Universidade Federal do ABC
Pós-Graduação em Tecnologias e Sistemas de Informação

Conectividade e Monitoramento IoT: Aplicação do Protocolo MQTT em Microcontroladores ESP32 e Raspberry para Supervisão de Sensores Analógicos e Digitais

MARCELO FERNANDES DE BARROS

SANTO ANDRÉ - SP, Setembro de 2024

MARCELO FERNANDES DE BARROS

Conectividade e Monitoramento IoT: Aplicação do Protocolo MQTT em Microcontroladores ESP32 e Raspberry para Supervisão de Sensores Analógicos e Digitais

Trabalho de Conclusão de Curso apresentado ao curso de pós-graduação em TSI - Tecnologia e Sistemas da Informação, da Universidade Federal do ABC, como requisito parcial para obtenção do diploma no curso de Tecnologias e Sistemas de Informação.

Universidade Federal do ABC – UFABC

Pós-Graduação em Tecnologias e Sistemas de Informação

Orientador: Profº Dr. Mario Alexandre Gazziro

SANTO ANDRÉ - SP

Setembro de 2024

Lista de ilustrações

Figura 1 – Diagrama em blocos da rede MQTT a ser montada, com os principais componentes. Fonte: Próprio autor 6
Figura 2 – Sensor DHT22. Disponível em https://components101.com/sensors/dht22-pinout-specs-datasheet. Acesso em: 15/05/2024. 7
Figura 3 – Sensor reed switch. Disponível em https://instrumentationtools.com/reed-switch-working-principle. Acesso em: 15/05/2024. 8
Figura 4 – Sensor reed switch. Disponível em https://altronics.cl/sensor-magnetico-mc-38. Acesso em: 20/05/2024. 8
Figura 5 – Sensor reed switch. Disponível em https://www.eletrolico.com.br/reles-eletromecanicos/rele-miniat-5a-5vcc-2-rev-pci. Acesso em: 21/05/2024. 9
Figura 6 – Telas de configuração do Narada MQTT Broker. Disponível em https://play.google.com/ Acesso em: 22/05/2024. 11
Figura 7 – Telas de configuração do MQTT Dashboad. Disponível em https://www.gameloop.com/ Acesso em: 25/05/2024. 12
Figura 8 – Esquema elétrico do cliente MQTT que será controlado pelo ESP32. Fonte: Próprio autor 13
Figura 9 – Esquema elétrico do cliente MQTT que será controlado pelo RASPBERRY PICO W. Fonte: Próprio autor 13
Figura 10 – Diagrama da rede MQTT mostrando em detalhes a comunicação entre os componentes conectados. Fonte: Próprio autor 14
Figura 11 – Protótipo utilizando ESP32. Fonte: Próprio autor 15
Figura 12 – Protótipo utilizando RASPBERRY PICO W. Fonte: Próprio autor . . 15
Figura 13 – Broker MQTT em operação. Fonte: Próprio autor 16
Figura 14 – Dashboard em operação. Fonte: Próprio autor 17
Figura 15 – Acionamento do ícone no dashboard responsável por acionar o relé do prototipo RASPBERRY e a resposta com a energização do led. Fonte: Próprio autor 17
Figura 16 – Acionamento do ícone no dashboard responsável por acionar o relé do prototipo ESP32 e a resposta com a energização do led. Fonte: Próprio autor 17
Figura 17 – Teste do reed switch no protótipo RASPBERRY e respectivo ícone no dashboard. Fonte: Próprio autor 18
Figura 18 – Teste do reed switch no protótipo ESP32 e respectivo ícone no dashboard. Fonte: Próprio autor 19

Lista de tabelas

Sumário

Lista de ilustrações **2**

Lista de tabelas **2**

Sumário **1**

1 INTRODUÇÃO **2**

1.1 Justificativa **4**

2 METODOLOGIA **6**

2.1 Componentes **7**

2.1.1 Sensor DHT22 7

2.1.2 Sensor reed switch 8

2.1.3 Relés Eletromecânicos 9

2.1.4 Microcontroladores 9

2.2 Aplicativos para smartphone **10**

2.2.1 Narada MQTT Broker 10

2.2.2 MQTT Dashboard 11

2.3 Esquema elétrico **12**

3 RESULTADOS **15**

4 DISCUSSÃO **20**

5 CONCLUSÃO **21**

A CÓDIGO FONTE PARA O RASPBERRY **22**

B CÓDIGO FONTE PARA O ESP32 . . . . . **25**

REFERÊNCIAS **28**

1 Introdução

A tecnologia está mudando rapidamente a maneira como interagimos com o mundo à nossa volta. Visando atender às mais novas demandas dos consumidores, empresas estão desenvolvendo hoje produtos com interfaces tecnológicas que seriam inimagináveis há uma década. (MAGRANI, 2018)

Isto envolve a interconexão de dispositivos inteligentes, permitindo a coleta, processamento e troca de informações em tempo real. Nesse contexto, a conectividade entre esses dispositivos é fundamental para o funcionamento eficiente dos sistemas IoT.

A Internet das Coisas (IoT) tem se tornado uma realidade cada vez mais presente em nosso cotidiano. A Internet das Coisas é a rede de objetos físicos que contém tecnologia embarcada para se comunicar e sentir ou interagir com seus estados internos ou o ambiente externo e a confluência de protocolos de redes sem fio eficientes, sensores aprimorados, processadores mais baratos e uma série de start-ups e empresas estabelecidas desenvolvendo a gestão e softwares de aplicação necessárias que finalmente tornou o conceito de Internet das Coisas popular. (VERMESAN; FRIESS, 2013)

O conceito emergente de internet das coisas é uma base crítica sobre a qual a visão para um planeta Inteligente será realizada. Se fazem necessárias mais abordagens avançadas de telemetria que permitem conectar todos os tipos de dispositivos, onde quer que estejam, entre si, com a Internet e com as empresas.

Um desses avanços é o protocolo de mensagens MQTT. É tão leve que pode ser suportado por alguns dos menores dispositivos de medição e monitoramento e transmitir dados em redes extensas e às vezes intermitentes. Também é de código aberto, o que facilita a adaptar-se a uma variedade de necessidades de mensagens e comunicação. (LAMPKIN et al., 2012).

O MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação leve e de código aberto projetado especificamente para aplicações de IoT e máquina a máquina (M2M). Foi desenvolvido pela IBM na década de 1990 e pensado para ser simples, confiável e eficiente em termos de largura de banda e consumo de energia.

Ele é mais leve que o protocolo HTTP 1.1 e, portanto, é uma opção muito interessante

sempre que precisamos enviar e receber dados quase em tempo real com um modelo de publicação-assinatura e exigimos a menor pegada possível, M2M e projetos incorporados, mas também está ganhando presença em aplicações web e aplicativos móveis que exigem mensagens seguras e uma distribuição eficiente de mensagens. (HILLAR, 2017)

O MQTT opera no modelo de publicação e assinatura (Publish/Subscribe) ou seja, os dispositivos conectados enviam mensagens para um intermediário centralizado chamado de "broker". Essas mensagens são publicadas em tópicos específicos, enquanto que outros dispositivos que desejam receber essas informações se inscrevem nos mesmos tópicos. Assim, eles recebem as mensagens relevantes. Essa abordagem permite uma comunicação assíncrona eficiente entre dispositivos, eliminando a necessidade de conexões persistentes e reduzindo a sobrecarga de comunicação.

