UM SISTEMA DE MONITORAMENTO INTELIGENTE DE INFRAESTRUTURA PREDIAL

1. INTRODUÇÃO

Este projeto tem como objetivo o desenvolvimento de um sistema de monitoramento inteligente para condomínios, utilizando tecnologias de Internet das Coisas (IoT) para acompanhar em tempo real aspectos essenciais da infraestrutura predial. O sistema monitora três áreas principais: o nível da caixa d'água, o consumo de energia elétrica e a abertura do portão principal, oferecendo maior controle, segurança e eficiência operacional para a administração do condomínio.

O sistema de monitoramento da caixa d'água utiliza um sensor ultrassônico acoplado a um microcontrolador ESP32 para medir o nível de água em tempo real. As medições são realizadas continuamente, garantindo um acompanhamento preciso. O ESP32 processa os dados obtidos e identifica variações inesperadas no nível. Essas variações podem indicar possíveis vazamentos no reservatório. Ao detectar anomalias, o sistema pode acionar alertas visuais ou sonoros. Isso permite uma resposta rápida para evitar o desperdício de água.

No controle de acesso, um sensor de fim de curso monitora a abertura e fechamento do portão, informando o status em tempo real. Este sensor também está ligado a um ESP32, responsável por processar os dados localmente e enviá-los para a nuvem. Já o monitoramento do consumo de energia elétrica é feito por meio de um sensor de corrente, conectado a um microcontrolador ESP32. Esse sensor registra o uso de energia no condomínio, permitindo análises que podem identificar picos de consumo, possibilitar economia e prevenir falhas na rede elétrica.

Todos os dispositivos se comunicam com um gateway central, que por sua vez envia os dados para a nuvem utilizando o protocolo MQTT, permitindo o acesso remoto por meio de um aplicativo móvel. Com isso, os usuários podem acompanhar o estado do sistema de forma prática e em tempo real, de qualquer lugar.

O principal objetivo deste projeto é oferecer uma solução acessível, eficiente e automatizada para o gerenciamento de recursos essenciais em condomínios, promovendo sustentabilidade, segurança e maior comodidade para os moradores e administradores.

2. OBJETIVOS

O projeto tem como objetivo principal desenvolver um sistema de monitoramento inteligente para infraestrutura predial, com foco em condomínios residenciais ou comerciais, utilizando tecnologias de Internet das Coisas (IoT) para garantir maior eficiência, segurança e autonomia na gestão de recursos essenciais.

Objetivos Específicos:

- Monitorar o nível da caixa d'água em tempo real.
- Monitorar o acesso físico ao condomínio.
- Monitorar o consumo de energia elétrica
- Centralizar a coleta e transmissão de dados.
- Disponibilizar uma interface remota de visualização.
- Promover a sustentabilidade e a modernização da gestão condominial.

3. FUNDAMENTAÇÃO TEÓRICA

3.1. ARQUITETURA DO SISTEMA

O presente sistema propõe o monitoramento e controle de dispositivos físicos por meio de nós baseados no microcontrolador ESP32, comunicando-se com o Gateway via Wi-Fi. Essa arquitetura segue o padrão de redes IoT, em que sensores coletam dados locais e os transmitem para processamento remoto, conforme mostra a Figura 1.

