



# Manual do Usuário

**KLIF  
PIRELLI**

## Controle do Documento

### Histórico de revisões

Data	Autor	Versão	Resumo da atividade
04/06/2023	Eduarda	1.1	Criação da seção 1 (completa)
05/06/2023	Marcos	1.2	Criação da seção 3 e 4
05/06/2023	Eduarda	1.3	Criação da seção 5
10/06/2023	Luiz	1.4	Atualização da seção 5
09/06/2023	Luiz	1.5	Criação da seção 2
10/06/2023	Luiz	1.6	Criação da seção 6
11/06/2023	Raab	1.7	Atualização da seção 6

# Índice

1. Componentes e Recursos	3
1.1. Componentes externos	3
1.2. Requisitos de conectividade	4
2. Guia de Montagem	5
3. Guia de Instalação	9
4. Guia de Configuração	11
5. Guia de Operação	12
6. Troubleshooting	14

# 1. Componentes e Recursos

## 1.1. Componentes externos

A solução proposta para resolver o problema de perda e extravio de tablets/notebooks na fábrica da PIRELLI consiste no desenvolvimento de um sistema de rastreio utilizando dispositivos com tecnologia de geolocalização. Esses dispositivos serão integrados aos tablets/notebooks utilizados pela empresa. Os dados de localização serão transmitidos em tempo real para um *dashboard* na plataforma Ubidots, proporcionando à empresa uma visão clara e estimada da localização de seus dispositivos. Para implementar essa solução, serão necessários um tablet/notebook e os componentes adicionais específicos:

- 1. **Esp32 Wroom com Antena - 38 pinos (2 pcs)**: É um circuito integrado, responsável pela execução e alimentação do protótipo.
- 2. **Jumper fêmea-fêmea - 20 cm (20 pcs)**: Responsável por algumas conexões eletrônicas entre o Esp e os componentes.
- 2. **Jumper macho-fêmea - 20 cm (20 pcs)**: Responsável por algumas conexões eletrônicas entre o Esp e os componentes.
- 2. **Jumper macho-macho - 20 cm (20 pcs)**: Responsável por algumas conexões eletrônicas entre o Esp e os componentes.
- 3. **Resistor 10k ohms (3 pcs)**: Responsável por limitar a diferença de potencial elétrico em partes específicas do circuito (Ex: Leds).
- 3. **Resistor 1k ohms (3 pcs)**: Responsável por limitar a diferença de potencial elétrico em partes específicas do circuito (EX: Push Button).
- 4. **LED - Vermelho (1 pcs)**: Responsável pelo alerta visual definido do escopo do projeto.
- 4. **LED - Amarelo (1 pcs)**: Responsável pelo alerta visual definido do escopo do projeto.
- 4. **LED - Verde (1 pcs)**: Responsável pelo alerta visual definido do escopo do projeto.
- 5. **Sensor Ultrassônico HC-SR04 (2 pcs)**: Responsável por medir distâncias de 2cm a 4cm entre o dispositivo e o equipamento monitorado.
- 6. **Display LCD 16x2 (1 pcs)**: Responsável pelo alerta visual escrito, definido do escopo do projeto.
- 7. **Kit RFID Mfrc522 (1 pcs)**: Responsável pela identificação e controle de acesso.

- **8. Sensor BME280 (1 pcs)**: Responsável pela medição de temperatura e umidade dos dispositivos *host*.
- **9. Cabo Micro USB (2 pcs)**: Responsável pela fonte de alimentação elétrica para o Esp.

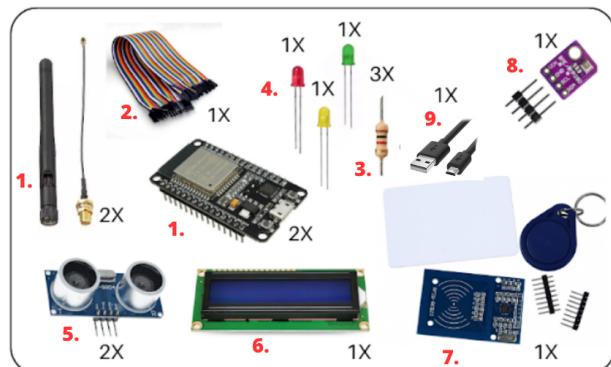


Imagen 1: Componentes utilizados e suas respectivas quantidades.

