

Manual de Instruções

**CÉLULA DE CARGA - GRUPO ASIMOV
IPT - SEÇÃO OBRAS CIVIS**

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
27/11/2023	Lídia Mariano e Keylla Oliveira	4.1	Criação do documento
30/11/2023	Lídia Mariano	4.2	Preenchimento da seções 1 e 6
6/12/2023	Lídia Mariano	4.4	Preenchimento da seção 4
8/12/2023	Lídia Mariano	4.5	Realização da seção 2 da documentação e refinação da seção 4.
9/12/2023	Lídia Mariano	4.6	Refinamento das seções 1 e 4.
10/12/2023	Keylla	4.7	Realização da seção 5
10/12/2023	Fernando Vasconcellos	4.8	Adição de textos instrutivos para a criação de conta nas plataformas

			utilizadas
18/12/2023	Fernando Vasconcellos	5.1	Alteração no espaçamento, parágrafos e organização das imagens.
19/12/2023	Fernando Vasconcellos	5.2	Alteração na seção 2 (Guia de montagem) e correção dos feedbacks passados no artefato na seção 4 (Guia de instalação)
19/12/2023	Lídia Mariano	5.3	Correção de ajustes requisitados no feedback do artefato, em especial de formatação de imagens e a falta de texto em algumas delas. Assim como a realização do sumário.
19/12/2023	Fernando Vasconcellos	5.4	Realização da seção 3 como um todo.

Índice

1. Componentes e Recursos	3
1.1. Componentes externos	3
1.2. Requisitos de conectividade	3
2. Guia de Montagem	7
2.1. Guia de montagem com case	8
2.1 Guia de montagem alternativo	9
3. Guia de Instalação	15
3.1 Componentes e suas respectivas ligações	15
4. Guia de Configuração	16
5. Guia de Operação	25
6. Troubleshooting	28

1. Componentes e Recursos

1.1. Componentes externos

- Dispositivos:

- Microcontrolador DOIT ESP32 DEVKIT v1: Placa de desenvolvimento para IoT com recursos de conectividade Wi-Fi e Bluetooth, atuando como o nó central de coleta e envio de dados.



-Antena Wi-Fi: Dispositivo que melhora a recepção e transmissão de sinais Wi-Fi, promovendo uma comunicação mais estável e eficiente para dispositivos conectados.



-Módulo leitor MICRO SD: Componente que permite a leitura de cartões de memória no formato microSD, facilitando o armazenamento e recuperação de dados coletados.



-Cartão MICRO SD 64GB: Cartão de memória removível com capacidade de armazenamento de 64 gigabytes, utilizado para armazenar e acessar grandes volumes de dados em dispositivos eletrônicos.



-Buzzer: Dispositivo sonoro que emite sons ou alertas sonoros para fornecer feedback quando a célula de carga medir um peso maior que 4kg.



-Bateria 4 volts: Fonte de energia portátil com uma tensão de 4 volts, utilizada para alimentar o dispositivo.



- Push Button: Botão de pressão momentânea utilizado para enviar um sinal de controle quando pressionado, capaz de iniciar a balança, pausar a leitura ou fazer a tara da medição.



- Chave HH: Integrada como um dispositivo de entrada para controlar funções como ligar/desligar a passagem de energia da bateria.



- Sensor BME280: Sensor de alta precisão utilizado para medições de temperatura, umidade e pressão atmosférica.



- Conversor HX711: Amplificador de instrumentação essencial para a leitura precisa de sinais provenientes da célula de carga.



- Célula de Carga: Dispositivo sensor que mede a carga aplicada em uma estrutura, fundamental para aplicações de monitoramento estrutural.



-Display LCD(16x2): Responsável por indicar para o usuário a leitura da célula de carga e do BME280 em tempo real.

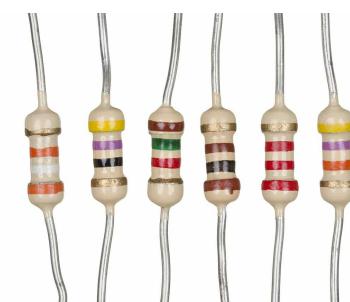


- LEDs: Componentes visuais utilizados para indicar estados específicos ou condições, oferecendo feedback visual. LED vermelho mostra quando a célula de carga mede um objeto

que ultrapassa 4kg e o LED RGB pode indicar diferentes cenários que o dispositivo pode enfrentar funcionamento perfeito: verde; sem internet: amarelo; célula de carga desconectada: vermelho; sensor não encontrado: roxo; LCD não encontrado: branco.



- Jumpers e Resistores: Componentes elétricos essenciais para realizar conexões e ajustes na montagem do circuito.



- Comunicação:

- Protocolo MQTT: Protocolo de comunicação leve e eficiente, ideal para a troca de mensagens entre os

dispositivos, usado para a comunicação entre o ESP32 e o servidor MQTT (broker).

- Broker MQTT (Ubidots): Plataforma Ubidots que atua como intermediário para receber, processar e encaminhar mensagens MQTT, facilitando a integração com a nuvem.

- Plataforma em Nuvem:

- Ubidots: Plataforma IoT baseada em nuvem que oferece serviços para armazenamento, visualização e análise em tempo real dos dados provenientes dos dispositivos conectados.

- Ferramentas de Desenvolvimento:

- Ambiente de Desenvolvimento Integrado (IDE) Arduino: Software utilizado para escrever, compilar e fazer upload de código para o ESP 32, oferecendo uma interface amigável para o desenvolvimento.