Como principais vantagens, podemos citar:

- Eficiência Energética: O MQTT consome pouca energia, sendo ideal para dispositivos alimentados por baterias;
- Baixa Largura de Banda: Mensagens compactas resultam em baixa sobrecarga de dados, economizando recursos de rede;
- Confiabilidade: Mensagens são armazenadas no broker e entregues quando o dispositivo está online novamente;
- Flexibilidade e Escalabilidade: Pode ser usado em pequenos projetos residenciais ou implantações em grande escala;
- Segurança: Suporta autenticação e criptografia de dados.

ESP32 é uma plataforma poderosa e econômica para o desenvolvimento de aplicativos IoT. O ESP32, desenvolvido pela Espressif Systems Company (Xangai, China), oferece uma poderosa combinação de recursos e capacidades para aplicações IoT. ESP32 possui os seguintes recursos: processador dual-core, conectividade Wi-Fi e Bluetooth integrada, um grande número de pinos de entrada/saída de uso geral (GPIO) e baixo consumo de energia. O ESP32 é equipado com um microprocessador Xtensa LX6 dual-core Tensilica (Santa Clara, CA, EUA), que fornece maior poder de processamento e facilita a multitarefa e a execução eficiente de tarefas complexas. ESP32 possui interfaces Wi-Fi e Bluetooth integradas que simplificam a conexão e comunicação com outros dispositivos ou redes.

Ele suporta vários protocolos Wi-Fi, como 802.11 b/g/n, e oferece opções de conectividade Bluetooth Classic e Bluetooth Low Energy (BLE). O ESP32 fornece muitos pinos GPIO que facilitam a conexão e o controle de dispositivos e sensores externos. Esses pinos suportam uma variedade de interfaces, incluindo SPI, I2C, UART e PWM. O ESP32 foi projetado para ser eficiente em termos energéticos, permitindo assim o desenvolvimento de aplicações IoT com eficiência energética. Ele oferece modos de suspensão e recursos de gerenciamento de energia que ajudam a reduzir o consumo de energia, tornando-o adequado para projetos alimentados por bateria ou com restrição de energia. O ESP32 pode ser conectado a monitores, telas sensíveis ao toque ou indicadores LED para fornecer uma interface amigável. (HERCOG et al., 2023)

Raspberry Pico W é uma placa microcontrolada de baixo custo e alta performance, com interfaces digitais flexíveis. Ela utiliza o microcontrolador RP2040 criado pela Raspberry Pi no Reino Unido. Trata-se de um processador Dual-Core ARM Cortex M0+, rodando em até 133MHz. Possui 264kB de memória SRAM e 2MB de memória Flash integrada na placa. Além dos pinos de entradas/saídas em furos, eles também estão disponíveis de forma castelada (com isso, você pode integrar a placa diretamente no seu projeto). Ela possui suporte a USB 1.1 Host e Device e modos de low power.

O Raspberry Pico W possui 26 GPIOs multifunção, com 2x SPI, 2x I2C, 2x UART, 3x ADC de 12-bits e 16 canais PWM. Seu clock é preciso e possui timer on-chip. Ela inclui um sensor de temperatura e bibliotecas para tratar ponto flutuante. Para suporte a periféricos personalizados, ela possui 8 IOs programáveis, ou PIO de máquina de estado.

Este trabalho propõe explorar a aplicação do protocolo MQTT em microcontroladores ESP32 e Raspberry PICO W para a supervisão de sensores analógicos e digitais. Através dessa pesquisa, buscamos contribuir para o avanço da tecnologia IoT e fornecer insights valiosos para a implementação de sistemas de monitoramento eficientes e de baixo custo.

1.1 Justificativa

A relevância deste trabalho reside na crescente demanda por soluções de Internet das Coisas (IoT) que sejam eficientes e econômicas. O ESP32 é um microcontrolador que oferece conectividade Wi-Fi e Bluetooth, ideal para projetos IoT.

O protocolo MQTT se destaca por sua leveza e eficiência, sendo amplamente adotado para comunicação entre dispositivos IoT devido ao seu modelo de publicação e assinatura que facilita a integração dos sistemas.

Este projeto propõe a utilização do ESP32 em conjunto com o MQTT para criar um sistema de monitoramento remoto de sensores, tais como temperatura e umidade, bem como dispositivos que dispõe de contato elétrico, para monitoramento de abertura e fechamento de portas por exemplo, proporcionando uma solução de baixo custo e de fácil implementação para ambientes residenciais e industriais. Através deste sistema, busca-se explorar as potencialidades do ESP32 e do MQTT, avaliando sua aplicabilidade prática e contribuindo para o avanço da automação residencial e industrial.

2 Metodologia

Este projeto consiste em promover um sistema de comunicação e monitoramento de variáveis, sejam elas analógicas ou digitais, bem como realizar o acionamento de dispositivos à distância, como por exemplo acionar lâmpadas, utilizando componentes de baixo custo e a rede WiFi disponível nas residências, comércios e indústrias.

Para a comunicação via protocolo MQTT, é necessária a utilização do broker, que seria uma espécie de servidor que irá gerenciar a comunicação entre os dispositivos conectados à rede, que neste caso serão o ESP32, o Raspberry Pico W e o dashboard que será utilizado para monitorar as variáveis.

Conforme mostrado na figura 1, tanto para o broker, quanto para o dashboard, serão utilizados aplicativos para celular, que podem ser baixados de forma gratuita, o que corrobora para a redução de custo do projeto.

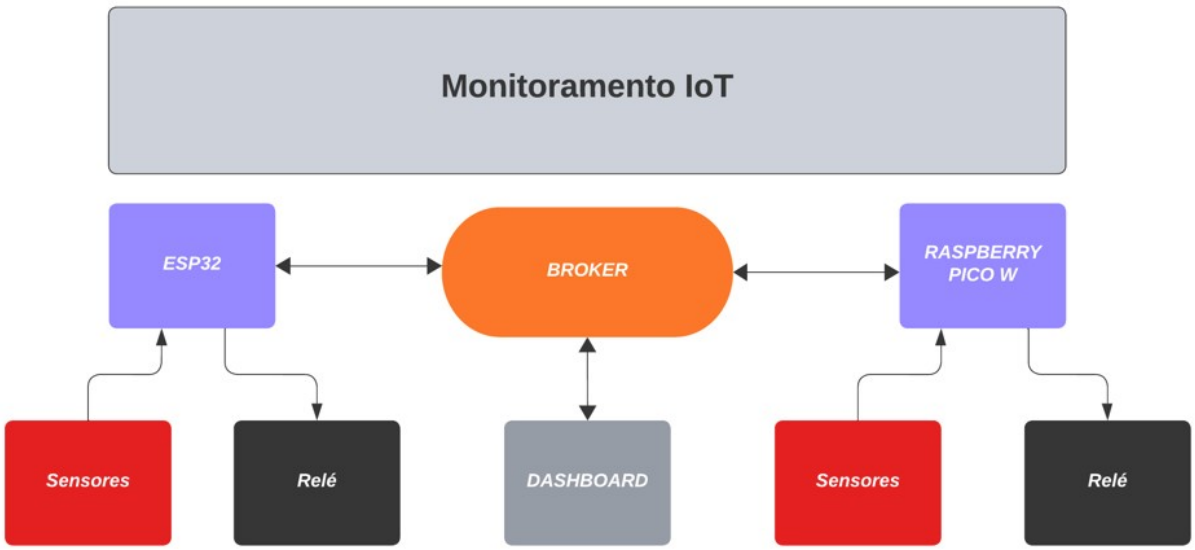

Figura 1 – Diagrama em blocos da rede MQTT a ser montada, com os principais componentes. Fonte: Próprio autor

2.1 Componentes

Além dos microcontroladores ESP32 e Raspberry Pico W citados, serão utilizados sensores de temperatura e umidade, sensor tipo reed switch e para chaveamento da saída será utilizado um relé.