## 1.2. Requisitos de conectividade

Para que o aplicativo funcione corretamente, é essencial ter uma conexão Wi-Fi disponível. Essa conexão é estabelecida utilizando as bibliotecas `<WiFi.h>` e `<WiFiClient.h>`. Ter acesso à Internet é crucial para permitir a comunicação entre os dispositivos e a plataforma Ubidots. Os dispositivos precisam estar conectados à Internet para enviar e receber dados por meio do painel de controle. Além disso, a biblioteca "UbidotsEsp32Mqtt.h" é utilizada para implementar o protocolo MQTT, que possibilita a conexão com o Ubidots.

A biblioteca "UbidotsEsp32Mqtt.h" foi especialmente desenvolvida para o microcontrolador ESP32 e possibilita a comunicação com a plataforma Ubidots usando o protocolo MQTT (*Message Queuing Telemetry Transport*). O Ubidots é uma plataforma de IoT (Internet das Coisas) que permite coletar, armazenar, visualizar e analisar dados de sensores e dispositivos conectados. Com a biblioteca "UbidotsEsp32Mqtt.h", você pode enviar dados de sensores ou variáveis do seu microcontrolador ESP32 para a plataforma Ubidots usando o protocolo MQTT, além de receber comandos ou notificações da plataforma.

Além disso, é importante mencionar o protocolo I2C (*Inter-Integrated Circuit*), que é utilizado para a comunicação entre o ESP32 e alguns componentes, como o dispositivo LCD e o sensor BME280. Esse protocolo permite a transmissão de dados entre dispositivos usando apenas dois fios, o que facilita a conexão e a transferência de informações. No caso desses componentes, é necessário conectá-los exclusivamente às portas 21 e 22 do ESP32, que são as portas designadas para a comunicação I2C.

## 2. Guia de Montagem

Para auxiliar na montagem do equipamento, fornecemos dois guias: o Guia de Montagem do ESP Host e o Guia de Montagem do ESP Client.

### HOST

No Guia de Montagem do ESP Host, utilizamos os seguintes componentes, conforme mostrado na Imagem 2, cada um com sua função específica:

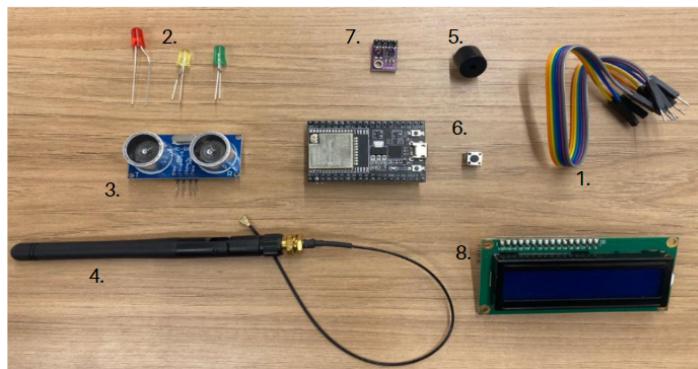


Imagen 2: Componentes utilizados no ESP Host.

**1. Jumpers:** São utilizados para fazer a conexão entre os componentes e o ESP Host.

**2. LEDs:** Os LEDs são conectados às saídas quinze (LED vermelho - Desconectado), zero (LED verde - Conectado) e dois (LED amarelo - Aguardando conexão) por meio de jumpers, juntamente com uma

saída GND. Cada LED possui um resistor de 300 ohms para limitar a corrente e evitar danos.

**3. Sensor ultrassônico:** A conexão com o ESP Host é feita através de jumpers. O pino VCC é conectado à entrada de energia, o pino trig é conectado à porta 26 e o pino Echo à porta 27. Além disso, o pino GND é conectado para aterramento, assim como nos LEDs.

**4. Antena WiFi:** É conectada ao ESP Host para ampliar a potência do sinal na comunicação entre os dispositivos.

Além disso, uma protoboard é utilizada para interconectar todos os componentes, e uma bateria de 3,3V é usada para alimentar cada dispositivo (*Host* e *Client*).

**5. Buzzer:** É conectado à porta 25 e ao GND para aterramento.

**6. PushButton:** É conectado à porta 32 e ao GND, juntamente com um resistor de 10k ohms.

**7. Sensor BME280:** A conexão com o ESP Host é feita pelos pinos SCL (porta 22), SDA (porta 21), VIN (entrada de energia) e GND (aterramento).