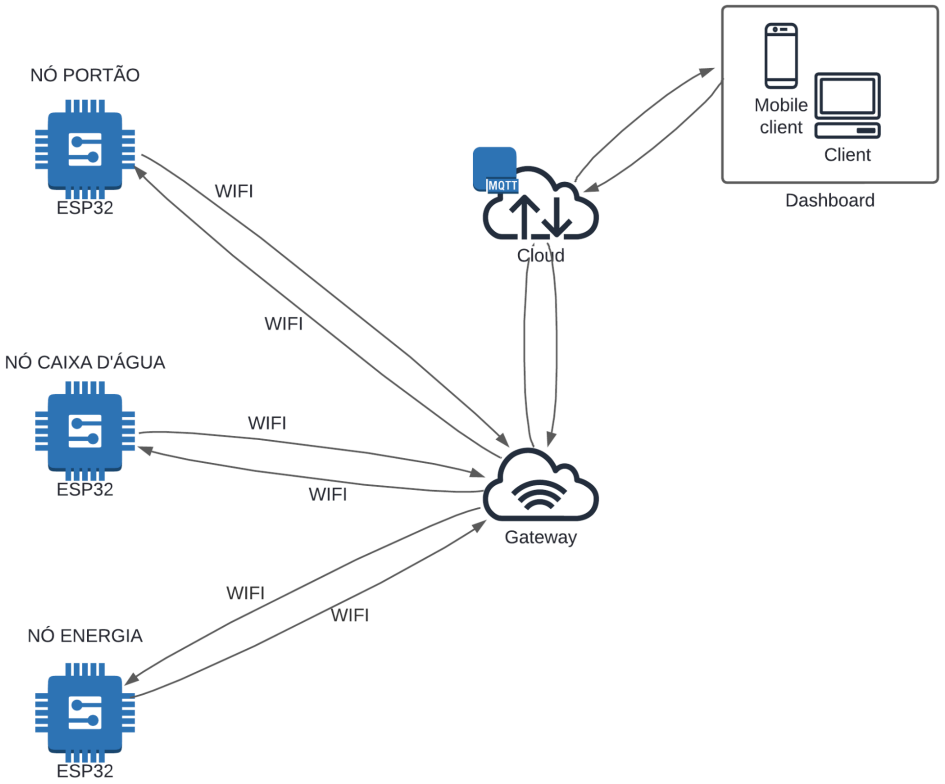

Figura 1. Arquitetura do sistema

O sistema segue a arquitetura clássica de Internet das Coisas (IoT), baseada em três camadas principais:

- Camada de Percepção: onde os sensores coletam os dados físicos do ambiente.
- Camada de Rede: responsável pela transmissão dos dados, usando Wi-Fi e o protocolo MQTT.
- Camada de Aplicação: onde os dados são processados e visualizados pelos usuários.

Essa abordagem modular permite escalabilidade, manutenção simplificada e integração com diferentes tipos de sensores e atuadores.

3.2. SERVIDOR MQTT

Em projetos de Internet das Coisas, a escolha do protocolo de comunicação entre sensores e servidor é essencial para garantir eficiência, confiabilidade e baixo consumo de recursos. Existem diversas alternativas como o protocolo HTTP, WebSockets e o MQTT.

O HTTP, apesar de amplamente difundido, impõe uma sobrecarga considerável ao exigir conexões persistentes ou requisições constantes, o que o torna inadequado para microcontroladores com recursos limitados.

Já o WebSocket, embora permita comunicação bidirecional, demanda maior complexidade de configuração no servidor e mais poder computacional no cliente.

Diante disso, optou-se pelo protocolo MQTT (Message Queuing Telemetry Transport), uma solução leve, assíncrona e orientada a eventos, ideal para cenários com conectividade intermitente e dispositivos de baixo custo. Ele adota um modelo de publicação e assinatura (pub/sub), onde os sensores publicam dados em tópicos definidos e os clientes interessados apenas se inscrevem nesses canais para recebê-los. Essa abordagem reduz o acoplamento entre os componentes do sistema e permite maior escalabilidade.

Como broker MQTT, foi utilizado o HiveMQ Cloud, uma solução baseada em nuvem que oferece autenticação via TLS, alta disponibilidade e uma interface de gerenciamento acessível. A escolha por esse serviço considerou a facilidade de integração com o ESP32, o suporte gratuito para projetos de pequeno porte e a eliminação da necessidade de configurar um broker local como o Mosquitto, simplificando o desenvolvimento e a implantação do sistema.

3.3. SERVIDOR BACKEND

Para garantir a persistência dos dados gerados pelos sensores e viabilizar seu acesso posterior de forma estruturada, será desenvolvido um servidor backend utilizando o framework Express, em Node.js. Esse servidor atuará como intermediário entre o broker MQTT e o sistema de visualização, sendo responsável por se inscrever nos tópicos relevantes do broker, receber as mensagens publicadas pelos sensores, processar as informações e armazená-las em um banco de dados relacional PostgreSQL.

A escolha do Express se deu pela sua leveza, simplicidade e ampla adoção na construção de APIs REST, o que favorece uma integração ágil com o dashboard e outros sistemas que venham a consumir os dados no futuro.

O uso de um banco de dados relacional permite realizar consultas eficientes, agregações e filtros temporais, importantes para análises e histórico. A estrutura modular do backend facilita a manutenção, testes e a eventual escalabilidade do sistema, podendo-se integrar futuramente mecanismos de autenticação, controle de acesso e até exportação de relatórios.