- Ferramenta de Edição de Código: Pode incluir editores de texto como Visual Studio Code, Sublime Text ou o próprio Arduino IDE, utilizados para a escrita e edição eficiente do código-fonte.

- Outros Componentes Opcionais:

- Servidores Web Locais ou em Nuvem: Componentes adicionais que podem ser utilizados para interfaces específicas ou serviços complementares.

- Banco de Dados: Se necessário, um componente para o armazenamento persistente de dados fora da plataforma, permitindo maior flexibilidade de gerenciamento de dados.

- Ferramentas de Monitoramento e Depuração: Utilizadas para acompanhar o desempenho do sistema e solucionar problemas durante o desenvolvimento, garantindo a integridade e eficiência da solução IoT.

1.2. Requisitos de conectividade

1.2.1 Rede Local (LAN):

A solução IoT é configurada para operar em uma rede local, estabelecendo uma conexão direta com o roteador ou ponto de acesso Wi-Fi disponível no ambiente. Isso possibilita a comunicação eficiente entre os dispositivos IoT e outros. O microcontrolador ESP 32, que integra recursos Wi-Fi, utiliza o protocolo Wi-Fi padrão para se conectar à rede local. Isso viabiliza a comunicação sem fio entre o ESP 32 e outros dispositivos conectados à mesma rede.

1.2.2 MQTT (Message Queuing Telemetry Transport):

O protocolo MQTT é empregado para a troca de mensagens entre os dispositivos IoT, como os sensores BME280, HX711, Célula de Carga, e o servidor MQTT (broker). Ele é escolhido por sua leveza e eficiência na transmissão de dados.

1.2.3 Broker MQTT (Ubidots):

O broker MQTT atua como intermediário crucial entre os dispositivos IoT e a plataforma em nuvem. Neste caso, o Ubidots é o back-end responsável por receber, processar e armazenar os dados transmitidos pelos dispositivos por meio do protocolo MQTT.

1.2.4 Protocolo I2C:

O protocolo I2C (Inter-Integrated Circuit) é uma forma de comunicação serial síncrona utilizada para transferir dados entre dispositivos em um sistema eletrônico. No projeto, o I2C é aplicado para facilitar a comunicação entre o microcontrolador ESP32, que age como o nó central, e dispositivos como o display LCD e o sensor de temperatura (BME280). Essa abordagem minimiza a quantidade de fios necessários e otimiza a integração, permitindo que o microcontrolador envie comandos e receba dados de forma eficiente desses dispositivos.

1.2.5 Plataforma em Nuvem (Ubidots):

Além de funcionar como back-end, o Ubidots oferece serviços de plataforma em nuvem, proporcionando armazenamento

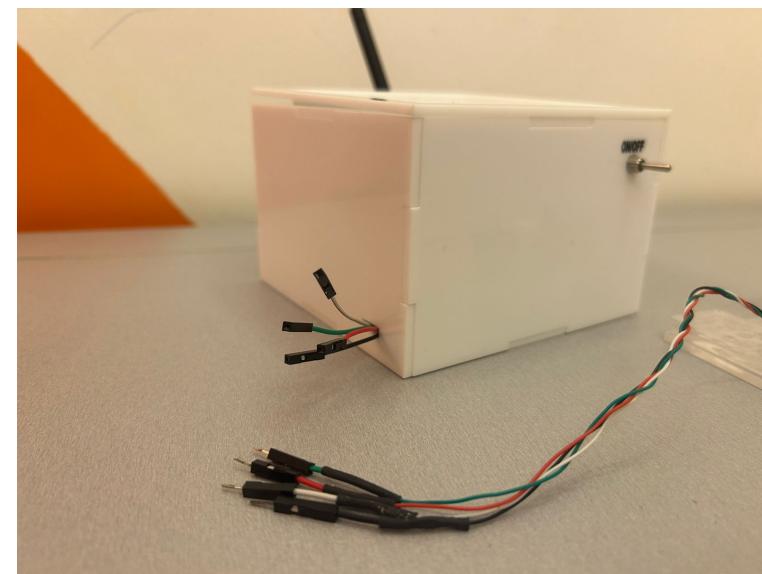
seguro, visualização gráfica e análise em tempo real dos dados provenientes dos dispositivos IoT.

2. Guia de Montagem

2.1 Guia de Montagem com case

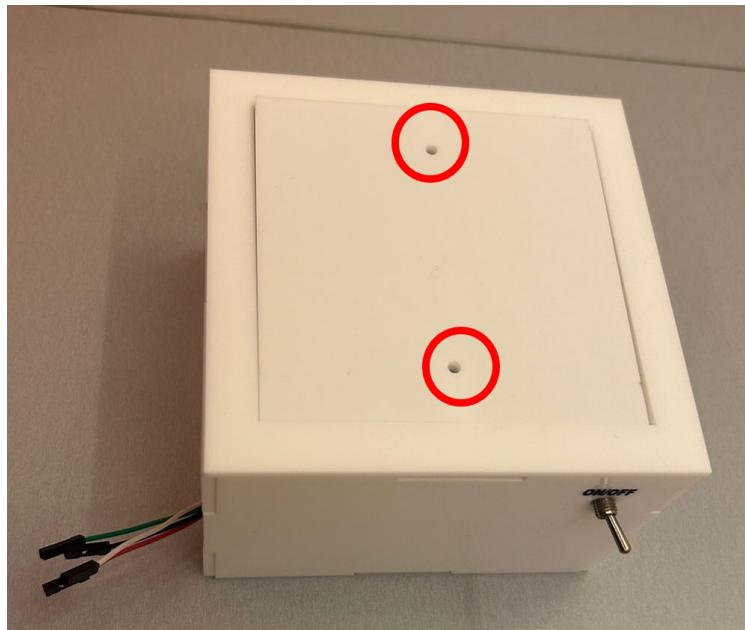
PASSO 1: Instalação da célula de carga no protótipo

Deve ser feita a conexão da célula de carga com o HX711. A cor de cada fio exposto deve coincidir com a cor dos fios da célula de carga.