**8. LCD:** A conexão do LCD com o ESP Host é feita pelos pinos SCL (porta 22), SDA (porta 21), VCC (entrada de energia) e GND (aterramento).

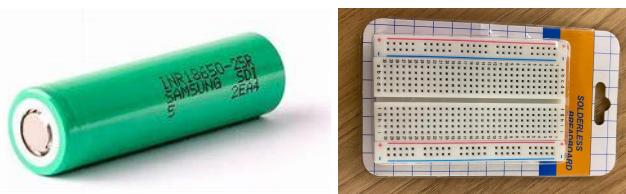


Imagen 3: Bateria e Protoboard.

Observação: É importante lembrar que todas as portas mencionadas acima podem ser modificadas no código-fonte, se necessário, exceto para o dispositivo LCD e o BME280, que se comunicam via protocolo I2C e devem ser exclusivamente nas portas 21 e 22.

## CLIENT

No Guia de Montagem do ESP *Client*, utilizamos os seguintes componentes:

- 1. Jumpers:** São utilizados para estabelecer a conexão entre os componentes e o ESP *Host*.
- 2. LEDs:** Os LEDs são conectados aos jumpers para estabelecer a conexão com as saídas quinze (LED vermelho - Desconectado), zero (LED verde - Conectado) e dois (LED amarelo - Aguardando conexão), juntamente com uma saída GND. Cada LED possui um resistor de 300 ohms para limitar a corrente e evitar danos.
- 3. Sensor ultrassônico:** A conexão com o ESP *Host* é estabelecida através dos jumpers. O pino VCC é conectado à entrada de energia, o pino trig é conectado à porta 26 e o pino Echo à porta 27. Além disso, o pino GND é conectado para aterramento, assim como nos LEDs.

Além disso, utiliza-se uma protoboard para interconectar todos os componentes, e uma bateria de 3,3V é utilizada para fornecer energia a cada dispositivo (*Host e Client*).

**4. Antena WiFi:** É conectada ao ESP *Host* para aumentar a potência do sinal na comunicação entre os dispositivos.



Imagen 4: Representação visual do dispositivo conectado ao tablet.

## Passo a Passo para a Montagem

### CLIENT



Imagen 5: ESP32 do *Client*.

**1.** Conecte o ESP32, via porta USB, com o tablet/notebook.



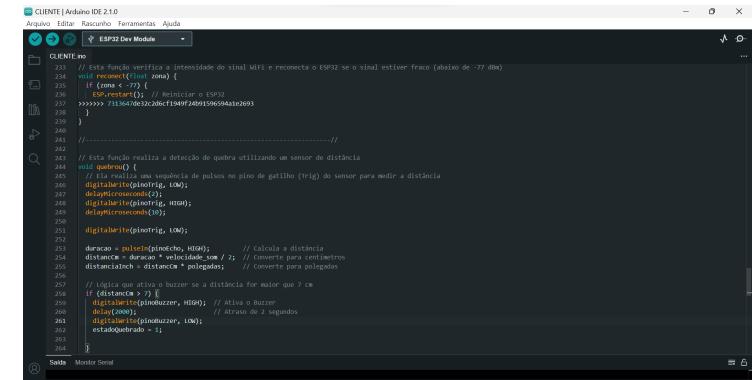
Imagen 6: Conectando o ESP32, via porta USB, com o notebook.

**2.** Insira a antena Wi-Fi na entrada superior do ESP32



Imagen 7: Inserindo a antena Wi-Fi no ESP32.

**3.** Utilize o Arduino IDE em conjunto com o código-fonte disponibilizado e faça testes progressivos.



```

CLIENT | Arduino IDE 2.1.0
Arquivo  Editor  Rascunho  Ferramentas  Ajuda
ESP32 Dev Module

O: CLIENT.h
131 // esta função verifica a intensidade do sinal WiFi e reconecta o ESP32 se o sinal estiver fraco (abaixo de -77 dBm)
132 void reconnect(ssid_t zone) {
133     // Zone
134     // ESP32
135     // >>>> 73136d4de32c2d6fc1940f24091596594a1c2693
136     // >>>>
137     // >>>>
138     // Esta função realiza a detecção de queda utilizando um sensor de distância
139     void pulseInQ() {
140         // Faz a leitura de uma sequência de pulsos no pino de gatilho (trig) do sensor para medir a distância
141         digitalWrite(pinoTrig, HIGH);
142         digitalWrite(pinoTrig, LOW);
143         digitalWrite(pinoTrig, HIGH);
144         delayMicroseconds(10);
145         digitalWrite(pinoTrig, LOW);
146         // Calcula a distância
147         duracao = pulseIn(pinoEcho, HIGH); // Calcula a distância
148         distancia = duracao * velocidade_som / 2; // Converte para centímetros
149         distanciaCm = distancia * polegadas; // Converte para polegadas
150         distanciaInches = distancia * polegadas;
151         // Lógica que ativa o buzzer se a distância for maior que 7 cm
152         if (distanciaCm > 7) {
153             digitalWrite(pinoBuzzer, HIGH); // Ativa o Buzzer
154             delay(2000);
155             digitalWrite(pinoBuzzer, LOW);
156             estadosQuadrado = 1;
157         }
158     }
}