3.4. DASHBOARD

A visualização dos dados em tempo real é uma parte essencial do projeto, pois torna as informações captadas pelos sensores acessíveis de maneira clara e intuitiva aos administradores e moradores do condomínio.

Após análise de diversas soluções disponíveis no mercado, optou-se pelo desenvolvimento de um dashboard próprio utilizando Flutter e Dart. Essa escolha permite a criação de uma aplicação multiplataforma com uma única base de código, capaz de rodar em navegadores, dispositivos Android e iOS. Além disso, o Flutter oferece um ecossistema moderno e produtivo para construção de interfaces responsivas e com boa experiência de usuário.

A interface será simples, com foco na funcionalidade e clareza, exibindo dados em tempo real através de gráficos, indicadores e tabelas. O dashboard se conectará ao servidor backend por meio de uma API REST, consumindo os dados armazenados no banco de dados e atualizando a interface dinamicamente. Essa arquitetura separada, com backend e frontend desacoplados, proporciona maior flexibilidade para futuras evoluções e facilita a manutenção do sistema como um todo.

3.5. TIPOS DE REDES

A escolha da tecnologia de rede para a comunicação entre os sensores e o sistema central foi orientada por critérios como alcance, compatibilidade com o hardware, facilidade de implementação e

custo. Entre as opções consideradas estavam o LoRaWAN, ZigBee, Bluetooth Low Energy (BLE) e Wi-Fi.

- O LoRaWAN, embora ofereça longo alcance e baixo consumo de energia, exige a presença de gateways específicos e uma infraestrutura dedicada, o que o tornaria inviável para o ambiente urbano e edificado do projeto.

- O ZigBee também apresenta bom desempenho energético e permite a criação de redes mesh, porém sua integração com o ESP32 requer módulos adicionais e configurações específicas.

- O BLE é eficiente em conexões ponto a ponto de curta distância, mas limitado em alcance e volume de dados transmitidos.

- O Wi-Fi, por sua vez, mostrou-se a escolha mais viável, considerando que o ESP32 possui conectividade Wi-Fi integrada, eliminando a necessidade de componentes extras. Além disso, redes Wi-Fi já estão presentes nos ambientes prediais em que o sistema será implantado, o que reduz os custos de infraestrutura. A maior largura de banda e a facilidade de acesso à internet tornam essa tecnologia adequada mesmo com o consumo de energia ligeiramente superior aos demais.

O projeto utiliza WLAN (Wi-Fi) como rede local sem fio, escolhida pela compatibilidade com o ESP32, facilidade de implementação e disponibilidade em ambientes prediais, garantindo comunicação estável e de baixo custo.

3.6. TIPOS DE PROTOCOLOS

Na camada de aplicação do sistema, é preciso selecionar um protocolo que equilibre desempenho, simplicidade e compatibilidade com os dispositivos embarcados. Foram avaliadas opções como HTTP/HTTPS, CoAP e MQTT.

- O uso de HTTP implica em maior consumo de recursos, uma vez que cada requisição carrega informações de cabeçalho e precisa ser processada individualmente, o que não é ideal em aplicações com envio constante de dados.

- O protocolo CoAP, desenvolvido especificamente para IoT, apresenta estrutura leve e suporte a redes com baixa largura de banda, mas ainda carece de ampla adoção e compatibilidade com ferramentas de visualização e brokers disponíveis no mercado.

- O MQTT, por sua vez, é um protocolo consolidado na comunidade de IoT, com bibliotecas maduras para o ESP32, ampla compatibilidade com plataformas em nuvem e suporte nativo a mecanismos de publicação e assinatura. Sua arquitetura favorece a confiabilidade e a escalabilidade, mesmo em redes sujeitas a instabilidades momentâneas. Por esses motivos, o MQTT foi adotado como protocolo principal na comunicação entre sensores e o sistema de backend, oferecendo desempenho consistente e facilidade de expansão do sistema no futuro.