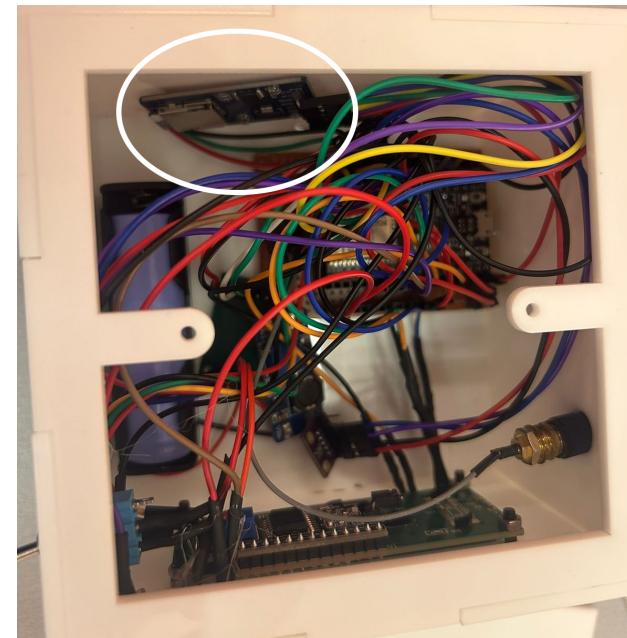
PASSO 2: Soltar os parafusos da case

Deve ser removidos os parafusos, indicados pelo círculo, da case para que se obtenha acesso aos equipamentos que estão no interior



PASSO 3: Configuração do cartão de memória

O cartão de memória deve ser inserido no espaço destacado na imagem.



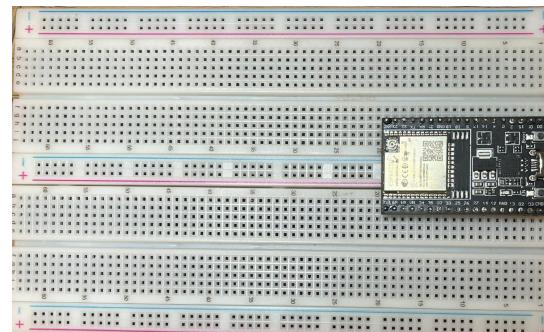
PASSO 4: Instalar a bateria no protótipo

É necessário instalar a bateria no espaço configurado para ela destacado na imagem.

2.2 Guia de Montagem alternativo

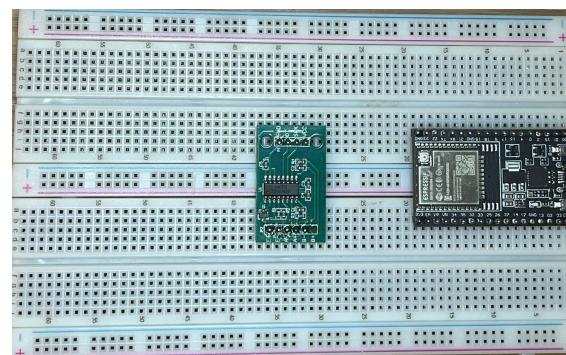
PASSO 1: Instalação do ESP32

O ESP32 deve ser encaixado na protoboard:



