizando sensores e medidores de energia integrados com microcontroladores. Os dados coletados e processados são enviados para uma aplicação em nuvem que os armazena. Para isto, a descrição da solução proposta é subdivida em 3 subsistemas responsáveis pelas etapas de:

- Aquisição de Dados
- Processamento e Envio de Dados
- Armazenamento das Informações

3.1 Aquisição de Dados

Para elaboração de um sistema de aquisição de dados formado por sensores de tensão e corrente é necessário conhecer as informações referentes aos parâmetros elétricos utilizados no funcionamento da grua, a máquina elétrica objeto do sistema de monitoramento.

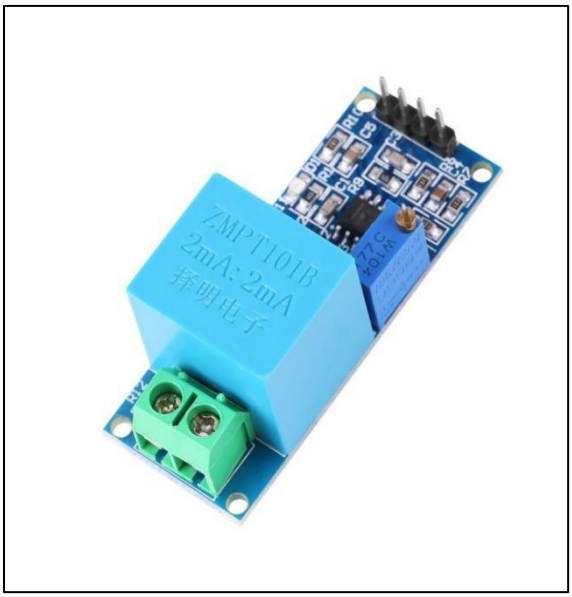
A máquina em questão traz em sua documentação a potência máxima de 47 kW com alimentação trifásica de 380 V. Com isto é possível compreender os requisitos mínimos de faixa de valores dos sensores para esta máquina para que sejam extraídas as leituras de suas variáveis elétricas de forma adequada, precisa e segura. Este dimensionamento é importante também para que não ocorra incidentes e cause danos aos componentes, dispositivos e cabeamento.

Outro ponto que deve ser levado em consideração é a complexidade de implantação e acoplamento deste sistema ao quadro elétrico de alimentação da máquina pois os sensores são dispositivos de contato próximo ao cabeamento o qual conduz alta potência devido dimensão da máquina. Portanto, os sensores escolhidos para este objetivo devem atender os parâmetros elétricos e não oferecer dificuldade ou perigo sem necessidade.

Com isto em vista, os sensores de tensão e corrente foram determinados considerando os fatores de precisão e resolução convenientes para o intervalo de valores que a máquina apresenta, complexidade de acoplamento com o cabeamento, segurança da instalação e custos dos sensores.

- Sensor de Tensão AC - Transformador - ZMPT101B
  É um módulo de alta precisão que tem como finalidade detectar a existência de tensão alternada em um circuito ou fazer a medição do valor de tensão, dispositivo da figura 5. O sensor de tensão utiliza o método de divisor de tensão para atenuar o sinal para efetuar a medição, processo ilustrado na figura 6, a seguir.

Figura 5 - Sensor de Tensão ZMPT101B



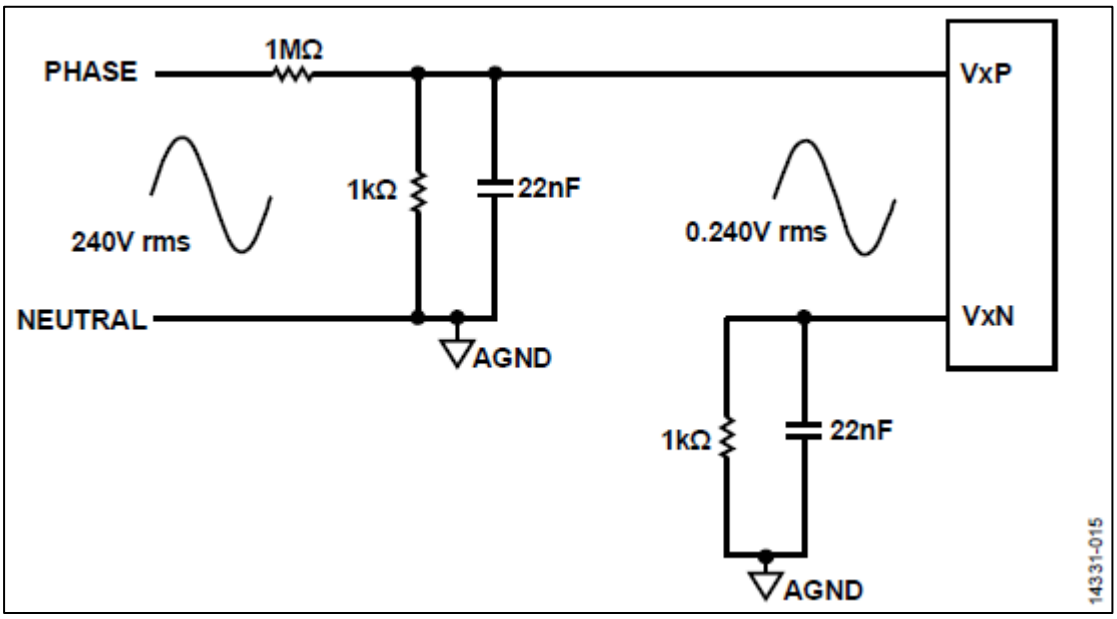
Fonte: (9)

Especificações:

- Tipo de sensor - detector de tensão / voltímetro
- Sinal de saída - 0 - 5VDC analógico
- Tensão de alimentação do módulo - 5 a 30VDC
- Tensão de entrada - 0 a 250VAC
- Proporção - 1000:1000
- Faixa linear - 0-1000V
- Linearidade - 0,2%

- Isolamento tensão - 4000V
- Precisão de leitura - ±1%
- Temperatura de operação - -40º a 70º celsius
- Dimensões - 22mm(L) X 20mm(A) X 51mm(C)

Figura 6 - Circuito de condicionamento de sinal de tensão



Fonte:(10)

- Sensor de Corrente Transformador – SCT013 50A - 1V

> O sensor de corrente alternada SCT-013 tem como principal vantagem o fato de não precisar de contato elétrico com o circuito para medir a corrente elétrica. Ou seja, não precisamos abrir o circuito para ligá-lo em série com a carga, basta apenas "abraçar" um dos fios ligados ao equipamento a ser monitorado. SCT é a sigla para Split-core Current Transformer, ou seja, Transformador de corrente de núcleo dividido (11).

A figura 8 traz a ilustração do funcionamento do sensor empregando o método de transformador de corrente para fazer a leitura e o condicionamento desse sinal analógico.

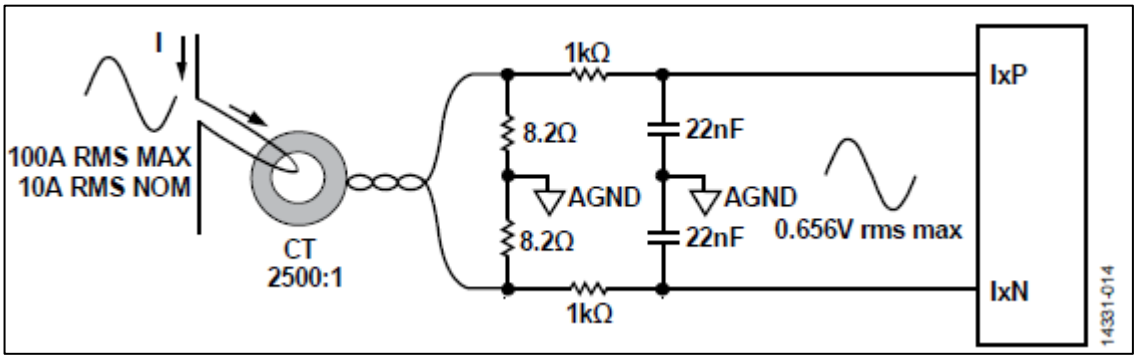
Figura 7 - Sensor de Corrente não-invasivo – SCT013 50A

Fonte:(12)

Especificações:

- Modelo - SCT-013-050;
- Corrente de entrada - 0-50A;
- Sinal de saída - 1VDC;
- Material do Core - Ferrite;
- Dielétrico - 6000V AC/1min;
- Taxa antichama - UL94-V0;
- Dimensão abertura - 13 x 13mm;
- Temperatura de trabalho - -25 a +70°C;
- Comprimento do cabo - 100cm;

Figura 8 - Circuito de Condicionamento de sinal de corrente



Fonte: (10)

A transmissão destes sinais analógicos provindos dos sensores deve atender os parâmetros de entrada de sinais da unidade microcontroladora, portanto é necessário um circuito de condicionamento do sinal. O condicionamento do sinal de entrada tem como objetivo é modular o sinal que atenda o intervalo de valores do dispositivo que recebe este sinal.

3.2 Processamento e Envio de Dados

Etapa central da solução para o monitoramento de energia da máquina pois os sinais analógicos são convertidos em digital, processados e são transformados em informações com os parâmetros de interesse sobre a operação em termos de correntes e tensões. Os dispositivos responsáveis por esta etapa são os dois microcontroladores STM32F407ZET6 e ESP8266, e o dispositivo de medição de energia o ADE9078.

- **ADE9078 -** Dispositivo de medição de energia, parte de uma classe chamada AFE (Analog Front-End). Responsável por receber os sinais analógicos, após o condicionamento do sinal vindo dos sensores, e converter em sinais digitais utilizando os ADCs. Com a leitura dos dados de tensão e corrente este medidor processa os dados e calcula dezenas de parâmetros diretos e derivados, como valores médios, RMS, instantâneos das tensões e correntes, potências ativas, reativa, aparente etc.

- **STM32F407ZET6** - Microcontrolador central do sistema o qual tem a função de consumir os dados disponibilizados pelo ADE9078 através de comunicação serial e dar sequência no fluxo de dados. Estes dados são preparados para a transmissão para nuvem, ou seja, organizados e formatados pelas tarefas do RTOS (Real-Time Operational System), o FreeRTOS, embarcado no microcontrolador para gerenciar e controlar do seu funcionamento a partir de rotinas programadas.