2.1.1 Sensor DHT22

Este sensor possui sinal de saída digital calibrado. Aplica técnica exclusiva de coleta de sinal digital e tecnologia de sensor de umidade, garantindo sua confiabilidade e estabilidade. Seus elementos sensores são conectados a um único chip de 8 bits de computador.

Cada sensor deste modelo é compensado e calibrado em temperatura em uma câmara de calibração precisa e o coeficiente de calibração é salvo em tipo de programa na memória OTP, quando o sensor estiver detectando, ele relacionará com o coeficiente salvo na memória.

Tamanho pequeno, baixo consumo e longa distância de transmissão (100m) permitem que o DHT22 seja adequado em todos os tipos de oportunidades de aplicação difíceis. Possui invólucro com quatro pinos e pode ser fornecido em uma placa com três pinos, tornando a conexão muito conveniente. A figura 2 mostra o modelo de sensor utilizado.

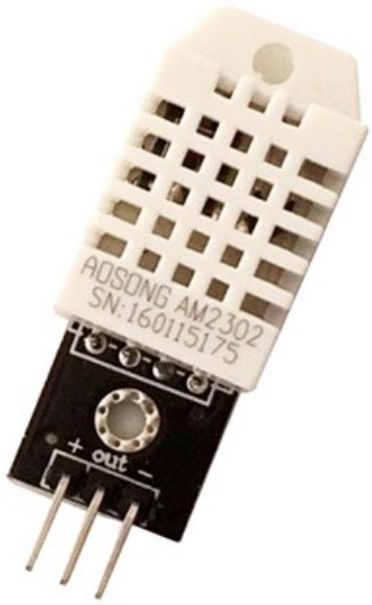

Figura 2 – Sensor DHT22. Disponível em https://components101.com/sensors/dht22-pinout-specs-datasheet. Acesso em: 15/05/2024.

2.1.2 Sensor reed switch

O sensor de campo magnético mais simples é um reed switch. Ele contém dois elementos ferromagnéticos de níquel e palheta de ferro em um tubo de vidro hermeticamente selado e com um gás inerte para minimizar o arco de contato e evitar a oxidação dos contatos.

Quando um ímã se aproxima da chave, sua força magnética fecha as palhetas. Da mesma forma, quando o ímã se afasta, as palhetas retornam ao seu estado aberto. Na figura 3 temos um exemplo do funcionamento do sensor.

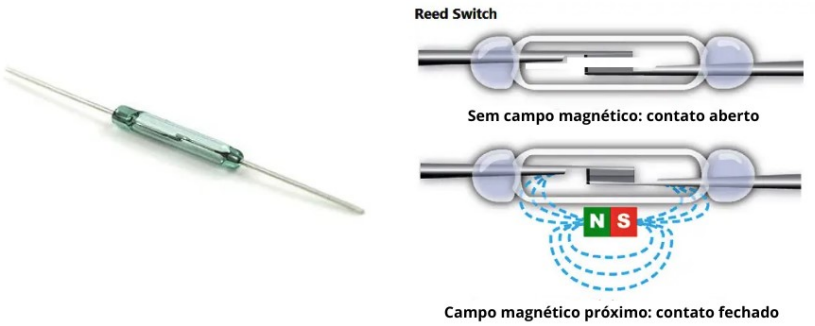

Figura 3 – Sensor reed switch. Disponível em https://instrumentationtools.com/reed-switch-working-principle. Acesso em: 15/05/2024.

Para atender à proposta do projeto, que visa utilizar o sensor reed para monitorar a abertura e fechamento de uma porta ou janela, será utilizado um modelo de sensor reed fabricado em invólucro plástico, pois nessa configuração ficará mais fácil a instalação do sensor no local desejado para o monitoramento. Na figura 4 mostra o modelo de sensor utilizado para otimizar a instalação do mesmo em portas e janelas.

Figura 4 – Sensor reed switch. Disponível em https://altronics.cl/sensor-magnetico-mc-38. Acesso em: 20/05/2024.

2.1.3 Relés Eletromecânicos

Os microprocessadores e microcontroladores trabalham com tensões de alimentação entre 3.3VDC e 5VDC. Desta forma, para acionamento de dispositivos externos ele disponibiliza tensão de alimentação desta ordem, o que se torna inviável caso haja a necessidade de acionamento de dispositivos que requerem maior potência (pequenos motores de corrente contínua, válvulas solenóides, etc.).

Sendo assim, se faz necessária a utilização de dispositivos que terão como função realizar a interface entre o microcontrolador e a carga a ser acionada. Estes dispositivos têm basicamente duas funções: acionar cargas acima das potências fornecidas pelos terminais de saídas dos microcontroladores, bem como isolar o circuito de controle contra possíveis surtos e sobrecargas que possam ocorrer na carga.

Para o projeto, será utilizado para o acionamento da trava que libera a abertura da porta / janela um relé eletromecânico com bobina de 5VDC e contatos que suportam até 8A de corrente elétrica.

Abaixo, na figura 5, temos um exemplo do modelo de relé eletromecânico utilizado na montagem do protótipo.

Figura 5 – Sensor reed switch. Disponível em https://www.eletrolico.com.br/reles-eletromecanicos/rele-miniat-5a-5vcc-2-rev-pci. Acesso em: 21/05/2024.

2.1.4 Microcontroladores

Para o projeto em questão foram escolhidos dois modelos de microcontroladores: o ESP32 e o Raspberry Pico W, pois deseja-se demonstrar que o protocolo MQTT não está restrita a somente uma plataforma, mas que de fato é um protocolo aberto, de baixo custo

e disponível em diversas plataformas.

Para a programação dos microcontroladores será utilizada a linguagem de programação MicroPython. De acordo com o próprio site, MicroPython é uma implementação enxuta e eficiente da linguagem de programação Python 3 que inclui um pequeno subconjunto da biblioteca padrão Python e é otimizada para ser executada em microcontroladores e em ambientes restritos. O MicroPython está disponível para uso geral sob a licença MIT, ou seja, permite que o software seja considerado open source e tratado sem restrições para o uso, modificação ou distribuição.

Para auxiliar na programação dos microcontroladores, será utilizado o software também gratuito Thonny, que irá transferir o programa criado em micropython para o microcontrolador.

2.2 Aplicativos para smartphone

Como mencionado anteriormente, o protocolo MQTT necessita de um broker, que funcionará como um gerenciador das mensagens trocadas entre os clientes conectados, bem como dashboard para monitorar as informações da rede.

Além disso, será utilizado um aplicativo que ficará responsável por exibir os valores de temperatura e umidade, além do status do sensor reed e da saída relé.

2.2.1 Narada MQTT Broker

Broker MQTT de código aberto que pode ser executado no Android. Útil para prototipagem e execução de um servidor MQTT em smartphones Android. A escolha do aplicativo para Android, em detrimento à plataforma IOS dos smartphones da APPLE tem como base dados do site Statcounter, atualizados em abril de 2024, onde se demonstra que no Brasil os smartphones Android utilizados ocupam 81,38% do mercado, enquanto que os aparelhos que utilizam plataforma IOS ocupam 18,39%.

Neste aplicativo é possível configurar dados de acesso à rede MQTT como por exemplo o endereço IP do roteador que será utilizado para a comunicação, nome de usuário e senha, para garantir maior segurança no acesso às informações. Na figura 6 temos exemplos da tela de configuração do aplicativo.