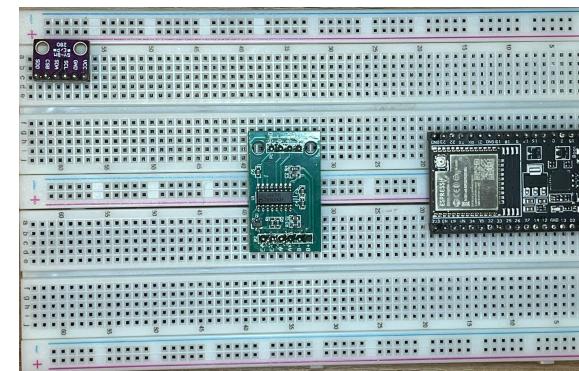
PASSO 2: Instalação do HX711

O HX711 deve também ser encaixado na protoboard próximo ao ESP32



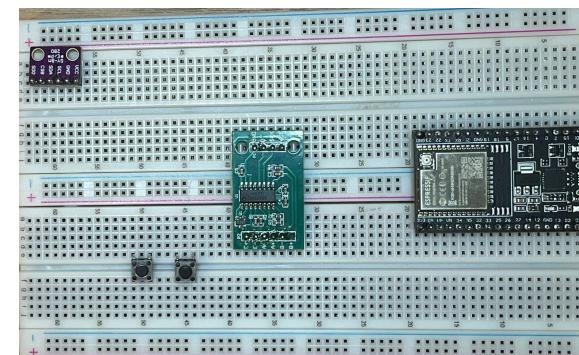
PASSO 3: Instalação do BME280

O BME280, sensor de temperatura, deve ser encaixada na protoboard



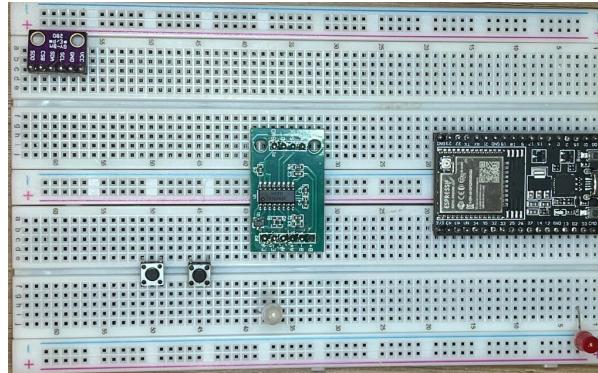
PASSO 4: Instalação dos botões

Os botões de iniciar/parar a leitura e o botão de tara devem ser encaixados na protoboard



PASSO 5: Instalação dos LEDs

Os LEDs devem ser encaixados na protoboard



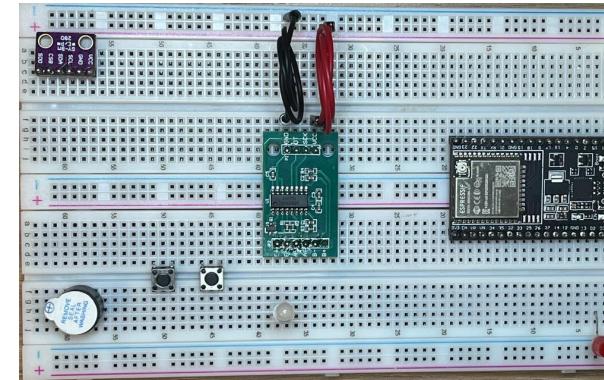
PASSO 6: Instalação do buzzer

O buzzer deve ser encaixado na protoboard

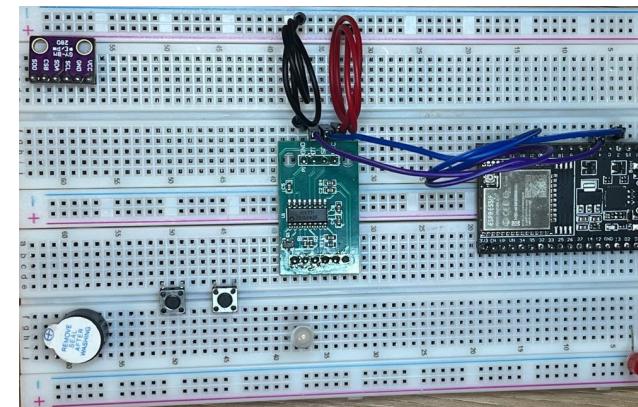


PASSO 7: Conexões do HX711

- VCC: VCC - 3.3V (Positivo) **Fio vermelho**
- GND: Ground (Negativo) **Fio preto**

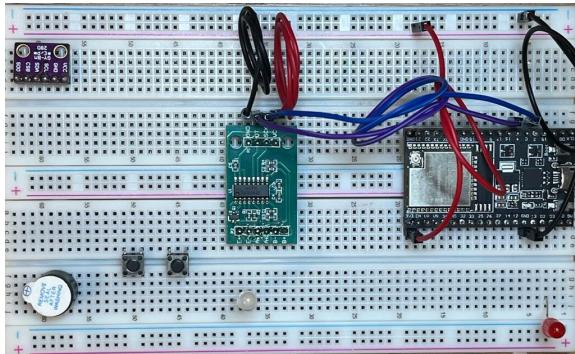


- DT: Porta do esp32 - 4 **Fio roxo**
- SCK: Porta do esp32 - 2 **Fio azul**

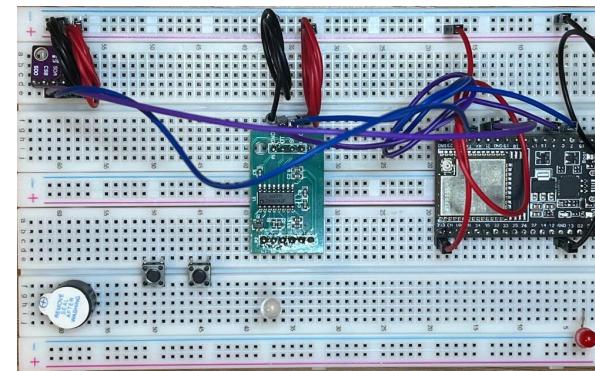


PASSO 8: Conexões do ESP32

- Porta GND: Ground(Negativo) Fio preto
- Porta 3.3V: Ground(Positivo) **Fio vermelho**