- **ESP8266** - Microcontrolador com a função de integrar o fluxo de dados da borda para a nuvem. O módulo Wi-Fi em sua placa permite a comunicação com a aplicação em nuvem para transmissão dos dados já transformados e prontos para serem armazenados em bancos de dados. O ESP8266 é o último dispositivo de borda do sistema de monitoramento que utiliza o protocolo de comunicação MQTT para a transmissão dos dados coletados, processados e transformados até o banco de dados da AWS.

3.2.1 ADE9078

O ADE90781 é um dispositivo de medição de energia altamente preciso e totalmente integrado. Faz interface com ambos os transformadores de corrente (CT) e sensores de bobina Rogowski, o ADE9078 permite aos usuários desenvolverem uma plataforma de metrologia trifásica. A figura 9 traz os valores máximos dos parâmetros do medidor ADE9078

Figura 9 - Valores absolutos máximos – ADE9078

ABSOLUTE MAXIMUM RATINGS

T_A = 25°C, unless otherwise noted.

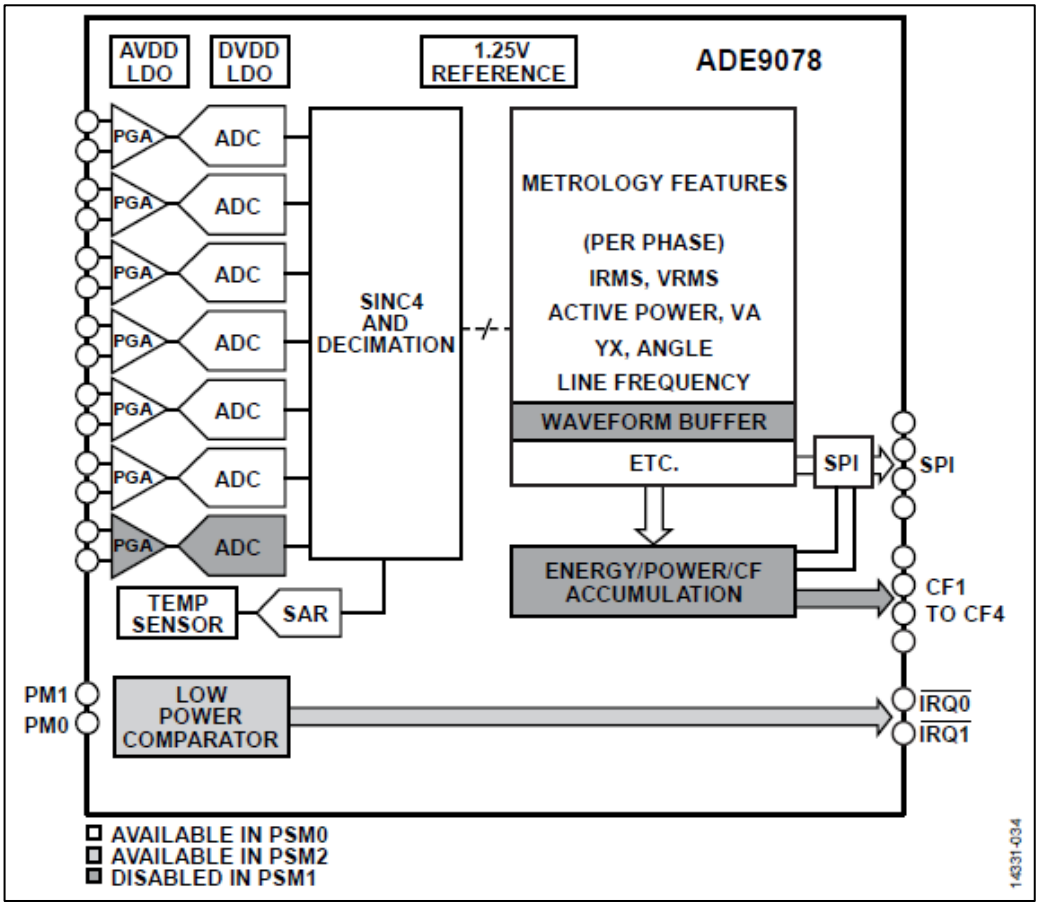
Table 3.

Parameter	Rating
VDD to GND	−0.3 V to +3.96 V
Analog Input Voltage to GND, IAP, IAN, IBP, IBN, ICP, ICN, VAP, VAN VBP, VBN, VCP, VCN	−1.9 V to +2 V
Reference Input Voltage to REFGND	−0.3 V to +2 V
Digital Input Voltage to GND	−0.3 V to V_{DD} + 0.3 V
Digital Output Voltage to GND	−0.3 V to V_{DD} + 0.3 V
Operating Temperature	
Industrial Range	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec)[1]	260°C
ESD	
Human Body Model[2]	4 kV
Machine Model[3]	200 V
Field Induced Charged Device Model (FICDM)[4]	1.25 kV

Fonte: (10)

Os sinais de entrada no IAP, IAN, IBP, IBN, ICP, ICN, VAP, VAN, VBP, VBN, VCP e VCN não devem exceder 0,6 V relativos para AGND, a referência de terra analógica. O diferencial em escala completa a faixa de entrada dos ADCs é de ±1 V de pico (0,707 V rms) e a tensão de modo comum máxima permitida nos pinos ADC não deve exceder ±0,1 V (10, p.22, tradução própria).

Figura 10 - Diagrama de blocos – ADE9078



Fonte: (10)

O diagrama de blocos da figura 10, de forma simples, a trajetória dos sinais desde sua entrada analógica, depois conversão para digital, em seguida a aplicação dos recursos de metrologia que fornecem as variáveis de interesse para o sistema e sua disponibilidade destes dados nas saídas SPI, CF1-CF4. O dispositivo medidor ADE9078 tem uma extensa documentação de modos de operação, detalhamento dos canais de correntes e tensões dentro do chip, relação de registradores. Portanto para o objetivo deste trabalho, uma relação de registrados de correntes, tensões e potências são apresentados nas tabelas 1, 2 e 3:

Tabela 1 - Registradores de Corrente – ADE9078

Table 9. Current Channel Measurement Update Rates

Register Name	Description	Update Rate (kSPS)	Register Name	Description	Update Rate (kSPS)
AI_SINC_DAT	IA sinc4 filter output	16	AIRMS	Filtered based total rms of IA	4
BI_SINC_DAT	IB sinc4 filter output	16	BIRMS	Filtered based total rms of IB	4
CI_SINC_DAT	IC sinc4 filter output	16	CIRMS	Filtered based total rms of IC	4
NI_SINC_DAT	IN sinc4 filter output	16	NIRMS	Filtered based total rms of IN	4
AI_LPF__DAT	IA sinc4 + IIR low-pass filter output and decimation	4	ISUMRMS	Filtered rms of sum of instantaneous currents (AI_PCF + BI_PCF + CI_PCF ± NI_PCF) (see the Neutral Current RMS, RMS of Sum of Instantaneous Currents section)	4
BI_LPF__DAT	IB sinc4 + IIR low-pass filter output and decimation	4			
CI_LPF__DAT	IC sinc4 + IIR low-pass filter output and decimation	4			
NI_LPF__DAT	IN sinc4 + IIR low-pass filter output and decimation	4	IPEAK	Peak current channel sample (see the Peak Detection section)	4
AI_PCF	Instantaneous current on IA	4			
BI_PCF	Instantaneous current on IB	4	ANGLx_x	Voltage to current or current to current phase angle (see the Angle Measurement section)	CLKIN/24 = 512
CI_PCF	Instantaneous current on IC	4			
NI_PCF	Instantaneous current on IN	4			

Fonte: (10)

Tabela 2 - Registradores de Tensão – ADE9078

Table 11. Voltage Channel Measurement Update Rates

Register Name	Description	Update Rate
AV_SINC_DAT	VA sinc4 filter output	16 kSPS
BV_SINC_DAT	VB sinc4 filter output	16 kSPS
CV_SINC_DAT	VC sinc4 filter output	16 kSPS
AV_LPF_DAT	VA sinc4 + IIR low-pass filter and decimator output	f_{DSP} = 4 kSPS
BV_LPF_DAT	VB sinc4 + IIR low-pass filter and decimator output	f_{DSP} = 4 kSPS
CV_LPF_DAT	VC sinc4 + IIR low-pass filter and decimator output	f_{DSP} = 4 kSPS
AV_PCF	Instantaneous voltage on VA	f_{DSP} = 4 kSPS
BV_PCF	Instantaneous voltage on VB	f_{DSP} = 4 kSPS
CV_PCF	Instantaneous voltage on VC	f_{DSP} = 4 kSPS
AVRMS	Filtered based total rms of VA	f_{DSP} = 4 kSPS
BVRMS	Filtered based total rms of VB	f_{DSP} = 4 kSPS
CVRMS	Filtered based total rms of VC	f_{DSP} = 4 kSPS
VPEAK	Peak current channel sample (see the Peak Detection section)	f_{DSP} = 4 kSPS
APERIOD	Line period measurement on VA	f_{DSP} = 4 kSPS
BPERIOD	Line period measurement on VB	f_{DSP} = 4 kSPS
CPERIOD	Line period measurement on VB	f_{DSP} = 4 kSPS
COM_PERIOD	Line period measurement on combined signal from VA, VB, VC (see the Combined Voltage Zero Crossing section)	f_{DSP} = 4 kSPS
ANGLx_x	Voltage to current or current to current phase angle (see the Angle Measurement section)	CLKIN/24 = 512 kSPS

Fonte: (10)

Tabela 3 - Registradores de Potência – ADE9078

Table 12. Active Power Related Register Update Rate

Register Name	Description	Update Rate
AWATT	Low-pass filtered total active power on Phase A	4 kSPS
BWATT	Low-pass filtered total active power on Phase B	4 kSPS
CWATT	Low-pass filtered total active power on Phase C	4 kSPS
AWATT_ACC	Accumulated total active power on Phase A	After the PWR_TIME 4 kSPS samples, from 500 µs to 2.048 sec
BWATT_ACC	Accumulated total active power on Phase B	After the PWR_TIME 4 kSPS samples, from 500 µs to 2.048 sec
CWATT_ACC	Accumulated total active power on Phase C	After the PWR_TIME 4 kSPS samples, from 500 µs to 2.048 sec
AWATTHR	Accumulated total active energy on Phase A	According to the settings in EP_CFG and EP_TIME; holds up to 211 sec of energy at full scale
BWATTHR	Accumulated total active energy on Phase B	According to the settings in EP_CFG and EP_TIME; holds up to 211 sec of energy at full scale
CWATTHR	Accumulated total active energy on Phase C	According to the settings in EP_CFG and EP_TIME; holds up to 211 sec of energy at full scale
APF	Phase A Power Factor (see the Power Factor section)	Every 1.024 sec
BPF	Phase A Power Factor (see the Power Factor section)	Every 1.024 sec
CPF	Phase A Power Factor (see the Power Factor section)	Every 1.024 sec