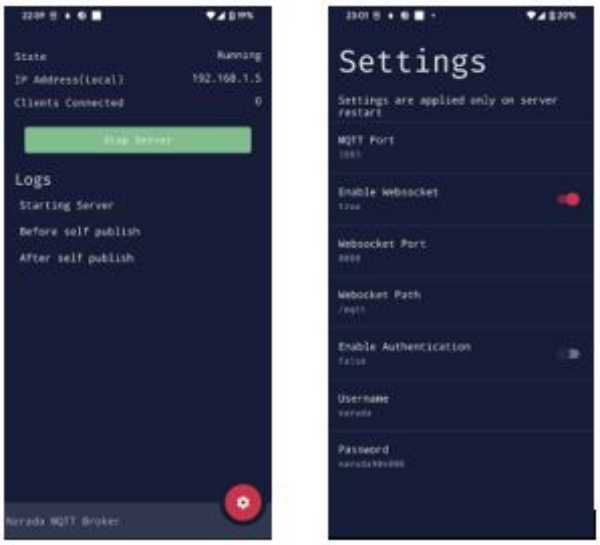

Figura 6 – Telas de configuração do Narada MQTT Broker. Disponível em https://play.google.com/store/apps/details?id=in.eswarm.narada. Acesso em: 22/05/2024.

2.2.2 MQTT Dashboard

O que é um dashboard? Várias respostas para esta pergunta podem ser dadas. Pode ser um aplicativo mobile, um arquivo em PDF, uma tela montada numa parede mostrando informações. Dashboard é uma exibição visual de dados usada para monitorar as condições e/ou facilitar a compreensão.

Um dashboard, no contexto de TI, é um painel visual que apresenta, de maneira centralizada, um conjunto informações: indicadores e suas métricas. Essas informações podem ser tanto indicadores da área de TI como de gestão empresarial. (**??**)

Em ambos os casos, esse recurso auxilia na tomada de decisões. Dashboards podem apresentar a saúde da empresa em uma única tela para o gestor ou compartilhados com toda a sua equipe. (**??**)

Um dashboard com enfoque em questões técnicas, infraestrutura por exemplo, serve para a análise do desempenho e da disponibilidade de dispositivos, aplicações e das tecnologias aplicadas aos processos da empresa. Já os dashboards para a gestão de negócios oferecem um panorama dos indicadores da performance geral da organização nessa área. (**??**)

MQTT Dashboard é um aplicativo mobile que, ao estar conectado ao Broker, será responsável por receber os dados enviados dos clientes, como por exemplo temperatura, umidade e status de porta aberta/fechada para visualização destas variáveis em tempo real

na tela. Também é possível configurar um botão para acionar a saída relé do protótipo. Na figura 7 temos as telas de configuração do aplicativo. (WEXLER; SHAFFER; COTGREAVE, 2017)

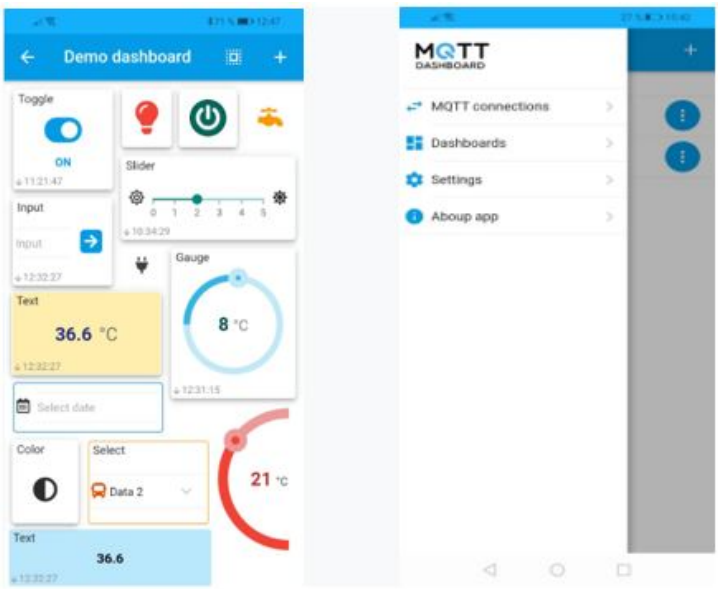

Figura 7 – Telas de configuração do MQTT Dashboad. Disponível em https://www.gameloop.com/br/game/tools/com.lapetov.mqtt. Acesso em: 25/05/2024.

2.3 Esquema elétrico

Abaixo, nas figuras 8 e 9, temos os esquemas elétricos dos circuitos que irão tratar os sinais dos sensores e enviar para o broker via protocolo MQTT. Um dos circuitos usará o ESP32 como base e o outro usará o RASPBERRY PICO W, mostrando assim que o protocolo pode ser utilizado em diferentes plataformas.

Em seguida, na figura 10, segue o diagrama mostrando todo o fluxo de dados da rede utilizando o protocolo MQTT.

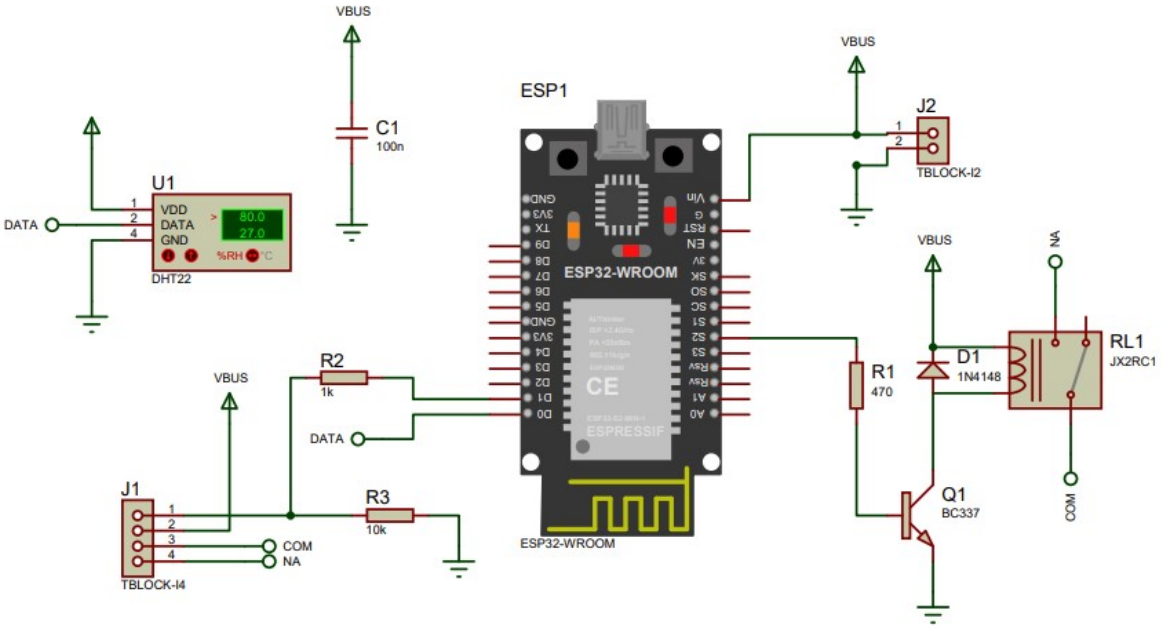

Figura 8 – Esquema elétrico do cliente MQTT que será controlado pelo ESP32. Fonte: Próprio autor