```

Imagen 8: Utilizando o Arduino IDE para programação e testes.

## HOST

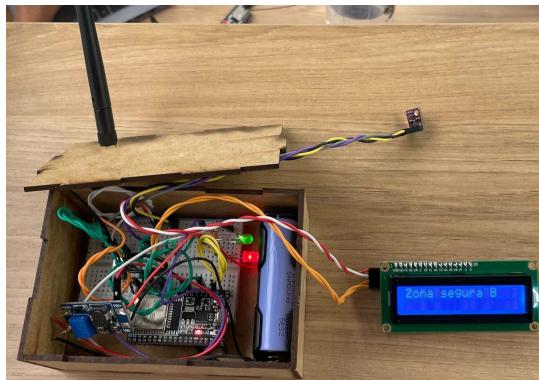


Imagen 9 : ESP32 do *Host*.

1. Insira os componentes eletrônicos em contato com a *protoboard*.

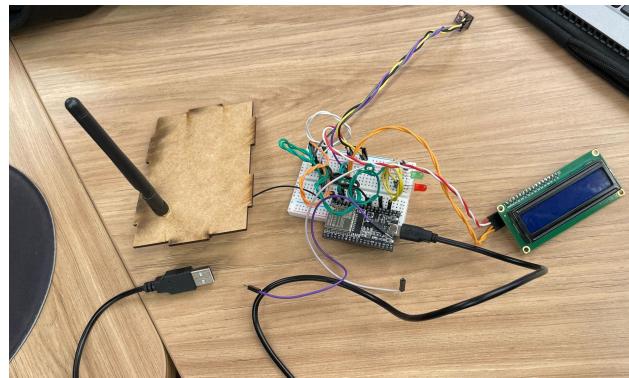


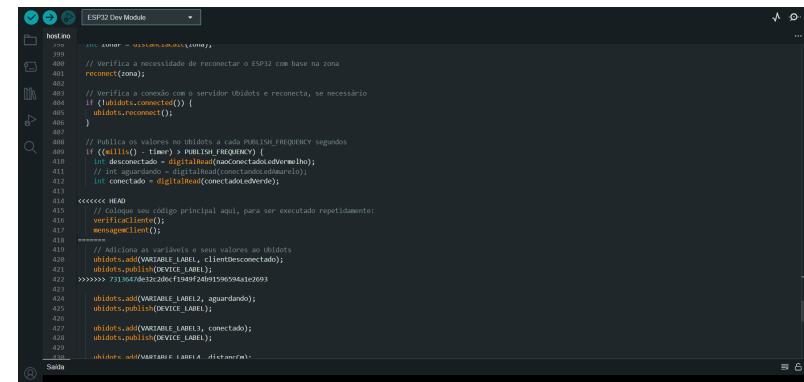
Imagen 10: Inserindo os componentes em contato com a *protoboard*.