4. METODOLOGIA

4.1. MATERIAIS UTILIZADOS

- Microcontroladores:
    - ESP32 DevKit S2 (portão)
    - ESP32 DevKit C3 supermini (2 unidades para nó de nível de água e nó de corrente)
- Sensores:

    - Reed switch + ímã (portão)
    - HC-SR04 (nível de água)
    - ACS712 20A (corrente elétrica)
- Componentes adicionais: resistores 10 kΩ, jumpers, protoboard, fonte 5V 2A
- Rede: Roteador Huawei WiFi AX2S
- Broker MQTT: HiveMQ (versão cloud gratuita)
- Dashboard: ….
- Software de programação: VSCode com extensão PlatformIO utilizando a biblioteca Arduino

4.2. PROCEDIMENTOS DE IMPLEMENTAÇÃO

4.2.1. Montagem do nó do portão

O nó do portão utiliza um sensor de fim de curso do tipo reed switch (Figura 2), que opera em estado normalmente aberto, que se fecha na presença de um campo magnético, permitindo detectar a proximidade de um ímã. Esses sensores são amplamente aplicados em sistemas de alarme e controle de acesso por serem confiáveis, baratos e de fácil instalação.

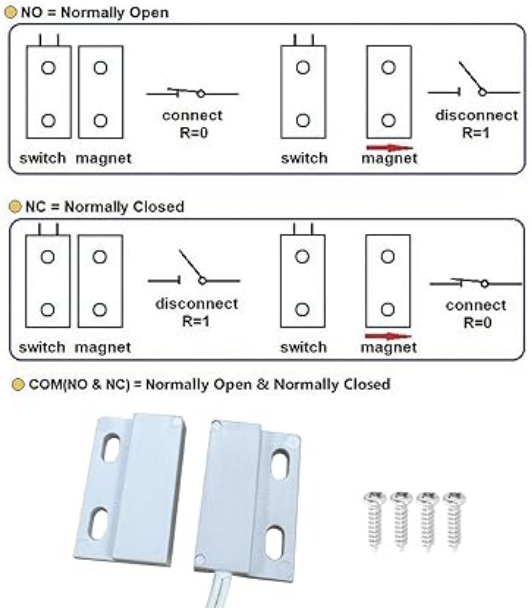

Figura 2. Sensor de fim de curso e diferenças entre NO (Normally Open) e NC (Normally Closed)[1].

Além do sensor, o nó também utiliza um microcontrolador Esp32 e uma bateria de 5 volts. O sensor de fim de curso está conectado ao Esp32 que, por sua vez, está sendo alimentado pela bateria, conforme mostra o diagrama de blocos da Figura 3.

[1] Disponivel em: https://www.amazon.com/Magnetic-Normally-Cabinet-Contact-Plastic/dp/B0CPLFT1QF

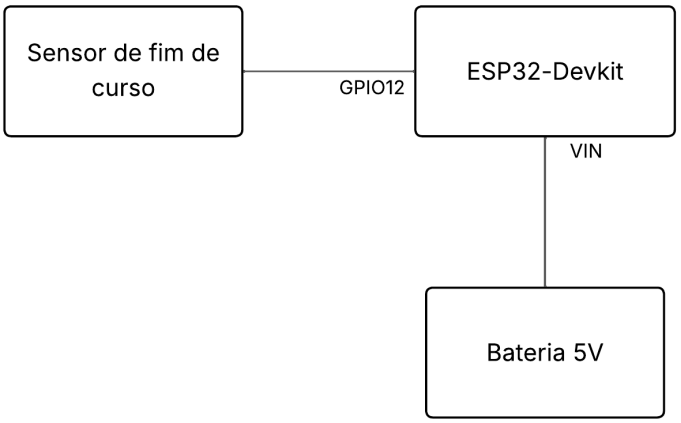

Figura 3. Diagrama de blocos do nó do portão.

Um dos terminais do sensor é ligado na entrada GPIO 32 do ESP32, e conectado a um resistor de 10kΩ com o GND, no formato de uma chave pulldown. O outro terminal é ligado à mesma bateria de 5V que o microcontrolador.