Fonte: (10)

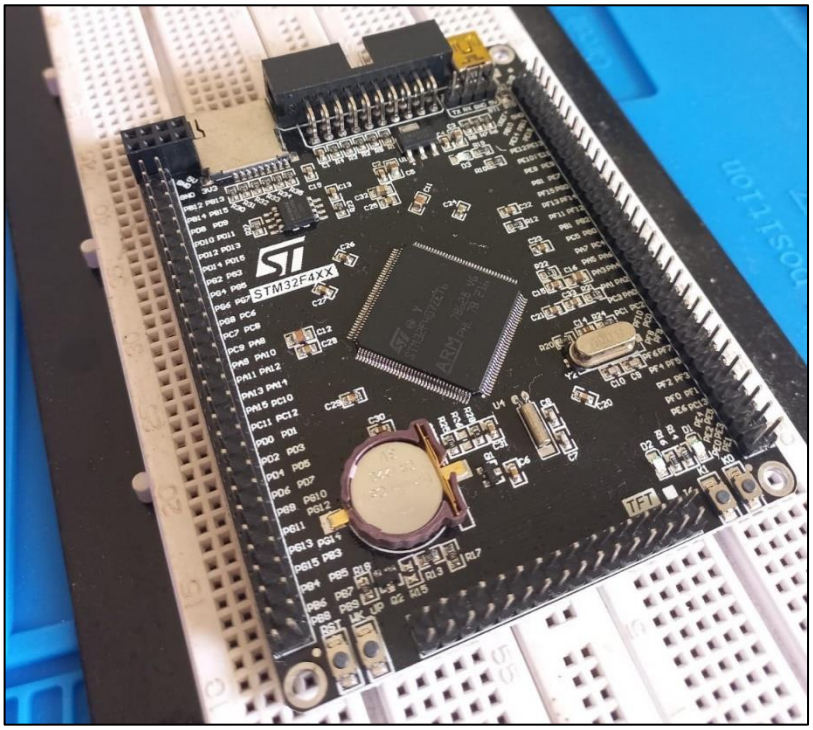
Para maiores informações das tabelas de registradores e os canais das variáveis no detalhe podem ser encontrados no *datasheet* do ADE9078.

3.2.2 STM32F407ZET6

O microcontrolador STM32F407ZET6 da STMicroelectronics com processador Arm 32-bit Cortex-M4 com FPU (unidade de ponto flutuante) e as seguintes características, apresentado na figura 11:

- Clock 168 MHz
- Memória RAM de maior que 1 Mb
- Interfaces CAN, UART, USART, USB, I2C, SPI, Ethernet
- 24 portas A/D de 12 bits
- 60 portas GPIO
- Depuração via SWD e JTAG
- Slot para cartão micro SD
- Dimensões 90 x 60 x 10 mm

Figura 11 - Placa STM32F407ZET6



Fonte: Autoria própria

Este STM32F4 é central para o sistema de monitoramento pois é responsável pela integração dos dispositivos de borda que fazem a interface com a aquisição de dados pelos sensores e com o microcontrolador de envio de dados processados para a nuvem. O dispositivo permite a utilização de sistema operacional de tempo-real, RTOS (real-time operational system) capaz de rotinas de tarefas visando performance, integridade dos dados e tornando a solução escalável para atender mais sistemas de aquisição de dados.

A empresa STMicroeletronics fornece todas as ferramentas de software que contemplam desde a escolha e pré-configuração de suas placas passando pelo desenvolvimento e depuração do código-fonte até a programação do microcontrolador. As ferramentas são:

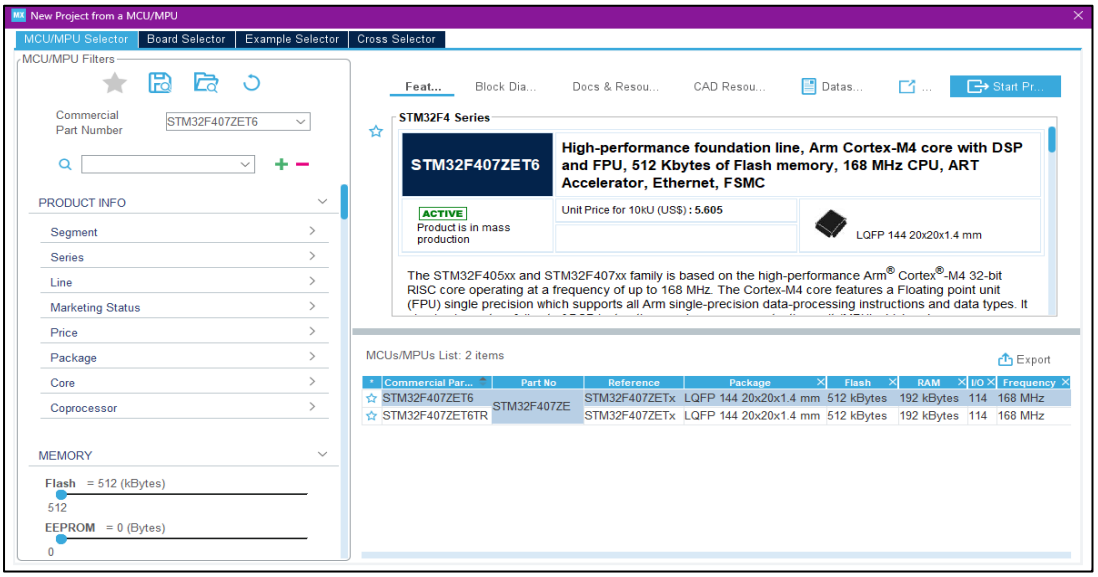
- STMCubeMX
- STMCubeIDE
- STMCubeProgrammer

3.2.2.1 STM32CubeMX

STM32CubeMX é uma ferramenta gráfica que permite uma configuração muito fácil de microcontroladores e microprocessadores STM32, bem como a geração do código C de inicialização correspondente para o núcleo Arm.® Cortex®-M ou uma árvore de dispositivos Linux® parcial para Arm.® Cortex®. A primeira etapa consiste em selecionar um microcontrolador, microprocessador ou plataforma de desenvolvimento STMicroelectronics que corresponda ao conjunto necessário de periféricos, ou um exemplo rodando em uma plataforma de desenvolvimento específica. Para microcontroladores e microprocessadores Arm.® Cortex®-M, a segunda etapa consiste em configurar cada software embarcado necessário graças a um solucionador de conflitos de pinagem, um auxiliar de configuração de árvore de relógio, uma calculadora de consumo de energia e um utilitário que configura os periféricos (como GPIO ou USART) e as pilhas de middleware (como USB ou TCP/IP). As pilhas padrão de software e middleware podem ser estendidas graças aos pacotes de expansão STM32Cube aprimorados (12, tradução própria).

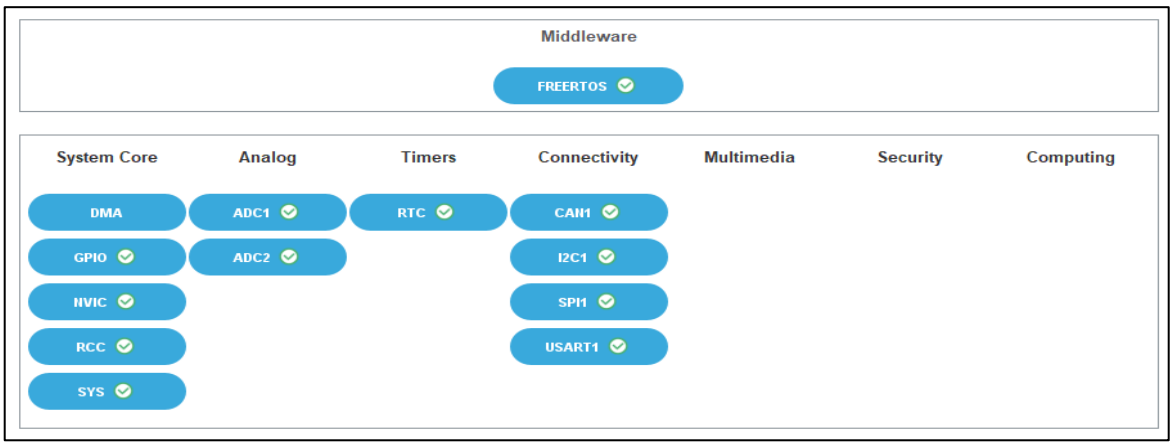
As figuras 12, 13, 14 e 15 a seguir ilustram o processo descrito acima para a criação de um projeto a partir pré-configurações do hardware e a tradução destas através da geração do código correspondente.