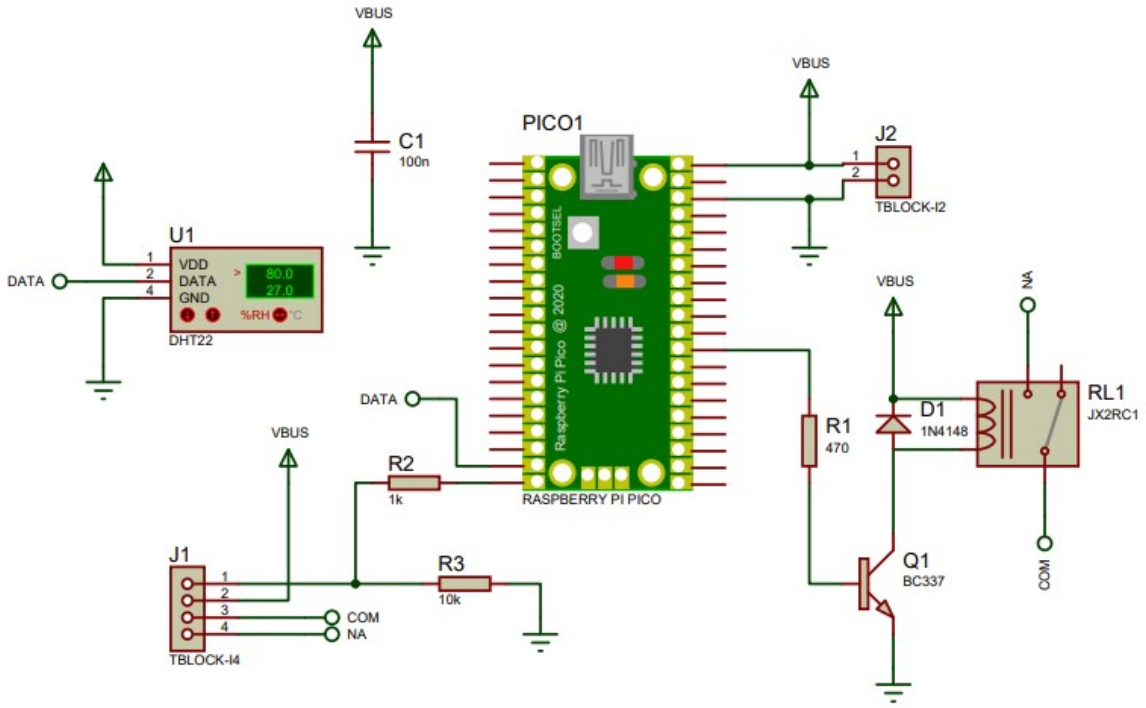

Figura 9 – Esquema elétrico do cliente MQTT que será controlado pelo RASPBERRY PICO W. Fonte: Próprio autor

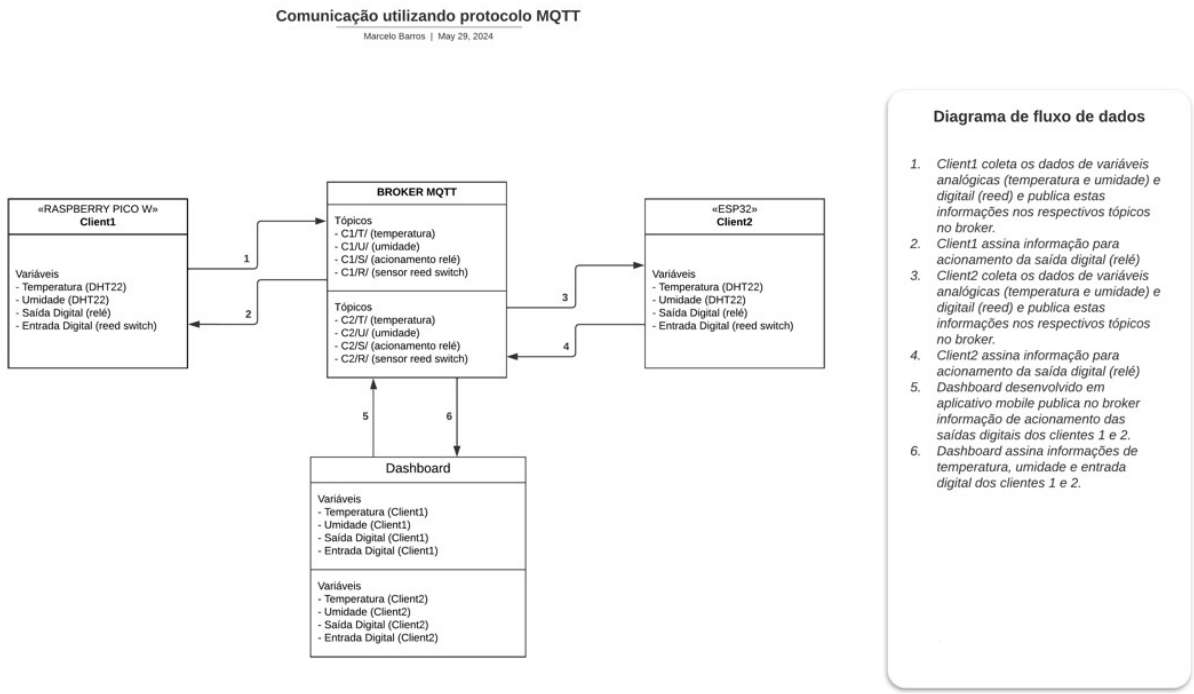

Figura 10 – Diagrama da rede MQTT mostrando em detalhes a comunicação entre os componentes conectados. Fonte: Próprio autor

3 Resultados

Foram desenvolvidos os protótipos a serem estudados, um deles com o ESP32 (figura 11) e o outro com o RASPBERRY PICO W (figura 12). Nos dois protótipos foram instalados o sensor de temperatura e umidade, o sensor reed switch e o relé para acionamento de cargas em corrente contínua ou alternada. Para melhor ilustração do acionamento de cargas pelo relé, foi também instalado um led que acenderá através do contato do relé.

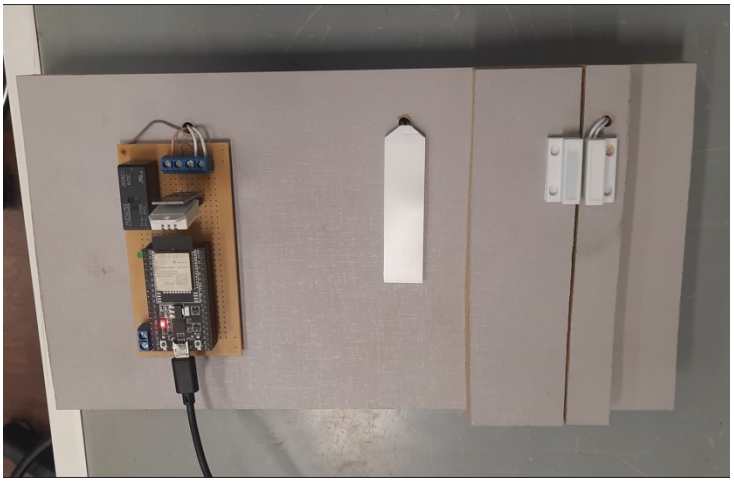

Figura 11 – Protótipo utilizando ESP32. Fonte: Próprio autor

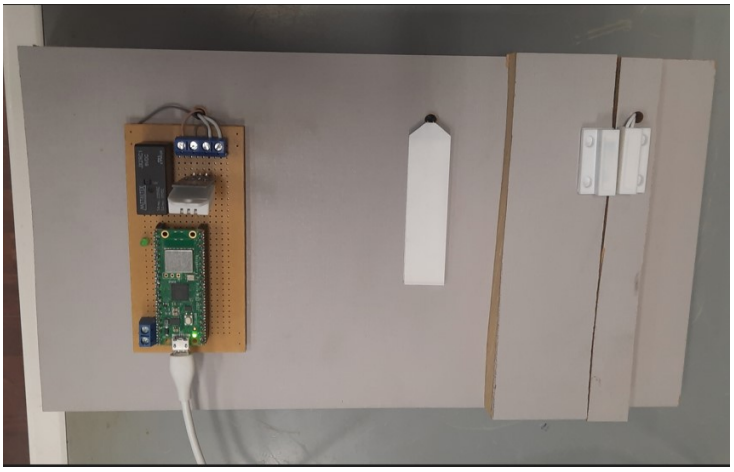

Figura 12 – Protótipo utilizando RASPBERRY PICO W. Fonte: Próprio autor

Para o Broker MQTT foi configurado o aplicativo mobile Narada MQTT (figura 13).