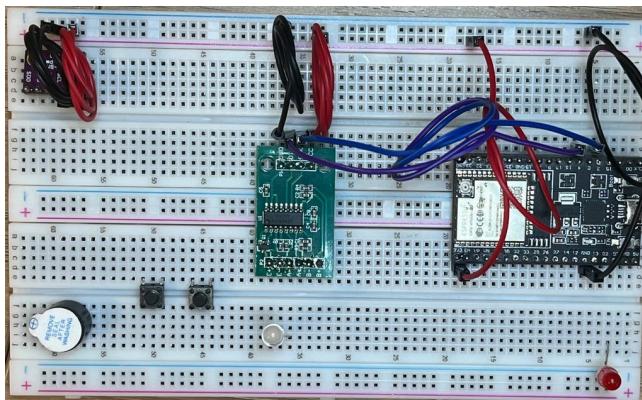


- SDA: Porta do esp32 - 21 Fio roxo
- SCL: Porta do esp32 - 22 Fio azul



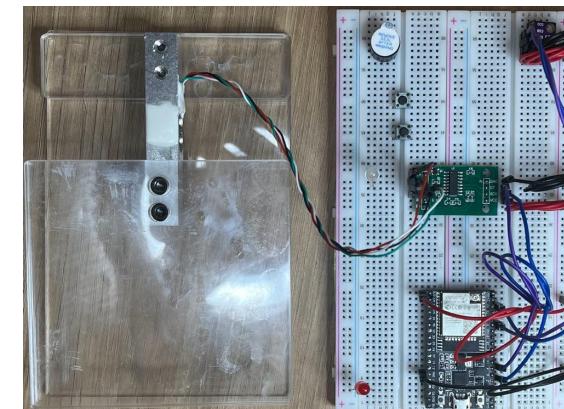
PASSO 9: Conexões do BME280

- GND: Ground (Negativo) Fio preto
- VCC: VCC - 3.3V (Positivo) **Fio vermelho**



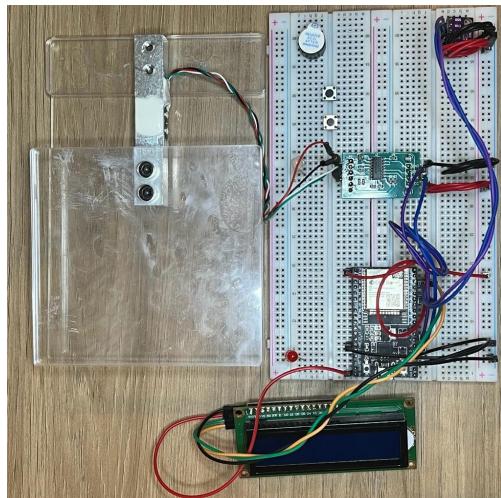
PASSO 10: Conexões da célula de carga

- **Fio verde:** Porta A + do HX711
- **Fio branco:** Porta A - do HX711
- **Fio preto:** Porta E - do HX711
- **Fio vermelho:** Porta E+do HX711



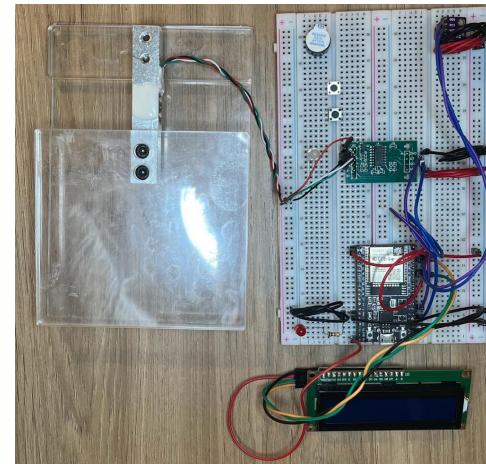
PASSO 11: Conexões do LCD

- VCC: VCC - 3.3V (Positivo) **Fio vermelho**
- GND: Ground (Negativo) Fio preto
- SDA: Porta do esp32 - 21 Fio verde
- SCL: Porta do esp32 - 22 **Fio laranja**



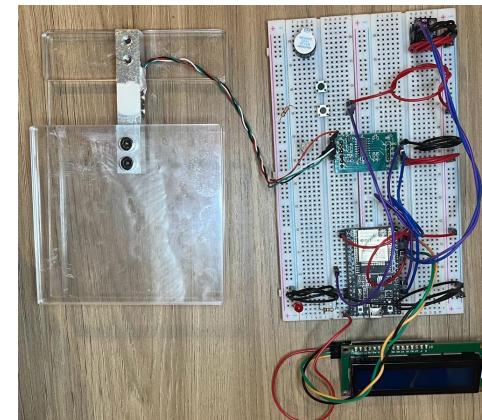
PASSO 12: Conexões do LED Vermelho

- Perna maior: conectar o resistor
- Perna menor: conectar o jumper **Fio vermelho**



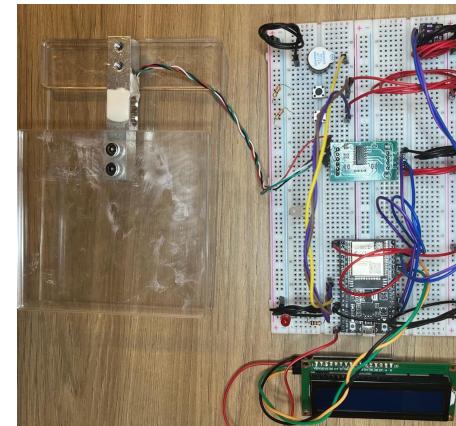
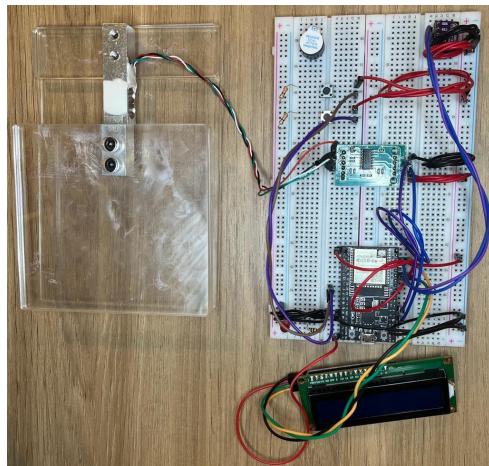
PASSO 13: Conexões do botão de iniciar/parar leitura