Figura 12 - Definição do microcontrolador no CubeMX



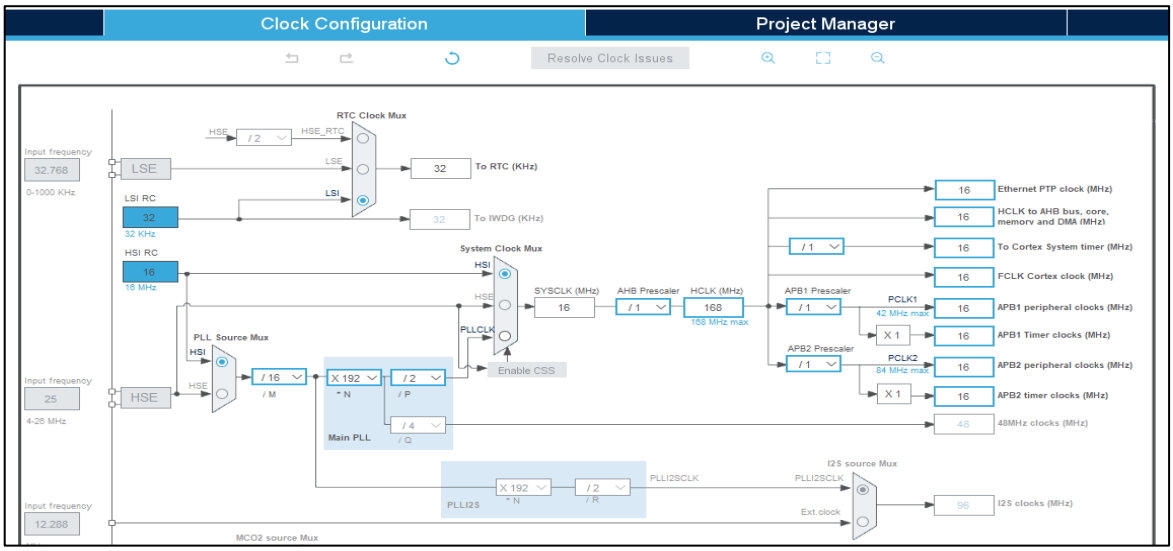
Fonte: Captura de tela do software STM32CubeMX - Autoria própria

Figura 13 - Configurações de portas e recursos STM32F407ZET6



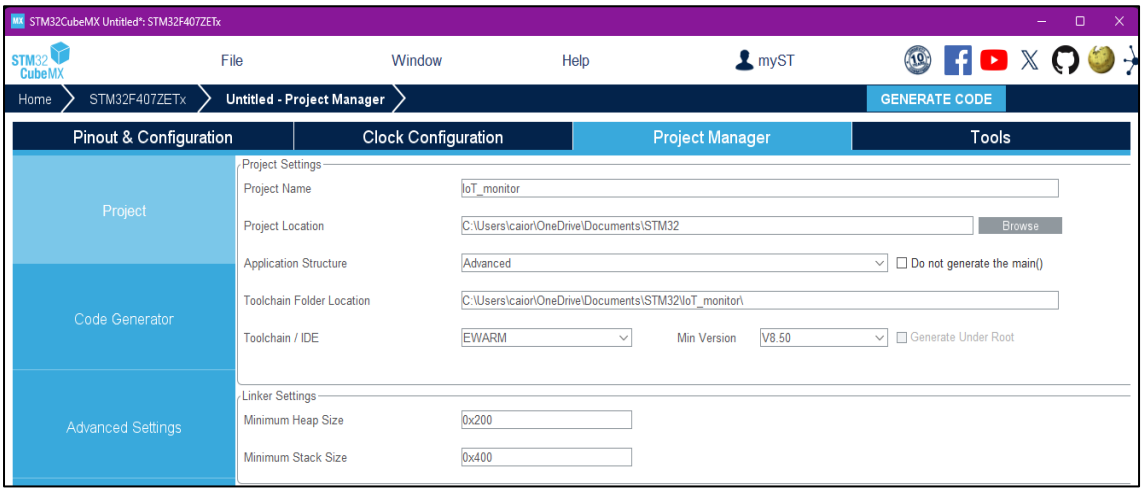
Fonte: Captura de tela do software STM32CubeMX - Autoria própria

Figura 14 - Configurações do Clock – STM32F407ZET6



Fonte: Captura de tela do software STM32CubeMX - Autoria própria

Figura 15 - Configurações de Projeto e Geração de Código – STM32F407ZET6



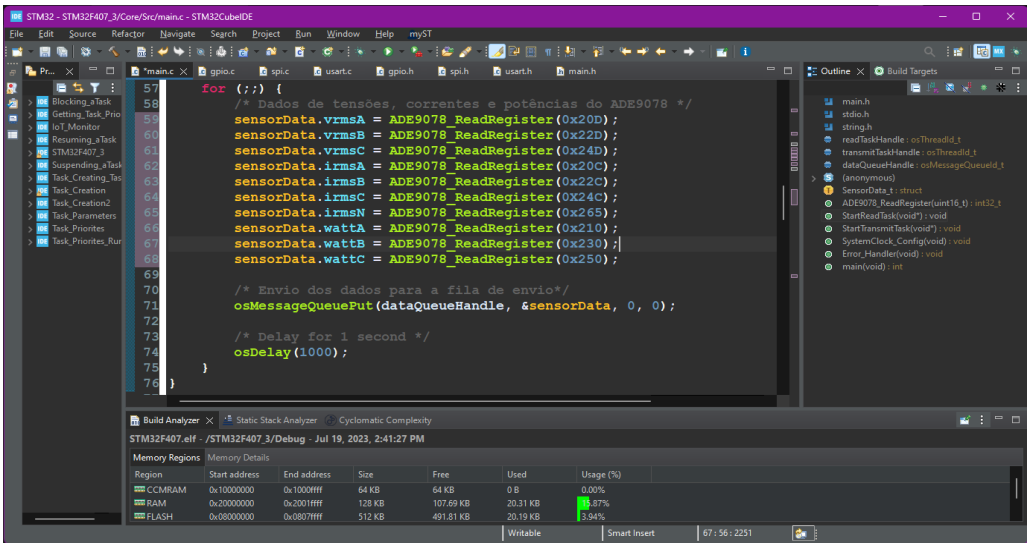
Fonte: Captura de tela do software STM32CubeMX - Autoria própria

3.2.2.1 STM32CubeIDE

> STM32CubeIDE é uma ferramenta de desenvolvimento multi-SO completa, que faz parte do ecossistema de software STM32Cube. STM32CubeIDE é uma plataforma avançada de desenvolvimento C/C++ com configuração de periféricos, geração de código, compilação de código e recursos de depuração para microcontroladores e microprocessadores STM32. É baseado na estrutura Eclipse®/CDT™ e na cadeia de ferramentas GCC para o desenvolvimento e GDB para a depuração. STM32CubeIDE integra funcionalidades de configuração STM32 e criação de projetos do STM32CubeMX para oferecer experiência de ferramenta completa e economizar tempo de instalação e desenvolvimento. Após a seleção de um MCU ou MPU STM32 vazio, ou microcontrolador ou microprocessador pré-configurado a partir da seleção de uma placa ou da seleção de um exemplo, o projeto é criado e o código de inicialização gerado. A qualquer momento durante o desenvolvimento, o usuário pode retornar à inicialização e configuração dos periféricos ou middleware e regenerar o código de inicialização sem impacto no código do usuário. STM32CubeIDE inclui analisadores de construção e pilha que fornecem ao usuário informações úteis sobre o status do projeto e requisitos de memória. Também inclui recursos de depuração padrão e avançados, incluindo visualizações de registros centrais da CPU, memórias e registros periféricos, bem como observação de variáveis ao vivo, interface Serial Wire Viewer ou analisador de falhas. (tradução própria, 14)

Nesta etapa, ilustrada na figura 16, de desenvolvimento é elaborado o código-fonte que permite a integração do ADE9078 e ESP8266 com o STM32F4 utilizando as portas pré-configuradas no STMCubeMX, SPI1 e USART1, respectivamente. A entrada de dados é configurada no código a partir do conhecimento dos registradores do ADE9078 a serem captados e relacionados com suas respectivas variáveis de tensões, correntes e potências de cada fase monitorada pelo sistema. A figura acima ilustra a captação de dados de algumas dessas variáveis fornecidas pelo AFE medidor de energia.

Figura 16 - Registradores ADE9078 - Código-fonte STM32F4



Fonte: Captura de tela do software STM32CubeIDE - Autoria própria

A utilização do FreeRTOS como sistema operacional do microcontrolador significa a divisão de tarefas e criação de rotinas para gerenciar a captação, transformação e envio de dados para o próximo dispositivo. Nesta solução, as tarefas do FreeRTOS separaram a operação do microcontrolador em, basicamente, duas etapas sendo a primeira a entrada de dados e a segunda a formatação e envio dos dados para o ESP8266.

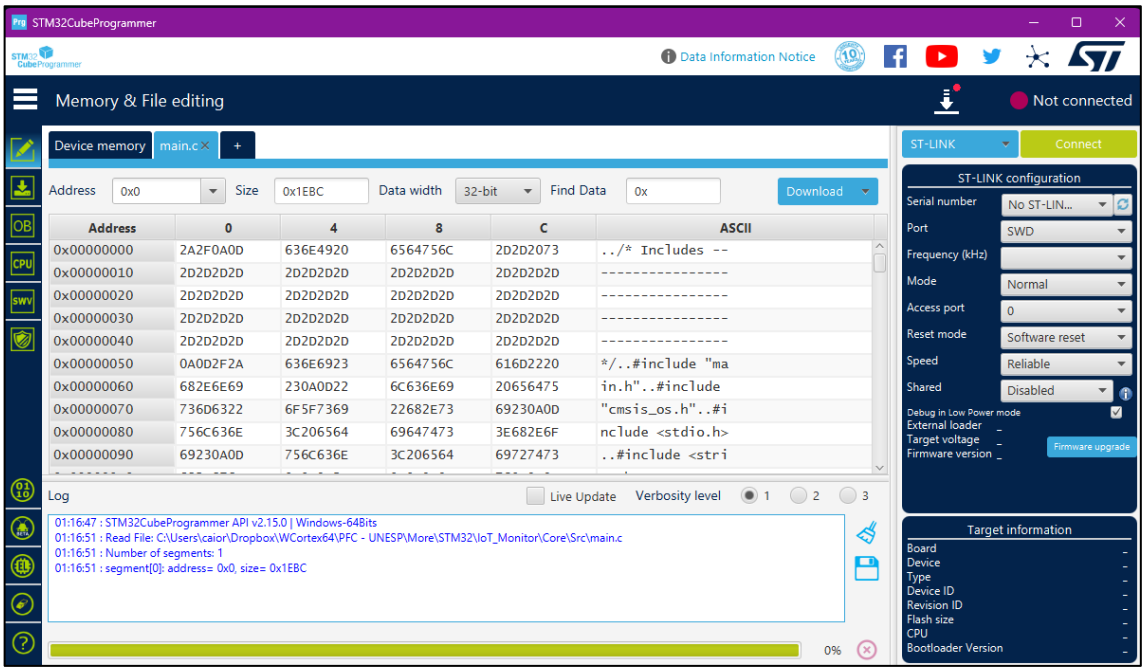
3.2.2.2 STM32CubeProgrammer

STM32CubeProgrammer (STM32CubeProg) é uma ferramenta de software multi-SO completa para programação de produtos STM32. Ele fornece um ambiente fácil de usar e eficiente para leitura, gravação e verificação da memória do dispositivo por meio da interface de depuração (JTAG e SWD) e da interface do bootloader (UART, USB DFU, I2C, SPI e CAN).