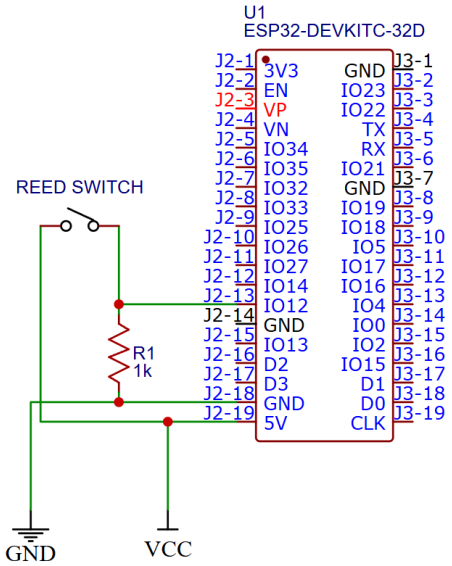

Figura 4. Esquemático do nó do portão

O sinal gerado por esse sensor é lido por um microcontrolador ESP32 (Figura 4), que alimentado por uma bateria de 5V realiza a comunicação com o sensor e transmite os dados ao gateway via rede Wi-Fi, possibilitando o monitoramento remoto do estado do portão.

A taxa de atualização deste nó é baseada em eventos (event-driven), ou seja, os dados são transmitidos sempre que ocorre uma mudança de estado no sensor (por exemplo, ao abrir ou fechar o portão), o que reduz o tráfego na rede e economiza energia.

4.2.2. Montagem do nó da caixa d'água

Esse nó utiliza um sensor ultrassônico HC-SR04 (Figura 5) que é utilizado para medir distâncias com base na emissão de ondas sonoras de alta frequência (40 kHz).

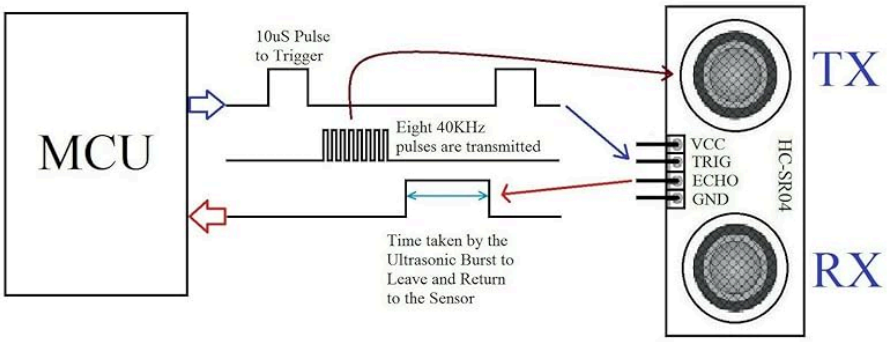

Figura 5. Módulo HC-SR04[2]

O módulo envia oito pulsos ultrassônicos, calcula o tempo de retorno (eco) e, utilizando a fórmula (distância = (tempo em nível alto × velocidade do som (340 m/s)) / 2) que determina a distância em centímetros até o obstáculo.

Possui alcance de aproximadamente 2 a 400 cm e precisão de ±3 mm. É muito usado em robótica, automação e medição de nível de líquidos devido ao seu baixo custo e facilidade de uso.

Assim como no nó do portão, este nó também utiliza um microcontrolador ESP32, que realiza a aquisição dos dados dos sensores por meio da entrada GPIO (Figura 6). O microcontrolador atua como interface entre o sensor e a nuvem, realizando o envio contínuo das informações para o gateway central do sistema via Wi-Fi.

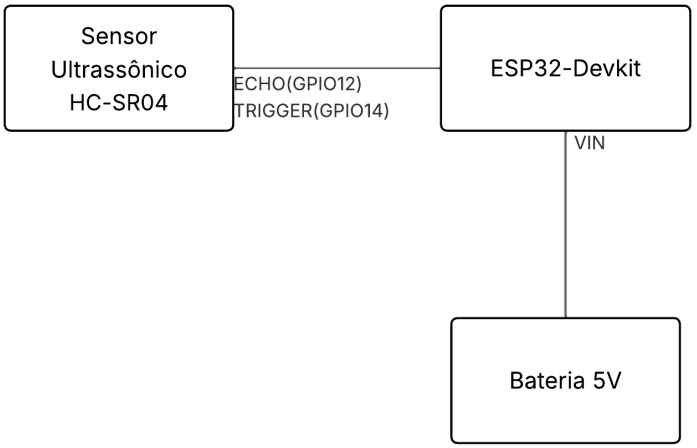

Figura 6. Diagrama de blocos do nó de caixa d'água

O sensor ultrassônico HC-SR04 possui quatro pinos: dois para alimentação (VCC e GND) e dois para comunicação (Trigger e Echo). Neste projeto, os pinos Trigger e Echo serão conectados ao ESP32 utilizando os pinos GPIO 14 e GPIO 12, respectivamente, conforme mostra o esquema da Figura 7.