- Porta Digital: Porta do esp32 - 25 **Fio roxo**
- Porta Positiva: VCC - 3.3V (Positivo) **Fio vermelho**
- Porta Negativa: GND (Negativo) Fio preto



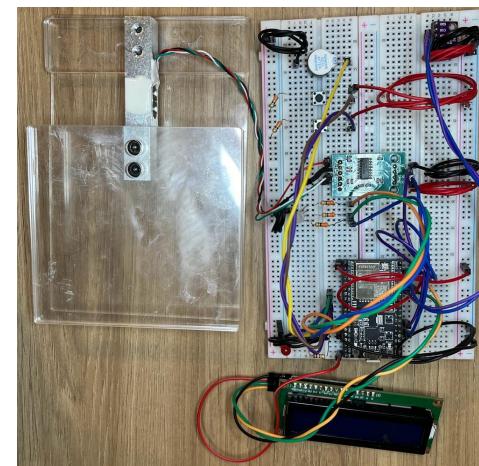
PASSO 14: Conexões do botão de tara

- Porta Digital: Porta do esp32 - 26 Fio marrom
- Porta Positiva: VCC - 3.3V (Positivo) Fio vermelho
- Porta Negativa: GND (Negativo) Fio preto



PASSO 16: Conexões do LED RGB

- Fio preto: Ground(negativo)
- Fio azul: porta do esp32 - 33
- Fio verde: porta do esp32- 32
- Fio laranja: porta do esp32 - 13

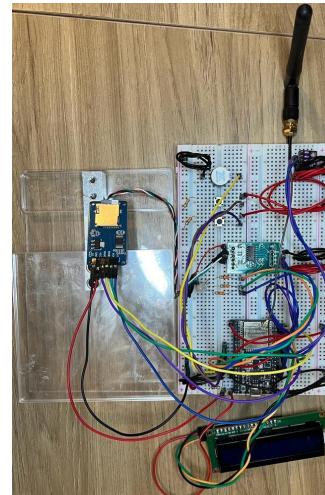
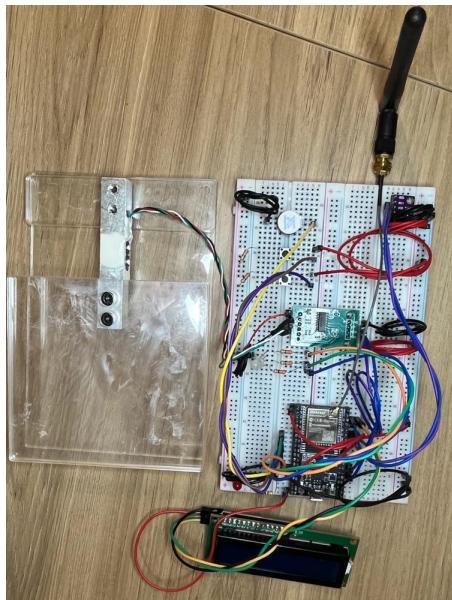


PASSO 15: Conexões do buzzer

- Fio amarelo: porta do esp32 - 27
- Fio preto: Ground(negativo)

PASSO 17: Conexão da antena

A antena WiFi deve ser conectada ao ESP32 no conector U.FL

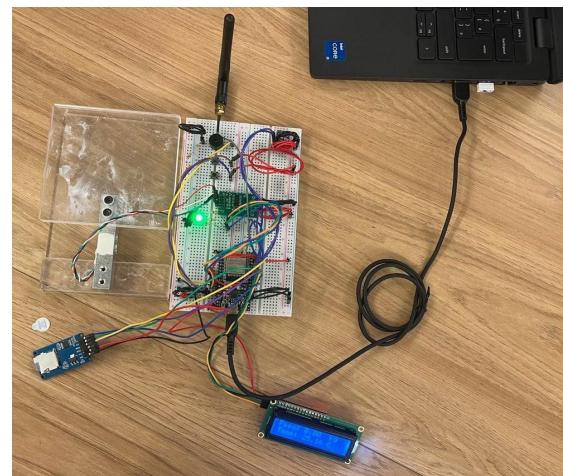


PASSO 18: Conexão do cartão SD

- GND: Ground (negativo) Fio preto
- VCC: Ground(positivo) **Fio vermelho**
- MOSI: porta do esp32 - 23 **Fio amarelo**
- MISO : porta do esp32 - 19 **Fio roxo**
- SCK: porta do esp32 - 18 **Fio verde**
- CS: porta do esp32 - 5 **Fio azul**