2. Coloque todo o protótipo na caixa de uso.



Imagen 11: Colocando o protótipo na caixa de uso

3. Utilize o Arduino IDE em conjunto com o código-fonte disponibilizado e faça testes progressivos.



```

host.ino
/*
  auto const = MEDIUMLOUDNESS;
}

// Verifica a necessidade de reconectar o ESP32 com base na zona
reconnect(zona);

// Verifica a conexão com o servidor Ubidots e reconecta, se necessário
if (!ubidots.connect()) {
    ubidots.reconnect();
}

// Publica os valores no Ubidots a cada PUBISH_FREQUENCY segundos
if ((digitalRead(desconectado) == 0) & (digitalRead(conectado) == 1)) {
    desconectado = digitalRead(unosconectadosdesconectado);
    // Int. aguardando o digitalRead(conectado, desconectado);
    int conectado = digitalRead(conectadoDesconectado);
}

<<<<<< HEAD
// Coloque seu código principal aqui, para ser executado repetidamente:
<<<<<< BODY
// Adicione as variáveis e suas telas no Ubidots
ubidots.add(VARIABLE_LABEL1, clientDesconectado);
ubidots.publish(DEVICE_LABEL);
>>>> 7316d470e32206c1f549d2db1950594a1c2693
ubidots.add(VARIABLE_LABEL2, aguardando);
ubidots.publish(DEVICE_LABEL);
ubidots.add(VARIABLE_LABEL3, conectado);
ubidots.publish(DEVICE_LABEL);
ubidots.setVariableLABELA(discoverewm);

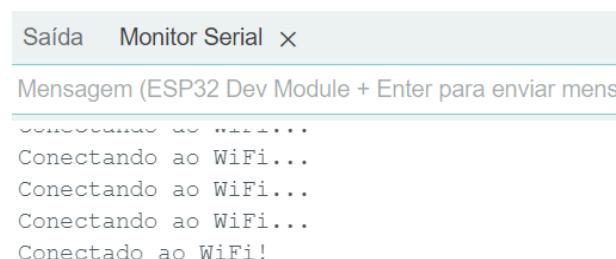
```

Imagen 12: Utilizando o Arduino IDE para programação e testes.

### 3. Guia de Instalação

#### **Passo 1 - Instalação do *Host* de Comunicação:**

- Escolha o local adequado na área fabril para instalar o *Host* responsável pela comunicação com os dispositivos ESP32. É importante considerar uma posição centralizada e de fácil acesso para garantir uma cobertura eficiente. Certifique-se de seguir as diretrizes de segurança e instalação específicas do local.
- Conecte o *Host* à fonte de energia adequada e aguarde até que ele se conecte à rede Wi-Fi da PIRELLI. Isso pode ser feito ligando o *Host* e permitindo que ele se conecte automaticamente à rede.



```

Saída Monitor Serial X
Mensagem (ESP32 Dev Module + Enter para enviar mens:
Conectando ao WiFi...
Conectando ao WiFi...
Conectando ao WiFi...
Conectado ao WiFi!

```

Imagen 15: *Host* de Comunicação conectado à rede Wi-Fi da PIRELLI.

#### **Passo 2 - Instalação dos Dispositivos ESP32 (*Client*):**

- Após o *Host* estar conectado à rede Wi-Fi da PIRELLI, ligue os dispositivos ESP32 *Client* em diferentes áreas da fábrica.

Certifique-se de seguir as instruções do fabricante para ligar corretamente os dispositivos.

- Os dispositivos ESP32 *Client* irão procurar automaticamente pelo servidor criado pelo Host e estabelecerão uma conexão com ele.
- Percorra a área da fábrica e registre a localização de cada roteador Wi-Fi desejado. Isso ajudará a criar pontos de referência para o sistema de monitoramento.
- Com base nessas localizações, crie pontos de referência para a configuração do sistema.

#### **Passo 3 - Configuração Concluída:**

- Após a conclusão dos passos anteriores, a instalação estará pronta para uso.
- Através da interface web Ubidots, o responsável pelo monitoramento terá acesso à localização do dispositivo móvel em tempo real.
- O sinal Wi-Fi e os pontos de referência configurados serão utilizados para monitorar e rastrear os dispositivos móveis dentro da área da fábrica.

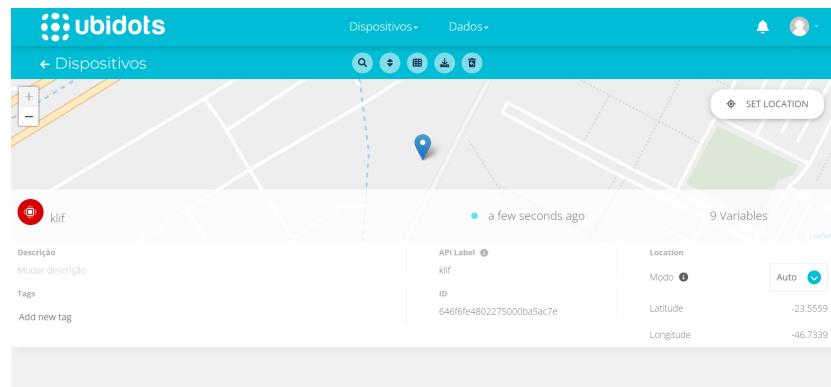


Imagen 16: Interface web Ubidots exibindo a localização do dispositivo móvel.