[2] Disponível em:
https://www.amazon.com/WWZMDiB-HC-SR04-Ultrasonic-Distance-Measuring/dp/B0B1MJJLJP

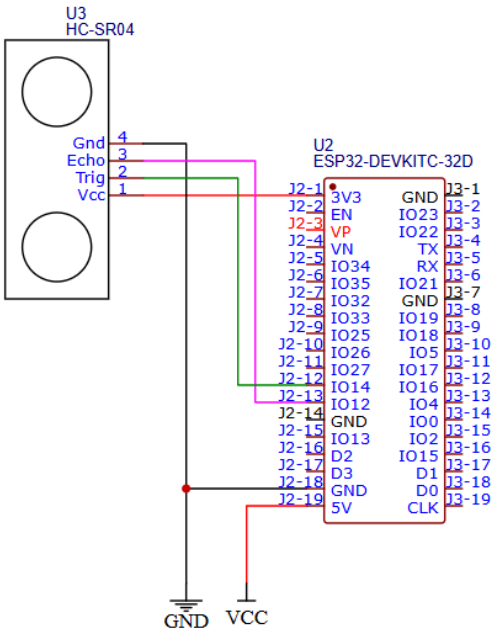

Figura 7. Esquemático nó da caixa d'água

A taxa de atualização para esse nó será a cada 2 minutos, permitindo uma resposta rápida a um possível vazamento e permitindo o monitoramento preciso via Dashboard.

4.2.3. Montagem do nó de energia

O nó é responsável pelo monitoramento do consumo de energia elétrica do prédio. Neste caso, será utilizado o sensor ACS712 (Figura 8), que é um sensor de corrente baseado em efeito Hall, capaz de medir corrente alternada (AC) ou contínua (DC) até 5A, 20A ou 30A conforme o modelo.

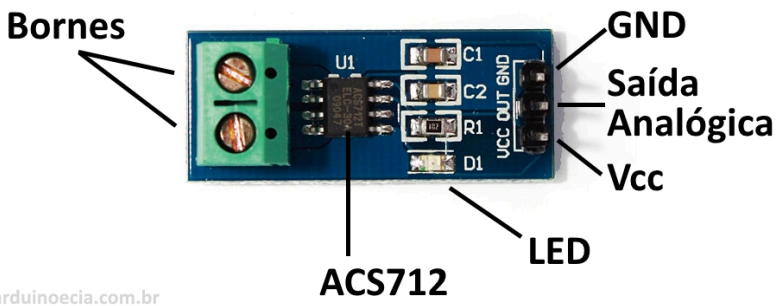

Figura 8. Módulo sensor de corrente ACS712[3].

Sua saída é analógica e proporcional à corrente medida. O sensor é isolado eletricamente do circuito principal, oferecendo segurança adicional, sendo comum em projetos de monitoramento de consumo elétrico.

O sinal analógico será lido pelo microcontrolador ESP32 por meio de uma GPIO, e processado para estimar a potência elétrica com base em valores de referência configurados (Figura 9).

[3] Disponível em: https://www.arduinoecia.com.br/como-usar-o-sensor-de-corrente-acs712