PASSO 19: Conectar o dispositivo a alimentação

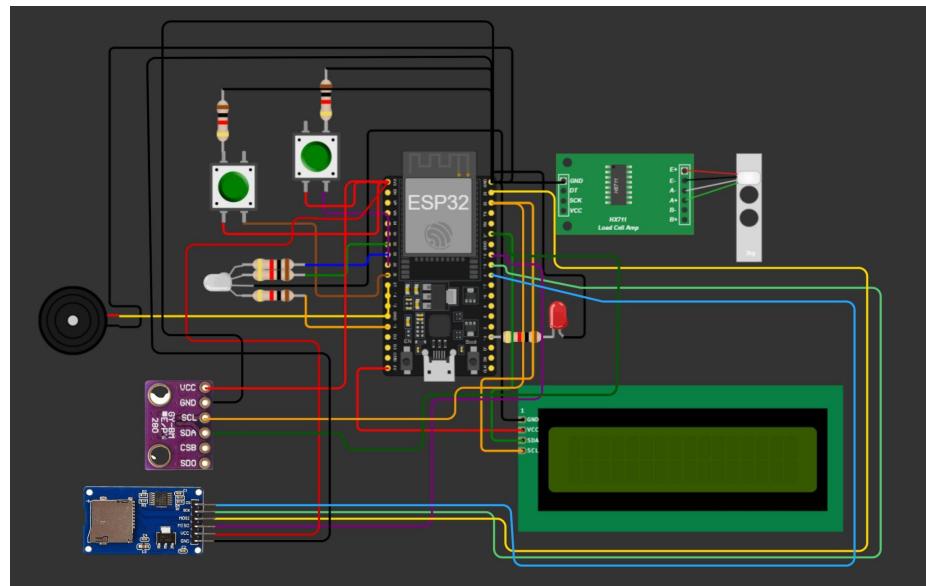
Um cabo USB deve ser conectado ao ESP32 e a alimentação, no caso da imagem, é um computador.



3. Guia de Instalação

A seção de guia de instalação foi substituída por um mapa de circuito do projeto, como solicitado pelo parceiro. Isso se deve ao fato de que a instalação do projeto varia de acordo com a estrutura civil que será instalada.

A imagem a seguir mostra todas as conexões feitas pelo microcontrolador ESP32 utilizado para a realização do projeto.



3.1 Componentes e suas respectivas ligações:

Cartão SD:

- GND: Ground (GND) Fio preto
- VCC: 3.3V (positivo) **Fio vermelho**
- MOSI: porta do esp32 - 23 **Fio amarelo**
- MISO : porta do esp32 - 19 **Fio roxo**
- SCK: porta do esp32 - 18 **Fio verde**
- CS: porta do esp32 - 5 **Fio azul**

BME280:

- GND: Ground (Negativo) Fio preto
- VCC: VCC - 3.3V (Positivo) **Fio vermelho**
- SDA: Porta do esp32 - 21 **Fio roxo**
- SCL: Porta do esp32 - 22 **Fio azul**

Buzzer:

- Lado positivo: porta do esp32 - 27 **Fio amarelo**
- Lado negativo: GND (Negativo) Fio preto

Led RGB:

- GND (Perna Maior): GND (negativo) Fio preto
- Canal Blue (Primeira perna da ponta oposta ao GND) porta do esp32 - 33 **Fio azul**

- Canal Green (Perna a esquerda do canal Blue): porta do esp32 - 32 **Fio verde**:
- Canal Green (Perna a esquerda do canal GND): porta do esp32 - 13 **Fio laranja**:

Botão de tara:

- Porta Digital: Porta do esp32 - 26 **Fio marrom**
- Porta Positiva: VCC - 3.3V (Positivo) **Fio vermelho**
- Porta Negativa: GND (Negativo) Fio preto + Resistor de 1K ohm no circuito

Botão de inicialização:

- Porta Digital: Porta do esp32 - 25 **Fio roxo**
- Porta Positiva: VCC - 3.3V (Positivo) **Fio vermelho**
- Porta Negativa: GND (Negativo) Fio preto + Resistor de 1K ohm no circuito

HX711:

- VCC: 3.3V (Positivo) **Fio vermelho**
- GND: GND (Negativo) Fio preto
- DT: Porta do esp32 - 4 **Fio roxo**
- SCK: Porta do esp32 - 2 **Fio azul**

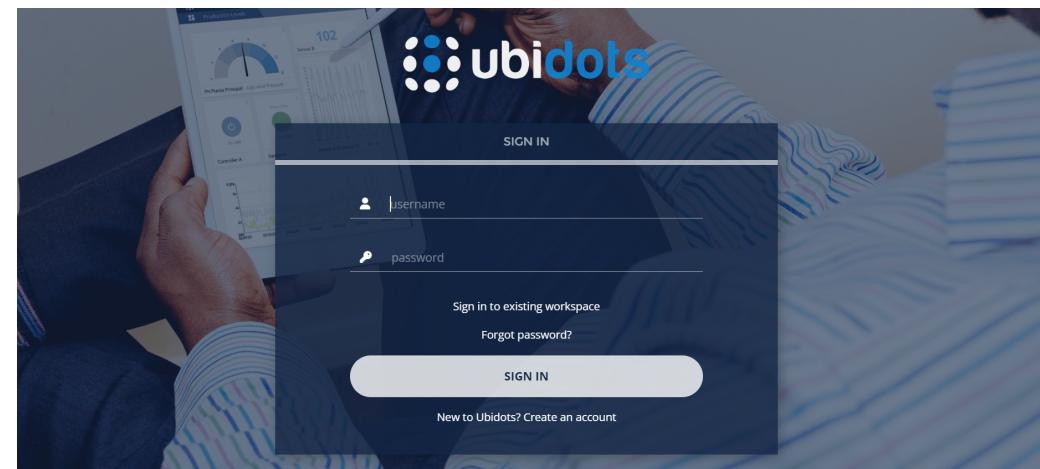
Led Vermelho:

- Perna maior (Catodo): conectar um resistor e ligar ao GND - Fio preto
- Perna menor (Anodo): Porta do esp32 - 15 **Fio vermelho**