STM32CubeProgrammer oferece uma ampla gama de recursos para programar memórias internas STM32 (como Flash, RAM e OTP), bem como memórias externas. Também permite programação e upload de opções, verificação de conteúdo de programação e automação de programação por meio de scripts. STM32CubeProgrammer é entregue em versões GUI (interface gráfica do usuário), presente na figura 17, e CLI (interface de linha de comando).

A programação marca a finalização a última etapa da configuração do microcontrolador. Nesta seção dedicada ao STM32F407ZET6 foi descrita de início ao fim com a utilização das ferramentas do ecossistema STM32Cube, escolha da placa, pré-configuração de portas de comunicação, velocidade de clock, definição de sistema operacional, desenvolvimento, depuração de código fonte e, finalmente, a programação da placa.

Figura 17 - Interface de programação do STM32F4



Fonte: Captura de tela do software STM32CubeIDE - Autoria própria

3.2.3 ESP8266

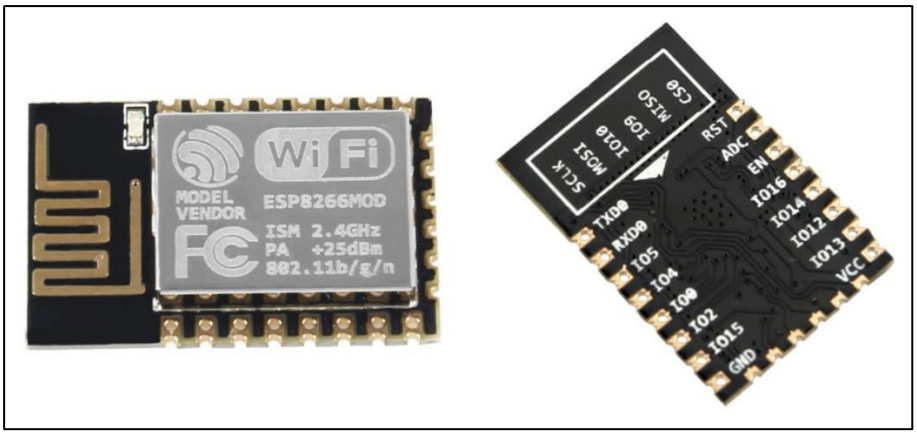
O Módulo ESP8266 é um dispositivo IoT (Internet das Coisas), dispositivo encontrado na figura 18, que consiste de um microprocessador ARM de 32 bits com suporte embutido à rede WIFI e memória flash integrada. Essa arquitetura permite que ele possa ser programado de forma independente, sem a necessidade de outras placas microcontroladoras como o Arduino, por exemplo. Vale ressaltar que o ESP8266 além de ser programado em LUA (linguagem de programação desenvolvida no Brasil), também é compatível com o ambiente de programação (IDE) do Arduino (15).

Especificações:

- Padrões wireless - IEEE 802.11b, IEEE 802.11g, IEEE 802.11n;

- Faixa de frequência - 2,4GHz;
- Antena Embutida;
- Interface - Serial UART (Tx / Rx);
- SPI Flash - 4MB (com possibilidade de expansão para até 16MB);
- Memória RAM - aproximadamente 50 kB;
- Segurança - WEP / WPA / TKIP / AES;
- Tensão de alimentação - 1,8 - 3,3 VDC;
- Conversor A/D - 10 bits ADC e Vin 0 à 1 VDC;
- GPIO - 11 portas (15)

Figura 18 - Módulo Wi-Fi – ESP8266



Fonte: (15)

Este é o último dispositivo de borda do sistema de monitoramento, sendo ele a via de comunicação da borda com o a nuvem. O ESP8266 recebe os dados pelas portas UART(Tx/Rx) e transmite para a nuvem da AWS via protocolo MQTT de forma leve, eficiente e segura.

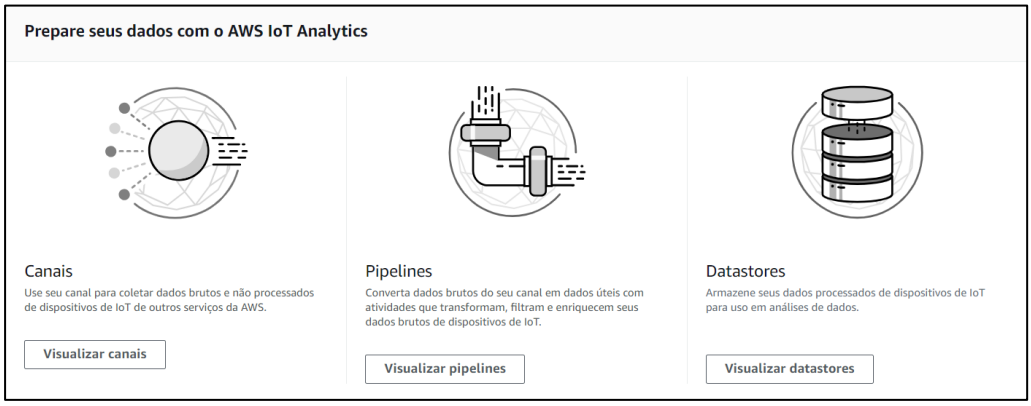
O código-fonte é desenvolvido com a ferramenta ArduinoIDE que oferece ampla diversidade de pacotes que suportam placas de microcontroladores, inclusive para o modelo de ESP8266. Portanto, esta última etapa até a nuvem consiste em retransmitir os dados já processados e formatados por todo o restante do sistema.

O broker(servidor) que recebe estes dados dentro da nuvem da AWS é o IoT Core onde é possível conectar dispositivos de borda, criar fluxos de processamento e armazenamento de dados.

3.3 Armazenamento de Dados

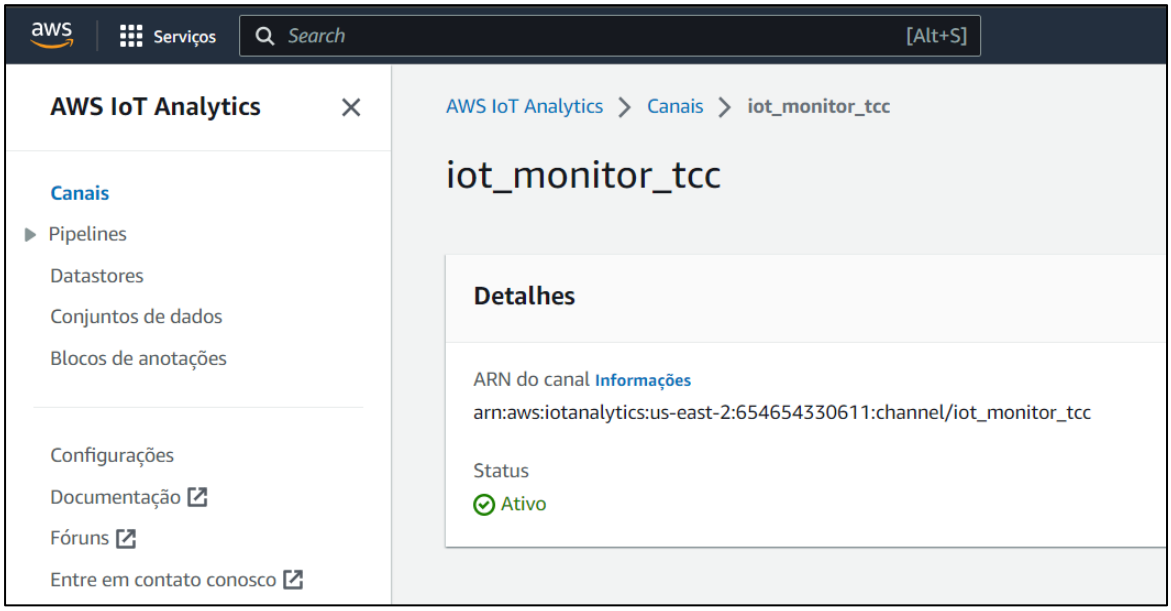
Etapa final do sistema de monitoramento que descreve uma solução de armazenar os dados dentro de tabelas da plataforma AWS a partir do fluxo de dados desenvolvido até a chegada dos dados via MQTT utilizando a ferramenta IoT Core. A partir deste ponto é ilustrada a forma de criar um fluxo de dados dentro da plataforma AWS utilizando o IoT Analytics com o direto armazenamento dos dados em banco de dados, os datastores. As figuras 19, 20, 21 e 22 abaixo ilustram este processo.

Figura 19 - Plataforma AWS – Painel IoT Analytics



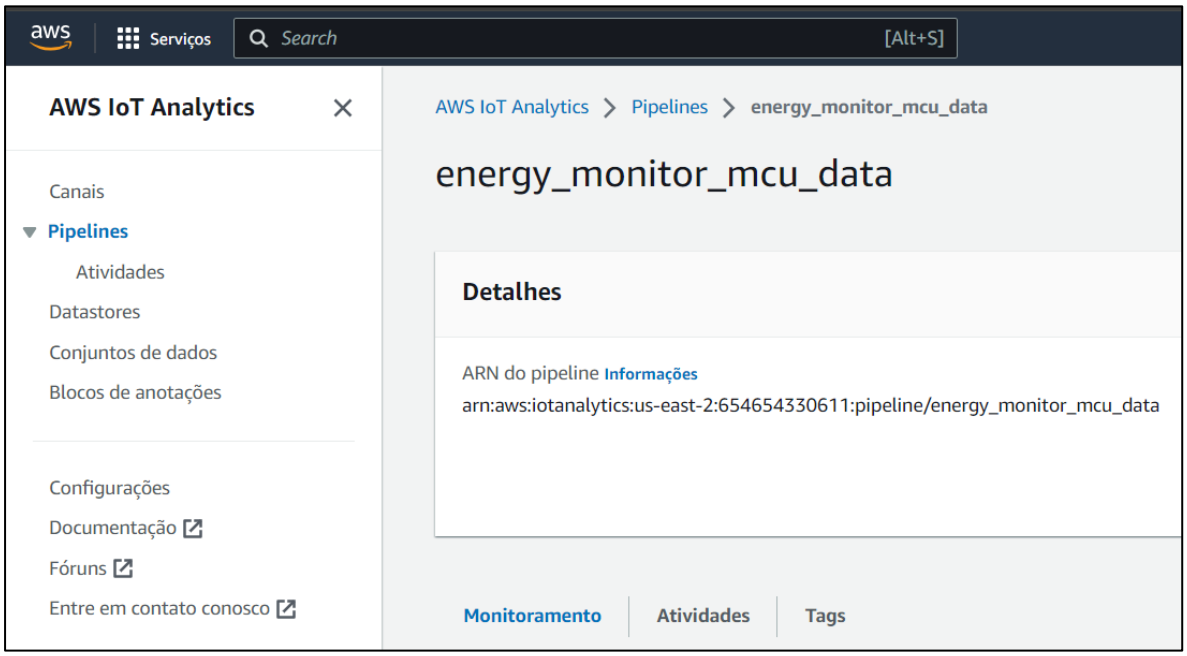
Fonte: Captura de tela da plataforma AWS (16) - Autoria própria