## 4. Guia de Configuração

Para realizar a configuração, siga os seguintes passos:

1. Conecte o protótipo ao seu tablet/notebook utilizando uma porta USB.
2. Abra o aplicativo "Arduino IDE" no seu tablet/notebook. Caso você não tenha o Arduino IDE instalado, você pode fazer o download do software em <https://www.arduino.cc/en/software>.
3. Verifique a porta correta para a conexão do protótipo. No Arduino IDE, siga os seguintes passos:
  - a. Clique em "Ferramentas" (*Tools*, em inglês) na barra de menu superior.
  - b. Selecione a opção "Porta" (*Port*, em inglês) e verifique qual porta está disponível para a conexão do protótipo. Geralmente, ela será identificada como "COMX" (em Windows) ou "/dev/ttyX" (em macOS/Linux), onde "X" é um número.
  - c. Anote o nome da porta selecionada, pois você precisará dele posteriormente.
4. Ligue o protótipo, garantindo que ele esteja conectado ao tablet/notebook por meio da porta USB.

5. Acesse o seguinte link no seu navegador: <https://profgodoiswk.iot.ubidots.com/app/dashboards/646babe6e15600ead1436fb6>.

(Observação: Para acessar o link, é necessário ter as credenciais corretas. *Username*: autobots *Password*: inteliGrupo2)

6. Nesse link, você poderá visualizar o *dashboard* que será conectado diretamente ao código do protótipo. Ele fornece uma representação visual dos dados coletados e controla as ações relacionadas ao sistema.

(A imagem abaixo mostra uma captura de tela do *dashboard* para ilustrar sua aparência e funcionalidades, oferecendo uma visão geral do que será apresentado quando você acessar o link.)

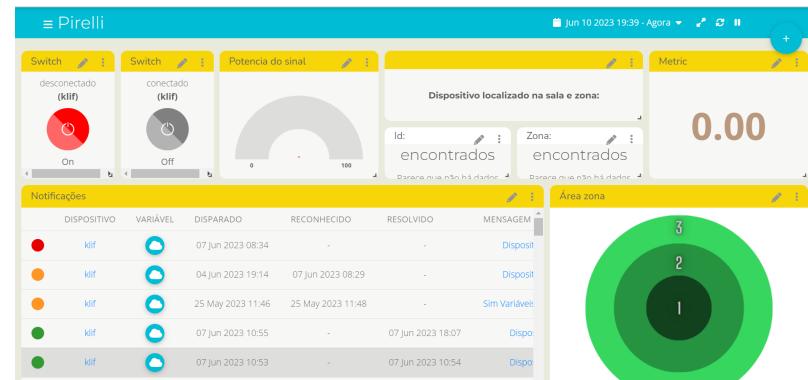


Imagen 17: Página inicial do *dashboard* no Ubidots.

## 5. Guia de Operação

O projeto utiliza a plataforma Ubidots, uma solução baseada em nuvem para Internet das Coisas (IoT), que permite coletar, armazenar, processar e visualizar dados de dispositivos conectados. Essa plataforma abrangente simplifica o desenvolvimento de aplicativos IoT, oferecendo recursos para coleta de dados de sensores, controle remoto de dispositivos e criação de painéis personalizados. O Ubidots é flexível e escalável, suportando diversos dispositivos e protocolos de comunicação. O acesso à plataforma é feito por meio do link fornecido anteriormente.

Ao acessar a plataforma Ubidots, o usuário terá acesso a um *dashboard*, conforme mostrado na imagem 17. Nesse painel, o usuário poderá verificar várias informações relevantes. Por exemplo:

**Conexão do dispositivo:** É possível verificar se o dispositivo está conectado à rede da empresa.



Imagen 18: Exemplo de dispositivo desconectado.



Imagen 19: Exemplo de dispositivo conectado.

**Localização do dispositivo:** É apresentada a sala (ID) e a zona em que o dispositivo está situado. A sala se refere a um espaço específico dentro de uma fábrica, enquanto a zona é uma área mais ampla que engloba várias salas. Por exemplo, a Sala 101 pode estar situada na Zona de Produção 01.