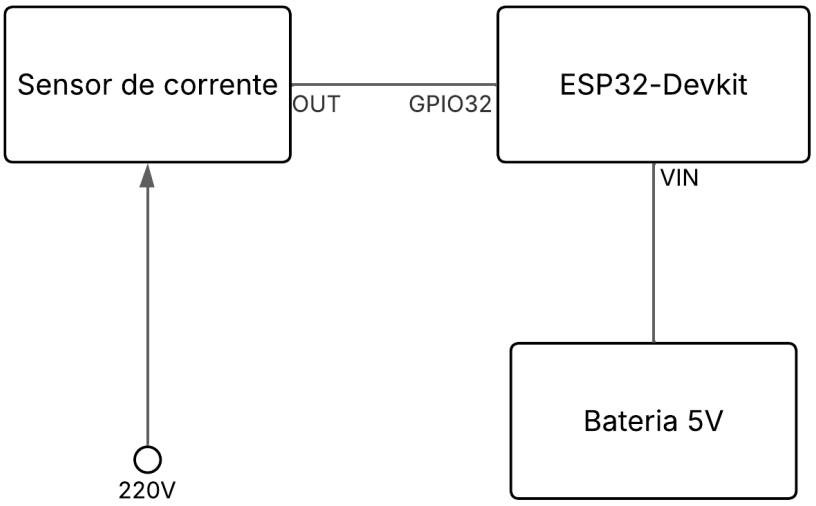

Figura 9. Diagrama de blocos do nó sensor de corrente

O sensor é ligado à rede elétrica de 220V através de dois bornes (fase e neutro), e ao microcontrolador serão ligados três terminais, os de alimentação (VCC e GND) e o que transmite a leitura do sensor, o terminal OUT, que será ligado à GPIO 32, conforme mostra a Figura 10.

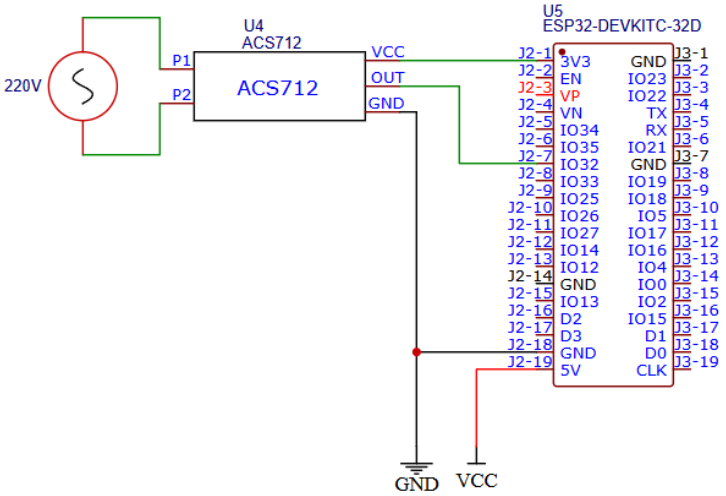

Figura 10. Esquemático Sensor de corrente

O microcontrolador fará amostragens em alta frequência localmente (entre 50 e 100 amostras por segundo), mas enviará valores médios ou agregados ao servidor MQTT em intervalos de aproximadamente 5 minutos, otimizando a largura de banda e o desempenho do sistema. Isso possibilita o acompanhamento remoto em tempo real do perfil de uso da energia elétrica no edifício.

4.2.4. Configuração do broker MQTT (HiveMQ)

- Criar conta em https://console.hivemq.cloud
- Criar instância do broker e definir credenciais (usuário/senha)
- Anotar URL e porta (8883 para TLS)
- Testar usando o servidor Express.js.
    - Autenticação com host, porta, usuário e senha

4.2.5. Configuração do dashboard

O dashboard será desenvolvido com Flutter e Dart, com suporte a múltiplas plataformas, incluindo navegadores e dispositivos móveis. A interface será simples, composta por gráficos, indicadores e tabelas. Os dados exibidos incluem:

- Gráfico de linha para o nível da caixa d'água
- Indicador de status (texto ou ícone) para o portão (aberto/fechado)
- Gráfico de barras ou histórico para o consumo de energia

A aplicação irá se conectar à API REST exposta pelo servidor backend para buscar os dados em tempo real e armazenados. As atualizações serão feitas automaticamente em intervalos regulares. O Flutter foi escolhido por sua eficiência, performance nativa e facilidade de manutenção da interface com componentes reutilizáveis.

4.2.6. Configuração do servidor backend

O servidor backend será desenvolvido com Node.js e Express, atuando como ponte entre o broker MQTT e o dashboard. Ele realiza a inscrição nos tópicos MQTT e recebe os dados publicados pelos dispositivos. Cada mensagem é processada, convertida em JSON e armazenada em um banco de dados PostgreSQL utilizando Sequelize como ORM.

O servidor também fornece uma API RESTful com endpoints que permitem ao dashboard consultar os dados de forma segura e eficiente, com suporte a filtros por período, tipo de sensor, entre outros.

Variáveis de ambiente são usadas para proteger credenciais e permitir flexibilidade na configuração. Essa abordagem centraliza a lógica do sistema, facilita escalabilidade e garante segurança na manipulação dos dados.