Figura 20 - IoT Analytics – Canais de dados do Monitor IoT



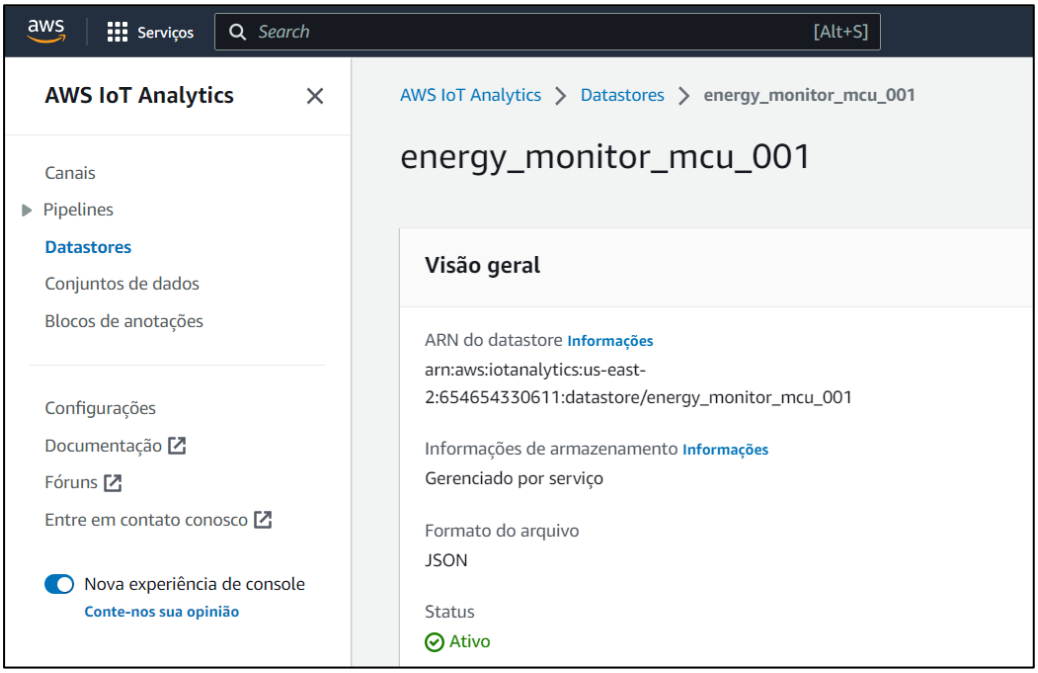
Fonte: Captura de tela da plataforma AWS (16) - Autoria própria

Figura 21 - IoT Analytics – Pipeline de dados IoT Monitor



Fonte: Captura de tela da plataforma AWS (16) - Autoria própria

Figura 22 - IoT Analytics – Datastore IoT Monitor



Fonte: Captura de tela da plataforma AWS (16) - Autoria própria

O ecossistema da AWS fornece uma grande quantidade de recursos para configurar o fluxo de dados vindos dos dispositivos IoT para bancos de dados na nuvem. Escolha do tipo

de banco de dados, disponibilidade, escalabilidade e custo para operação conforme a demanda e criticidade da operação do sistema embarcado em relação ao armazenamento dos dados.

Este fluxo foi empregado pois os dados são os valores resultantes da computação em borda feita pelo ADE9078 que transforma as leituras dos sensores nos valores prontos para o armazenamento, não sendo necessária mais nenhuma manipulação dentro da plataforma AWS, evitando geração de custo de computação em nuvem.

4 RESULTADOS E DISCUSSÕES

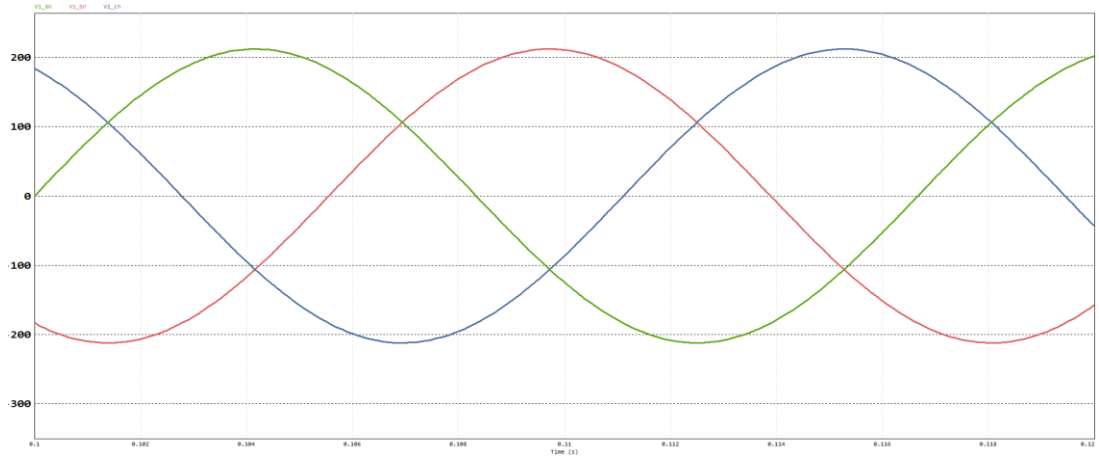
Nesta seção são apresentados os resultados da utilização das ferramentas de desenvolvimento e de simulação o quais demostram os sinais referentes a operação da máquina elétrica e os sinais produzidos pelos sensores que são transmitidos para o medidor de energia.

Os códigos de programação que estabelecem a integração entre os dispositivos de borda e a nuvem estão presentes nos apêndices A e B.

Os resultados da simulação da operação da máquina elétrica trazem as correntes e tensões do sistema trifásico captados pelos sensores acoplados e seus respectivos sinais de saída. A simulação foi parametrizada para gerar valores de corrente e tensão próximos aos limites do transformador de corrente e do sensor de tensão.

Com isto, foram obtidos os resultados apresentados nas figuras a seguir, as formas de ondas dos sinais de entrada da máquina elétrica e sinais de saída dos sensores.

Figura 23 - Formas de onda das tensões de entrada



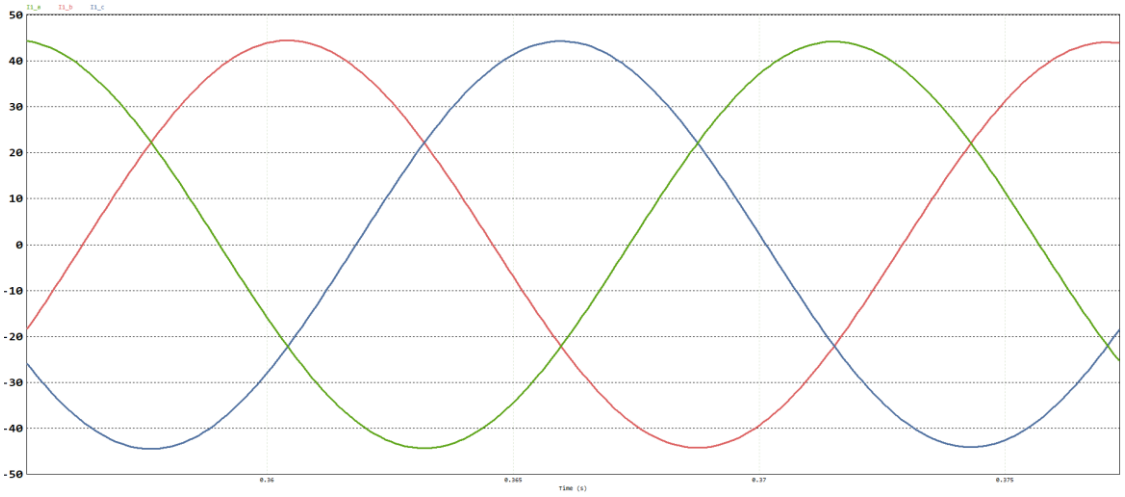
Fonte: Captura de tela do software PSIM (17) - Autoria própria

Tabela 4 - Valores máximos e mínimos dos sinais de entrada da máquina

Sinal Máquina	Máximo	Mínimo	RMS
Tensão A (V)	212,29	-212,29	149,87
Tensão B (V)	212,26	-212,22	149,81
Tensão C (V)	212,19	-212,25	149,80
Corrente A (A)	48,95	-48,36	34,34
Corrente B (A)	48,56	-48,82	34,15
Corrente C (A)	48,37	-48,83	34,30

Fonte: Autoria própria

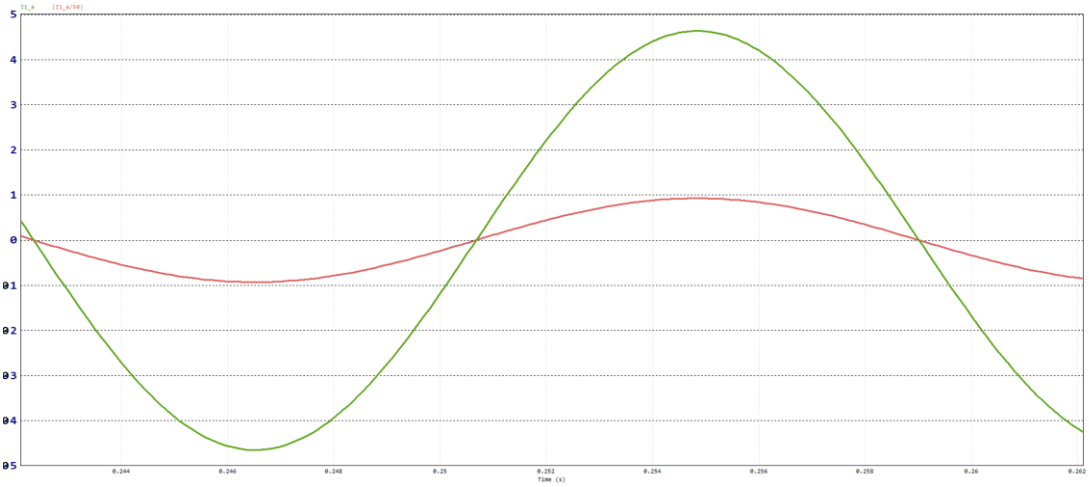
Figura 24 - Formas de onda das correntes de entrada



Fonte: Captura de tela do software PSIM (17) - Autoria própria

A próxima figura traz as duas formas de ondas em que se associam através do sensor transformador de corrente. É possível notar o intervalo de valores do sinal de saída do sensor fica dentro do valor nominal do sensor, de ±1V, para uma das fases.