Imagen 20: Exemplo de dispositivos localizados em sala e zona.

**Potência do sinal:** A potência do sinal é exibida para determinar a localização mais próxima do dispositivo.



Imagen 21: Exemplo de exibição da potência do sinal.

**Histórico de desconexões:** É possível visualizar a quantidade de vezes que o dispositivo se desconectou da rede em um intervalo pré-definido pelo usuário (por exemplo, uma semana).

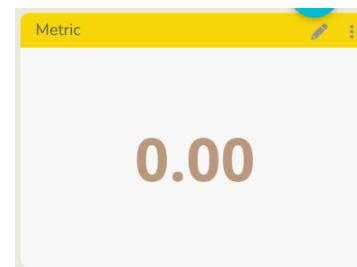


Imagen 22: Exemplo de histórico de desconexões.

Além disso, na aba "Dados" do *dashboard*, o usuário tem a opção de programar eventos. Quando esses eventos ocorrem, uma notificação pré-programada é enviada por meio das ações disponíveis, como mostrado na imagem 23. Essas ações permitem ao usuário definir notificações personalizadas para serem acionadas em situações específicas, proporcionando maior controle e gerenciamento dos dispositivos conectados.

	Enviar correio electrónico
	Enviar SMS
	Chamada de voz
	Atribuir uma variável
	Folga
	Trigger Webhook

Imagen 23: Funções da aba Dados no *dashboard*.

## 6. Troubleshooting

Aqui estão algumas situações de falha comuns relacionadas ao dispositivo KLIF, que usa MQTT para enviar dados para a plataforma Ubidots na nuvem. Ações recomendadas para solucionar esses problemas estão incluídas. Observe que este documento não abrange todos os componentes, mas se concentra nos sintomas e métodos para identificar a possível origem dos problemas relacionados ao uso do dispositivo. Esses problemas podem ser devido a problemas de software ou configuração do dispositivo.

#	Problema	Possível solução
1	Perda de conexão do <i>Host</i> com o WiFi	Caso o próprio software não consiga restabelecer a conexão, recomenda-se que reinicie o dispositivo através do botão de reset.
2	Perda de conexão do <i>Client</i> com o WiFi	Caso o próprio software não consiga restabelecer a conexão, recomenda-se que reinicie o dispositivo através do botão de reset.
3	Falha no envio de dados	Verifique se as credenciais de

	para a plataforma Ubidots	autenticação da plataforma estão corretas e se o dispositivo está conectado à internet. Caso necessário, verifique as configurações de rede do dispositivo.
4	Falha na coleta de dados	Verifique se os sensores estão corretamente conectados e configurados.
5	Erro de autenticação MQTT	Verifique se as credenciais de autenticação MQTT estão corretas e se o dispositivo está configurado corretamente para se comunicar com o servidor MQTT. Caso necessário, atualize as configurações de autenticação.
6	Erro de porta (Imagens 24 e 25)	Para solucionar esse erro, escolha a porta adequada para o ESP32.

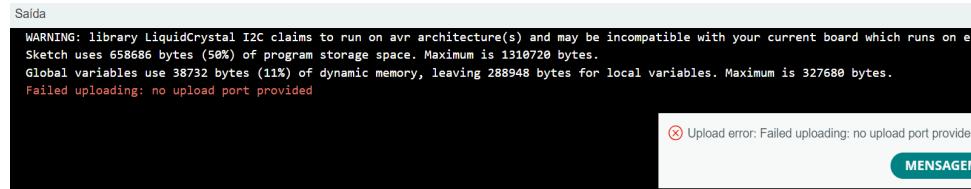


Imagen 24 (problema 6): Erro de porta.

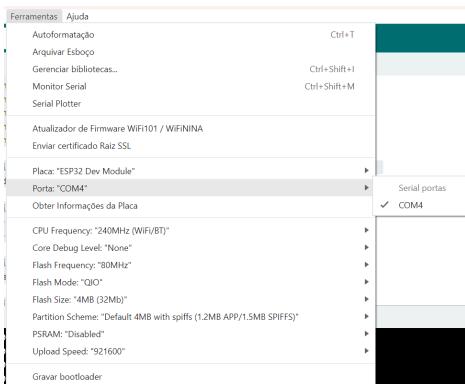


Imagen 25 (problema 6): Correção do erro de porta.