Após a configuração e inicialização do broker, pôde ser observado a conexão do broker com os protótipos, visto que é possível o monitoramento das atividades de publicação e assinaturas dos tópicos definidos (C1 = Client1(RASPBERRY); C2 = Client2 (ESP32); T = Temperatura; U = Umidade; R = Reed Switch; S = Saída relé).

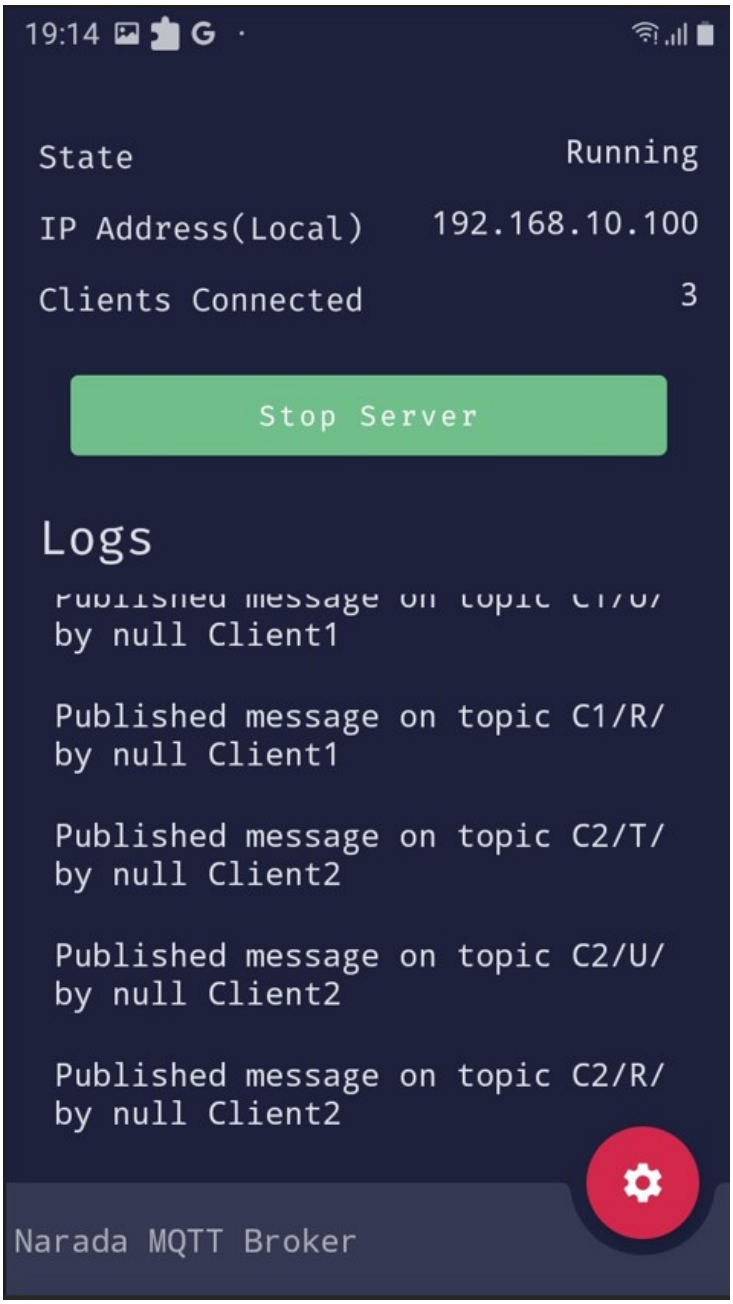

Figura 13 – Broker MQTT em operação. Fonte: Próprio autor

Em seguida, foi realizada a configuração e montagem do dashboard. para tanto, foi utilizado o aplicativo mobile MQTT Dashboard (figura 14). Após vincular os ícones do dashboard com os tópicos, os dados de temperatura e umidade passaram a ser exibidos, bem como ficaram disponíveis o ícone de status do reed switch e o ícone de acionamento

da saída relé.

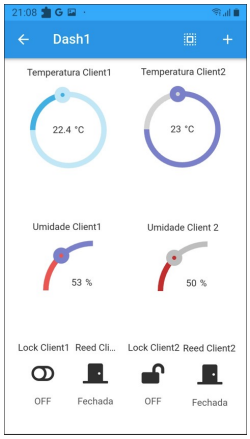

Figura 14 – Dashboard em operação. Fonte: Próprio autor

Após esta etapa, foram iniciados os testes de acionamento da saída relé (figuras 15 e 16) e teste de status do reed switch (figuras 17 e 18)

Figura 15 – Acionamento do ícone no dashboard responsável por acionar o relé do protótipo RASPBERRY e a resposta com a energização do led. Fonte: Próprio autor

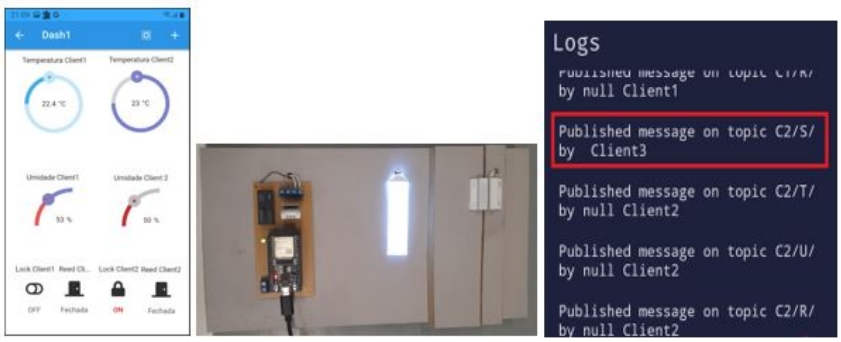

Figura 16 – Acionamento do ícone no dashboard responsável por acionar o relé do protótipo ESP32 e a resposta com a energização do led. Fonte: Próprio autor

Ao acionar o ícone de acionamento da trava do Client1 (saída relé, identificado como "lock" no dashboard), o mesmo publica no tópico C1/S/ o valor do payload definido no programa gravado no RASPBERRY. No caso do ESP32, o mesmo publicará no tópico C2/S/ como sendo Client2. Os microcontroladores então, configurados para realizar varredura a cada 5 segundos, verificam que houve alteração no tópico e executam a ação definida no programa. Neste caso, a ação definida é ligar o relé e, consequentemente, acionar a carga que, no caso, é um led.

Dessa forma, concluiu-se que os comandos via dashboard para acionamento da saída relé de ambos os protótipos funcionaram de acordo com o que foi projetado. Para o teste do reed switch, foi montado no protótipo uma estrutura que tem por objetivo simular o funcionamento de uma porta ou janela.