Figura 25 - Forma de ondas dos sinais de corrente de entrada e de saída do sensor de corrente



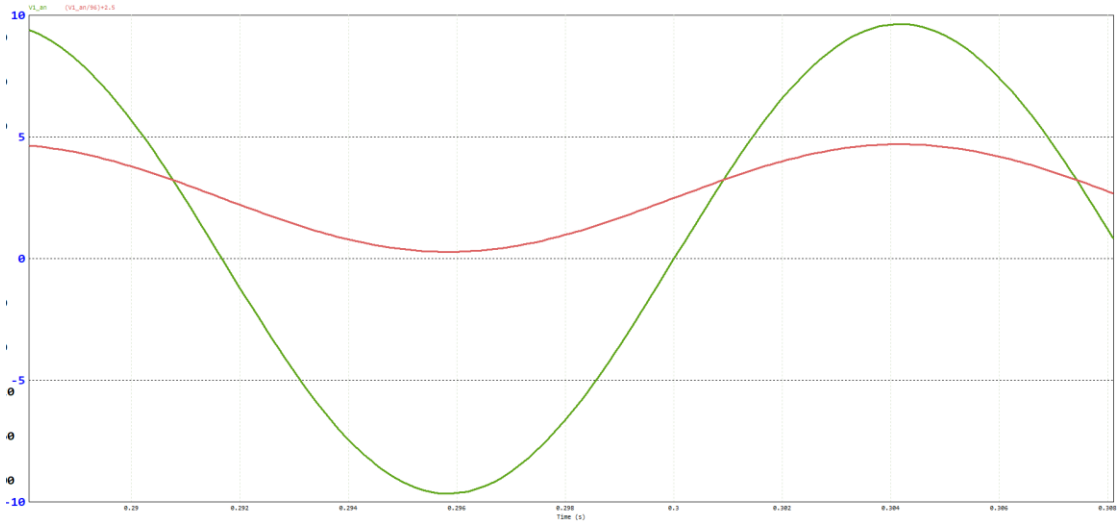
Fonte: Captura de tela do software PSIM (17) - Autoria própria

Tabela 5 - Valores máximos e mínimos dos sinais de saídas dos sensores

Sinal Sensor	Máximo	Mínimo
Tensão A (V) – CC	4,71	0,29
Tensão B (V) – CC	4,71	0,29
Tensão C (V) – CC	4,71	0,29
Corrente A (V) – CA	0,98	-0,97
Corrente B (V) – CA	0,97	-0,98
Corrente C (V) – CA	0,97	-0,98

Fonte: Autoria própria

Figura 26 - Forma de ondas dos sinais de tensão de entrada e saída do sensor de tensão



Fonte: Captura de tela do software PSIM (17) - Autoria própria

A figura acima traz as formas de onda o sinal da tensão na máquina e a tensão de saída do sensor de tensão, dentro do intervalo de valor máximo para o dispositivo que fica de 0 a 5V. A tabela 5 traz todos dos valores relacionados aos sinais de saída dos sensores para cada fase da máquina.

A última tabela apresenta os valores de potência RMS para cada uma das fases dentro do intervalo de tempo das capturas das figuras acima.

Tabela 6 - Valores de potência RMS da máquina elétrica por fase

Potência Fases	**RMS**
Potência A (W)	5084,36
Potência B (W)	5030,20
Potência C (W)	5080,18

Fonte: Autoria própria

Portanto, os resultados apresentados respeitam os valores máximos de tensão e corrente para os sensores usados no sistema de aquisição de dados proposto. É notável que o sensor de corrente está no limite de sua capacidade enquanto o sensor de tensão ainda possui margem até atingir o seu valor máximo de 250 V. Nessa condição, a potência total consumida pela máquina passa de 15 KW, em torno de 1 terço da potência máxima descrita.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 Conclusão

Este trabalho apresentou a descrição e desenvolvimento de um sistema de monitoramento de máquina elétrica utilizando as tecnologias de Internet da Coisas para coletar, processar e armazenar dados a partir das leituras das variáveis elétricas, fornecendo informações do comportamento elétrico para análise de desempenho e eficiência.

O sistema elaborado emprega sensores de corrente e tensão para coleta de sinais analógicos integrado com um medidor de energia responsável por calcular parâmetros elétricos de correntes, tensões e potências associadas a operação da máquina. O sistema de aquisição de dados é integrado a microcontroladores com função de transmitir os dados processados para uma aplicação em nuvem, fornecendo informações para futuras análises de desempenho e eficiência da máquina elétrica.

O sistema proposto como solução para o monitoramento descreveu as etapas necessárias para a leitura dos sinais analógicos vindos dos sensores em contato com a alimentação da máquina, da necessidade do condicionamento do sinal para ser entregue ao medidor de energia ADE9078 e a sua integração com os microcontroladores através do desenvolvimento de código-fonte para estabelecer comunicação e fluxo de dados com destino a aplicação em nuvem da AWS.

Portanto, o sistema tem capacidade de ser implementado utilizando os dispositivos empregados na solução e integrados através da programação dos microcontroladores para que as informações sejam enviadas a nuvem.

5.2 Trabalhos Futuros

- Inserir módulo de alimentação independe utilizando baterias para não ocorrer o desligamento do sistema de monitoramento em caso de queda energia.
- Adicionar datalogger e memória para caso de longos períodos sem conectividade com a nuvem com intuito de não ter perda de dados.
- Emprego de outras tecnologias de protocolos de conectividade de IoT como LoraWan e Zigbee.
- Utilização de outros medidores de energia da classe do ADE9078 para máquinas ou dispositivos monofásicos e bifásicos.
- Inserir tela LCD para interface gráfica com o sistema de borda para sinalização de alarmes de parâmetros elétricos da máquina, status de conectividade e leitura dos sensores.
- Utilização das ferramentas da AWS para análise e elaboração de indicadores de desempenho, eficiência e manutenção.
- Desenvolvimento de dashboard para que permita acompanhar a operação das máquinas em tempo-real.

REFERÊNCIAS

[1] BERNAL, J.; SRIDHAR, B. **Industrial IoT for Architects and Engineers:** Architecting secure, robust, and scalable industrial IoT solutions with AWS. PACKT. Birmingham, Reino Unido. 2023.

[2] EMILIO, M DI PAOLO. **Data Acquisition Systems:** From Fundamentals to Applied Design. SPRINGER. London, United Kingdom - 2013

[3] BERNSTEIN, H. **Measuring Electronics and Sensors:** Basics of Measurement Technology, Sensors, Analog and Digital Signal Processing. SPRINGER. London, United Kingdom. 2014

[4] KHANDAI, Sandeep Kumar; JAIN, Sachin Kumar. Comparison of sensors performance for the development of wrist pulse acquisition system. *In:* TENCON 2017 - 2017 IEEE REGION 10 CONFERENCE, **Proceedings** [...]. [Piscataway]: IEEE, 2017, Penang, Malaysia. 2017, pp. 2870-2875.

[5] BUTUN, **I. Industrial IoT:** Challenges, Design Principles, Applications, and Security – SPRINGER. Guttenberg, Suécia. 2020

[6] MARINESCU, D. C. **Cloud Computing:** Theory and Practice 2 ed. MORGAN KAUFMAN. Cambridge, Estados Unidos

[7] HILLAR, G. C. **MQTT Essentials:** A Lightweight IoT Protocol – PACKT PUBLISHING. Birmingham, Reino Unido

[8] GUERMANDI, V.I. **Inteligência de negócio aplicado à análise de processos da indústria 4.0 através do software PowerBI**. 2022. 78 f. Trabalho de Graduação (Bacharelado em Engenharia de Controle e Automação) - Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista, Sorocaba, 2022.

[9] MASTERWALKER SHOP. **Sensor de tensão ZMPT101B – Transformador de tensão** [*S. I.*], [202-?]. Disponível em: <https://www.masterwalkershop.com.br/sensor-de-tensao-ac-0-a-250vac-zmpt101b>. Acesso em - 14 de maio de 2024.

[10] ANALOG DEVICES. **Data Sheet - ADE9078:** manual. 2016. Disponível em: <https://www.analog.com/media/en/technical-documentation/data-sheets/ade9078.pdf>. Acesso em - 9 de outubro de 2023.

[11] FERMAC. **SENSOR DE CORRENTE NÃO-VASIVO 50A SCT-013.** [*S. I.*], [202-?]. Disponível em: <https://www.fermarc.com/produto/sensor-de-corrente-nao-invasivo-50a-sct-013.html>. Acesso em - 14 de maio de 2024.

[12] STMICROELECTRONICS. **STM32Cube initialization code generator.** [*S. I.*], [201-?]. Disponível em: https://www.st.com/en/development-tools/stm32cubemx.html />. Acesso em 20 de maio de 2024.

[13] STMICROELECTRONICS. **Integrated Development Environment for STM32.** [*S. I.*], [201-?]. Disponível em: <https://www.st.com/en/development-tools/stm32cubeide.html>. Acesso em 20 de maio de 2024.

[14] MASTERWALKER SHOP. **Módulo WiFi ESP8266 ESP-14.** [*S. I.*], [202-?]. Disponível em: https://www.masterwalkershop.com.br/modulo-wifi-esp8266-esp-14. Acesso em: 12 maio 2024.

[15] MASTERWALKER SHOP. **Módulo Wi-Fi ESP 8266 – ESP 12E.** [*S. I.*], [202-?]. Disponível em: <https://curtocircuito.com.br/modulo-wifi-esp8266-esp-12e.html>. Acesso em 12 de maio de 2024.

[16] AMAZON WEB SERVICES INC. **IoT Analytics.** AWS, 2024 <https://us-east-2.console.aws.amazon.com/console/home?region=us-east-2>. Acesso em 11 de maio de 2024.

[17] ALTAIR ENGINEERING INC. **PSIM Professional.** 2024.0.0.2471**.** PSIM LLC. 2024.