O intuito é mostrar que, ao abrir a "porta", o reed switch irá abrir seu contato elétrico interno. Dessa forma será enviado o sinal para uma entrada do microcontrolador que, identificando a alteração do status, irá publicar no tópico C1/R/ o valor do payload definido no programa. Neste caso, o dashboard receberá esta informação e fará a alteração do ícone que representa a porta aberta/fechada.

No dashboard também pode ser observado que o monitoramento da temperatura e umidade está funcionando normalmente, com atualizações a cada 5 segundos. Este tempo pode ser alterado visando uma maior varredura desta e das demais variáveis.

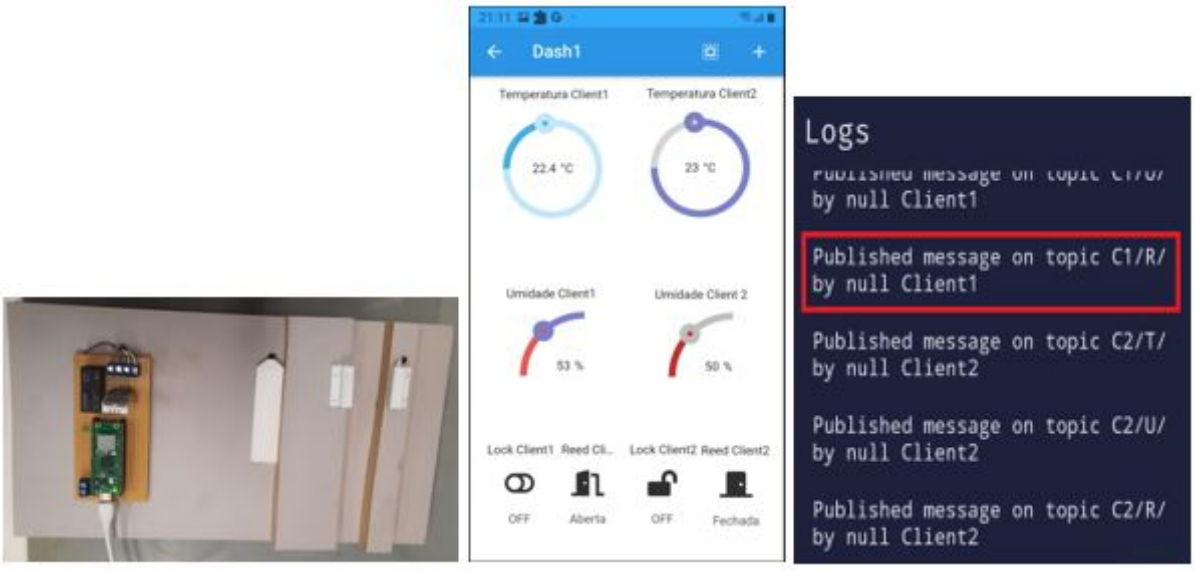

Figura 17 – Teste do reed switch no protótipo RASPBERRY e respectivo ícone no dashboard. Fonte: Próprio autor

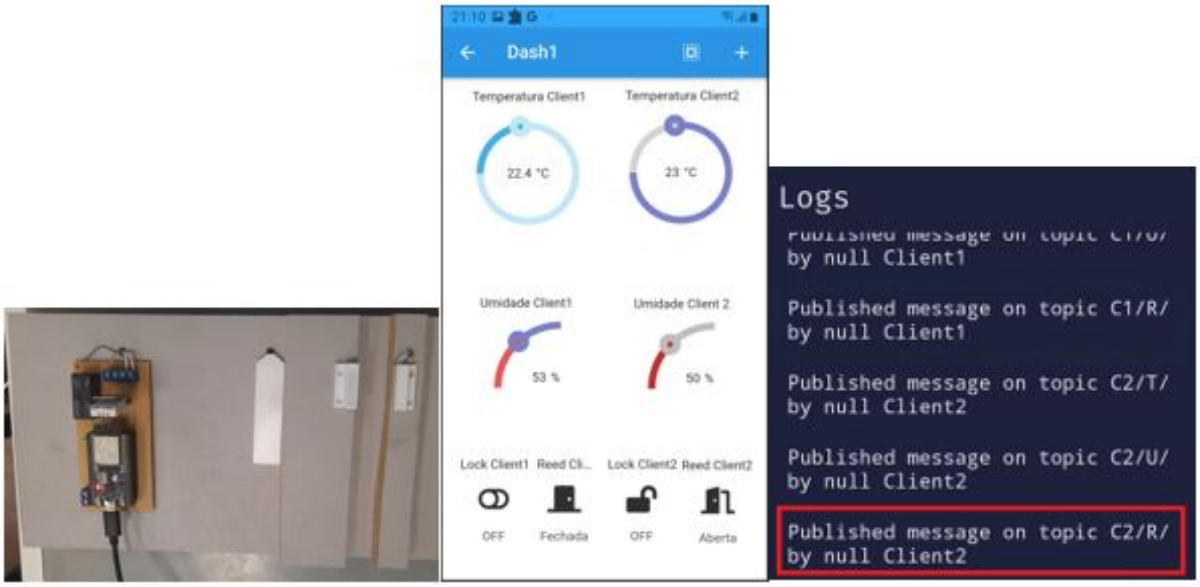

Figura 18 – Teste do reed switch no protótipo ESP32 e respectivo ícone no dashboard. Fonte: Próprio autor

Assim, comprovou-se o pleno funcionamento do reed switch e sua respectiva indicação no dashboard.

Deve-se salientar que os tópicos identificados com “C1” se referem ao protótipo RASPBERRY e os tópicos identificados com “C2” se referem ao protótipo ESP32.

4 Discussão

O protocolo MQTT funcionou adequadamente durante os testes dos dispositivos conectados. Não foram verificadas falhas de conexão entre os componentes, bem como todos os comandos de acionamento de dispositivos e recebimento de status das variáveis foram executados de maneira satisfatória.

Para questões de custo e segurança, foi utilizado roteador de uso residencial totalmente desconectado da internet. O sistema pode ser utilizado para monitoramento online utilizando-se servidores por exemplo, mas para isso deverão ser considerados custos adicionais (utilização de brokers privados, pois utilização de brokers públicos podem ocasionar vazamento de dados) e também o nível de qualidade de serviço (QoS), prevendo evitar possíveis ataques externos à rede.

A utilização de dois microcontroladores distintos visa demonstrar que o protocolo pode ser executado em diferentes hardwares, dando maiores opções para avaliação no que diz respeito aos fatores custo / desempenho

Como ponto de melhoria, a implementação de alimentação de todos os componentes da rede por meio de bateria para os casos onde houver queda de energia por curto prazo de tempo.

Deve-se ter atenção ao número de dispositivos conectados ao broker MQTT para não gerar problemas de latência excessiva, perda de dados dos tópicos ou problemas de conctividade.

5 Conclusão

O protocolo MQTT foi criado em 1999 para utilização em áreas como petróleo e gás, onde era necessário o monitoramento de variáveis em locais remotos e de difícil acesso. Em 2010 o protocolo passou a ser gratuito e aberto, onde abriu-se uma enorme gama de oportunidades de utilização em outras áreas.

Este trabalho tem por objetivo demonstrar que o protocolo MQTT pode ser utilizado em componentes de baixo custo, o que permite sua utilização não somente pelas empresas, mas também nos comércios e residências que desejam automatizar e monitorar processos e ambientes com pouco investimento e de maneira autônoma.