APÊNDICE A – Código-Fonte STM32F4

1. main.c

```
* Includes ------------------------------------------------------------------*/
#include "main.h"
#include <stdio.h>
#include <string.h>

/* Handles for FreeRTOS tasks and message queue */
osThreadId_t readTaskHandle;
osThreadId_t transmitTaskHandle;
osMessageQueueId_t dataQueueHandle;

/* Estrutura de dado - sensores */
typedef struct {
    int32_t vrmsA;
    int32_t vrmsB;
    int32_t vrmsC;
    int32_t irmsA;
    int32_t irmsB;
    int32_t irmsC;
    int32_t irmsN;
    int32_t wattA;
    int32_t wattB;
    int32_t wattC;
} SensorData_t;
```

```
/* Função para ler dados - ADE9078 via SPI */
int32_t ADE9078_ReadRegister(uint16_t regAddress) {
    uint8_t txData[3]; // Data to send - address and a dummy byte
    uint8_t rxData[3]; // Data to receive
    int32_t result = 0;

    txData[0] = (regAddress >> 8) & 0xFF;
    txData[1] = regAddress & 0xFF;
    txData[2] = 0x00;
    /* Selecionando  ADE9078 definindo CS LOW */
    HAL_GPIO_WritePin(ADE9078_CS_PORT, ADE9078_CS_PIN, GPIO_PIN_RESET);
    /* Envio para o register address */
    HAL_SPI_Transmit(&hspi1, txData, 2, HAL_MAX_DELAY);
    /* Receber register value */
    HAL_SPI_Receive(&hspi1, rxData, 3, HAL_MAX_DELAY);
    /* Deselecionando  ADE9078 definindo CS HIGH */
    HAL_GPIO_WritePin(ADE9078_CS_PORT, ADE9078_CS_PIN, GPIO_PIN_SET);

    /* Combinando os dados*/
    result = (rxData[0] << 16) | (rxData[1] << 8) | rxData[2];

    if (result & 0x800000) {
        result |= 0xFF000000;
    }

    return result;
}
```

```
/* FreeRTOS task para ler os dados do ADE9078 */
void StartReadTask(void *argument) {
    SensorData_t sensorData;

    for (;;) {
        /* Dados de tensões, correntes e potências do ADE9078 */
        sensorData.vrmsA = ADE9078_ReadRegister(0x20D);
        sensorData.vrmsB = ADE9078_ReadRegister(0x22D);
        sensorData.vrmsC = ADE9078_ReadRegister(0x24D);
        sensorData.irmsA = ADE9078_ReadRegister(0x20C);
        sensorData.irmsB = ADE9078_ReadRegister(0x22C);
        sensorData.irmsC = ADE9078_ReadRegister(0X24C);
        sensorData.irmsN = ADE9078_ReadRegister(0x265);
        sensorData.wattA = ADE9078_ReadRegister(0x210);
        sensorData.wattB = ADE9078_ReadRegister(0x230);
        sensorData.wattC = ADE9078_ReadRegister(0x250);

        /* Envio dos dados para a fila de envio*/
        osMessageQueuePut(dataQueueHandle, &sensorData, 0, 0);

        /* Delay for 1 second */
        osDelay(1000);
    }
}

/* FreeRTOS task para envio de dados via UART */
void StartTransmitTask(void *argument) {
```

```
    SensorData_t sensorData;

    for (;;) {
        /* Esperando a mensagem vinda da fila */
        if (osMessageQueueGet(dataQueueHandle, &sensorData, NULL, osWaitForever) ==
osOK) {
            char msg[256];
            /* Formatação dos dados para o envio */
            snprintf(msg, sizeof(msg),
                    "{\"VRMS_A\" - %ld, \"VRMS_B\" - %ld, \"VRMS_C\" - %ld,"
                    " \"IRMS_A\" - %ld, \"IRMS_B\" - %ld, \"IRMS_C\" - %ld, \"IRMS_N\" - %ld, "
                    "\"Watt_A\" - %ld, \"Watt_B\" - %ld, \"Watt_C\" - %ld}\r\n",
                    sensorData.vrmsA, sensorData.vrmsB, sensorData.vrmsC, sensorData.irmsA,
                                        sensorData.irmsB, sensorData.irmsC,
sensorData.irmsN,
                    sensorData.wattA, sensorData.wattB, sensorData.wattC);

            /* Transmissão via UART */
            HAL_UART_Transmit(&huart1, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
        }
    }
}

/* Function to configure the system clock */
void SystemClock_Config(void) {
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /* Configure the main internal regulator output voltage */
```

```
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /* Init RCC Oscillators */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 336;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 7;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
    Error_Handler();
  }

  /* Initialize the CPU, AHB and APB buses clocks */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK |
RCC_CLOCKTYPE_SYSCLK
                        | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK) {
    Error_Handler();
  }
```

```
}

/* Error handler function */
void Error_Handler(void) {
    while (1) {
        // Stay here
    }
}

/* Main function */
int main(void) {
    /* Initialize the HAL Library */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_SPI1_Init();
    MX_USART1_UART_Init();

    /* Initialize the kernel */
    osKernelInitialize();

    /* Create the queue for sensor data */
    dataQueueHandle = osMessageQueueNew(10, sizeof(SensorData_t), NULL);
```

```
    /* Create the tasks */
    readTaskHandle = osThreadNew(StartReadTask, NULL, NULL);
    transmitTaskHandle = osThreadNew(StartTransmitTask, NULL, NULL);

    /* Start the scheduler */
    osKernelStart();

    /* Infinite loop */
    while (1) {
    }
}
```

2. main.h

```
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"
#include "cmsis_os.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"

/* Defines -------------------------------------------------------------------*/
#define ADE9078_CS_PIN GPIO_PIN_4
#define ADE9078_CS_PORT GPIOA

/* Function Prototypes -------------------------------------------------------*/
int32_t ADE9078_ReadRegister(uint16_t regAddress);
void StartReadTask(void *argument);
void StartTransmitTask(void *argument);
```

3. gpio.c

```
/* Includes ------------------------------------------------------------------*/
#include "gpio.h"

/* Configure GPIO */
void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    HAL_GPIO_WritePin(GPIOA, ADE9078_CS_PIN, GPIO_PIN_RESET);

    GPIO_InitStruct.Pin = ADE9078_CS_PIN;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

4. gpio.h

```
#ifndef __GPIO_H
#define __GPIO_H

#ifdef __cplusplus
extern "C" {
```

```
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* GPIO Ports */
#define ADE9078_CS_PIN GPIO_PIN_4
#define ADE9078_CS_PORT GPIOA

/* Function prototypes -----------------------------------------------*/
void MX_GPIO_Init(void);

#ifdef __cplusplus
}
#endif

#endif /* __GPIO_H */
```

5. spi.c

```
/* Includes ------------------------------------------------------------------*/
#include "spi.h"

/* SPI1 init function */
void MX_SPI1_Init(void)
{
  /* Parâmetros de Configuração */
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
```

```
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_16;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
}

void HAL_SPI_MspInit(SPI_HandleTypeDef* spiHandle)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(spiHandle->Instance==SPI1)
  {
    /* SPI1 clock ativação */
    __HAL_RCC_SPI1_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**SPI1 GPIO Configuração
    PA5     ------> SPI1_SCK
    PA6     ------> SPI1_MISO
    PA7     ------> SPI1_MOSI
```

```
    */
    GPIO_InitStruct.Pin = GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  }
}
```

6. spi.h

```
#ifndef __SPI_H
#define __SPI_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* Function prototypes -------------------------------------------------------*/
void MX_SPI1_Init(void);

#ifdef __cplusplus
}
#endif

#endif /* __SPI_H */
```

7. usart.c

```
/* Includes ------------------------------------------------------------------*/
#include "usart.h"

/* UART1 init function */
void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
}

void HAL_UART_MspInit(UART_HandleTypeDef* uartHandle)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(uartHandle->Instance==USART1)
  {
    /* USART1 clock ativação */
```

```
    __HAL_RCC_USART1_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**USART1 GPIO Configuração
    PA9     ------> USART1_TX
    PA10     ------> USART1_RX
    */
    GPIO_InitStruct.Pin = GPIO_PIN_9|GPIO_PIN_10;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF7_USART1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
  }
}
```

8. usart.h

```
#ifndef __USART_H
#define __USART_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* Function prototypes -----------------------------------------------*/
void MX_USART1_UART_Init(void);
```

```
#ifdef __cplusplus
}
#endif

#endif /* __USART_H */
```

APÊNDICE B – Código-Fonte ESP8266

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

const char* ssid = "nome_da_conexao_SSID";
const char* password = "senha_PASSWORD";
const char* mqttServer = "your-aws-endpoint.iot.your-region.amazonaws.com";
const int mqttPort = 8883;
const char* mqttUser = "IOT_MQTTusername";
const char* mqttPassword = "IOT_MQTTpassword";
const char* ca_cert = \
"-----BEGIN CERTIFICATE-----\n" \
"YOUR_CA_CERTIFICATE\n" \
"-----END CERTIFICATE-----\n";
const char* client_cert = \
"-----BEGIN CERTIFICATE-----\n" \
"YOUR_CLIENT_CERTIFICATE\n" \
"-----END CERTIFICATE-----\n";
const char* client_key = \
"-----BEGIN PRIVATE KEY-----\n" \
"YOUR_CLIENT_PRIVATE_KEY\n" \
"-----END PRIVATE KEY-----\n";

WiFiClientSecure espClient;
PubSubClient client(espClient);
```

```
void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }

    Serial.println("Connected to WiFi");

    espClient.setCACert(ca_cert);
    espClient.setCertificate(client_cert);
    espClient.setPrivateKey(client_key);

    client.setServer(mqttServer, mqttPort);

    while (!client.connected()) {
        Serial.println("Connecting to MQTT...");

        if (client.connect("ESP8266Client", mqttUser, mqttPassword)) {
            Serial.println("Connected");
        } else {
            Serial.print("Failed with state ");
            Serial.print(client.state());
            delay(2000);
```

```
        }
    }
}

void loop() {
    if (Serial.available()) {
        String message = Serial.readStringUntil('\n');
        Serial.print("Received message - ");
        Serial.println(message);

        if (client.publish("AWS topic", message.c_str())) {
            Serial.println("Message sent successfully");
        } else {
            Serial.println("Error sending message");
        }
    }

    client.loop();
}
```