A Código Fonte para o RASPBERRY

```
# Importe as bibliotecas necess rias
import time, utime
from umqtt.simple import MQTTClient
import machine
from machine import Pin
import random
import dht

# Dados para acesso rede WiFi
SSID = "IoT_"
SSID_PASSWORD = "12345678"

# Comando para conectar rede WiFi
def do_connect():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('Conectando    rede...')
        sta_if.active(True)
        sta_if.connect(SSID, SSID_PASSWORD)
        while not sta_if.isconnected():
            print("Conectando....")
            utime.sleep(1)
    print('Conectado!!! Configura  o de rede:', sta_if.ifconfig())

print("Conectando com o WiFi...")
do_connect()

# Para conectar ao MQTT_BROKER
MQTT_BROKER = "192.168.10.100"
CLIENT_ID = "Client1"

#Lista de topicos
SUBSCRIBE_TOPIC = b"C1/S/"
PUBLISH_TOPIC1 = b"C1/R/"
PUBLISH_TOPIC2 = b"C1/T/"
PUBLISH_TOPIC3 = b"C1/U/"

# Setup das entradas do Raspberry
led = Pin("LED",Pin.OUT)
p29 = Pin(22,Pin.OUT)
p20 = Pin(15,Pin.IN)

# Publicando mensagens no broker MQTT a cada 5 segundos
```

```
last_publish = time.time()
publish_interval = 5

# Recebendo mensagem dos assinantes (subscribers)
def sub_cb(topic, msg):
    print((topic, msg))
    if msg.decode() == "O":
       # led.value(1)
        p29.value(1)

    else:
        if msg.decode() == "o":
        #    led.value(0)
            p29.value(0)


def reset():
    print("Resetando...")
    time.sleep(5)
    machine.reset()

# Conex o com o broker
def main():
    print(f"Iniciando conex o com MQTT Broker :: {MQTT_BROKER}")
    mqttClient = MQTTClient(CLIENT_ID, MQTT_BROKER, keepalive=60)
    mqttClient.set_callback(sub_cb)
    mqttClient.connect()
    mqttClient.subscribe(SUBSCRIBE_TOPIC)
    print(f"Conectado com MQTT  Broker :: {MQTT_BROKER}, e aguardando
    t picos")
    while True:
            # Monitoramento das variaveis
            led.value(1)
            mqttClient.check_msg()
            global last_publish

            sensor = dht.DHT22(Pin(14))
            sensor.measure()
            print(f"Temperatura : {sensor.temperature():.1f} C ")
            print(f"Umidade     : {sensor.humidity():.1f}%")

            if (time.time() - last_publish) >= publish_interval:

                mqttClient.publish(PUBLISH_TOPIC2, str(sensor.
    temperature()))
                mqttClient.publish(PUBLISH_TOPIC3, str(sensor.humidity()
    ))
```

```
88
89                  if p20.value():
90                      mqttClient.publish(PUBLISH_TOPIC1, "fechada")
91
92                  else:
93                      mqttClient.publish(PUBLISH_TOPIC1, "aberta")
94
95                  last_publish = time.time()
96              time.sleep(1)
97
98
99  if __name__ == "__main__":
100     while True:
101         try:
102             main()
103         except OSError as e:
104             print("Error: " + str(e))
105             reset()
106
```

B Código Fonte para o ESP32

```
# Importe as bibliotecas necess rias
import network
import time, utime
from umqtt.simple import MQTTClient
import machine
from machine import Pin
import random
import dht

# Dados para acesso rede WiFi
SSID = "IoT_"
SSID_PASSWORD = "12345678"

# Comando para conectar rede WiFi
def do_connect():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('Conectando    rede...')
        sta_if.active(True)
        sta_if.connect(SSID, SSID_PASSWORD)
        while not sta_if.isconnected():
            print("Conectando....")
            utime.sleep(1)
    print('Conectado!!! Configura  o de rede:', sta_if.ifconfig())

print("Conectando com o WiFi...")
do_connect()

# Para conectar ao MQTT_BROKER
MQTT_BROKER = "192.168.10.100"
CLIENT_ID = "Client2"

#Lista de topicos
SUBSCRIBE_TOPIC = b"C2/S/"
PUBLISH_TOPIC1 = b"C2/R/"
PUBLISH_TOPIC2 = b"C2/T/"
PUBLISH_TOPIC3 = b"C2/U/"

# Setup das entradas do Raspberry
led = Pin(16,Pin.OUT)
p29 = Pin(32,Pin.OUT)
p20 = Pin(22,Pin.IN)

```

```
44 # Publicando mensagens MQTT a cada 5 segundos
45 last_publish = time.time()
46 publish_interval = 5
47
48 # Recebendo mensagem do Broker
49 def sub_cb(topic, msg):
50     print((topic, msg))
51     if msg.decode() == "O":
52         led.value(1)
53         p29.value(1)
54
55     else:
56         if msg.decode() == "o":
57             led.value(0)
58             p29.value(0)
59
60
61 def reset():
62     print("Resetando...")
63     time.sleep(5)
64     machine.reset()
65
66 def main():
67     print(f"Iniciando conex o com MQTT Broker :: {MQTT_BROKER}")
68     mqttClient = MQTTClient(CLIENT_ID, MQTT_BROKER, keepalive=60)
69     mqttClient.set_callback(sub_cb)
70     mqttClient.connect()
71     mqttClient.subscribe(SUBSCRIBE_TOPIC)
72     print(f"Conectado com MQTT  Broker :: {MQTT_BROKER}, e aguardando
    fun   o callback ser chamada!")
73     while True:
74             # Monitoramento das variaveis
75             mqttClient.check_msg()
76             global last_publish
77
78             sensor = dht.DHT22(Pin(23))
79             sensor.measure()
80             print(f"Temperatura : {sensor.temperature():.1f} C ")
81             print(f"Umidade     : {sensor.humidity():.1f}%")
82
83             if (time.time() - last_publish) >= publish_interval:
84
85                 mqttClient.publish(PUBLISH_TOPIC2, str(sensor.
    temperature()))
86                 mqttClient.publish(PUBLISH_TOPIC3, str(sensor.humidity()
    ))
87
```

```
88
89                  if p20.value():
90                      mqttClient.publish(PUBLISH_TOPIC1, "fechada")
91
92                  else:
93                      mqttClient.publish(PUBLISH_TOPIC1, "aberta")
94
95                  last_publish = time.time()
96              time.sleep(1)
97
98
99  if __name__ == "__main__":
100     while True:
101         try:
102             main()
103         except OSError as e:
104             print("Error: " + str(e))
105             reset()
106
```

Referências

HERCOG, D. et al. Design and implementation of esp32-based iot devices. iot sensors and technologies for education. **MPDI Open Acess Journal**, sn, v. 1, p. 5, 2023. Citado na página 4.

HILLAR, G. **MQTT Essentials - A Lightweight IoT Protocol: Send and receive messages with the MQTT protocol for your IoT solutions**. [S.l.]: Packt Publishing, 2017. Citado na página 3.

LAMPKIN, V. et al. **Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry**. [S.l.]: Redbooks, 2012. Citado na página 2.

MAGRANI, E. **A Internetdas Coisas**. [S.l.]: FGV Editora, 2018. Citado na página 2.

VERMESAN, O.; FRIESS, P. **Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems**. [S.l.]: River Publishers, 2013. Citado na página 2.

WEXLER, S.; SHAFFER, J.; COTGREAVE, A. **The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios**. [S.l.]: Wiley Publishing, 2017. Citado na página 12.