4. Guia de Configuração

PASSO 1: Autenticação do usuário

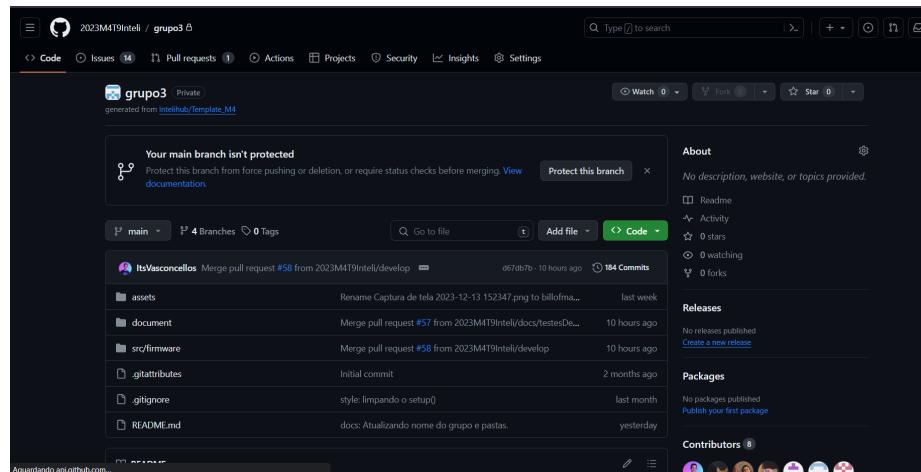
O usuário deve realizar o login na plataforma Ubidots, colocando “username” e “password” cadastrados. O link para a plataforma é: <https://industrial.ubidots.com/accounts/signin/>



PASSO 2: Baixando o projeto

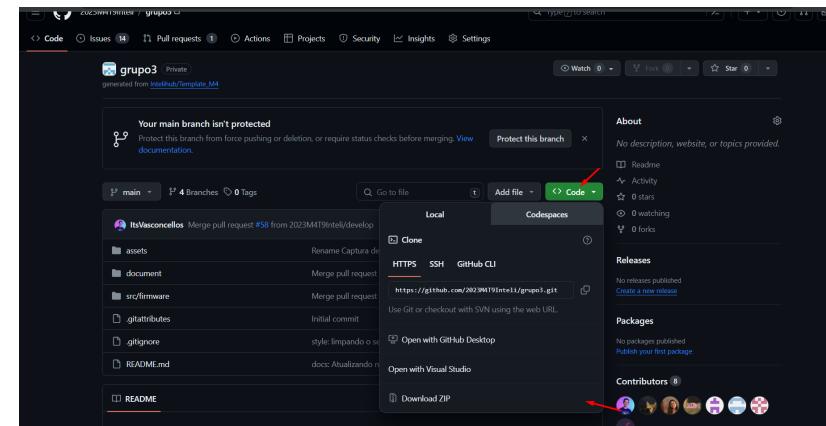
2.1 Entrando no git:

Deve acessar o seguinte link
<https://github.com/2023M4T9Inteli/grupo3/>. O mesmo deve se assemelhar a essa captura de tela:



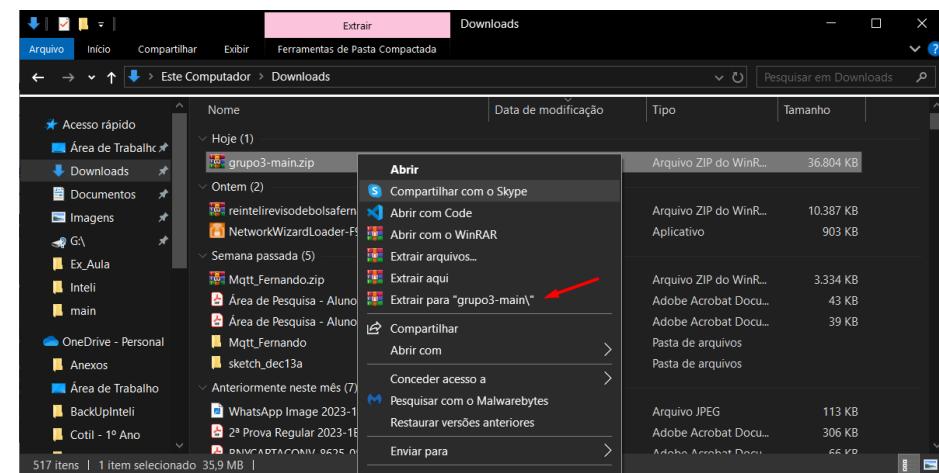
2.2 Baixar o projeto:

Clique no botão verde escrito “Code” e clique em “Download zip”
 após isso:



2.3 Extraiendo o zip:

Após baixar o arquivo, entre na pasta download do computador e clique com o botão direito, em cima do arquivo, e selecione a opção de extrair zip:



PASSO 3: Configuração da WiFi

3.1. Caminho do arquivo:

Configurar a WiFi do local dentro do código de configurações gerais do protótipo no arquivo “main.ino”, presente dentro da pasta “src” do projeto.

Nome	Data de modificação	Tipo	Tamanho
.git	01/12/2023 15:52	Pasta de arquivos	
assets	30/11/2023 09:49	Pasta de arquivos	
document	30/11/2023 09:49	Pasta de arquivos	
outros	21/11/2023 08:30	Pasta de arquivos	
src	21/11/2023 15:23	Pasta de arquivos	
.gitattributes	Data da criação: 30/10/2023 15:08 Tamanho: 245 KB Pastas: vscode, firmware Arquivos: readme.md	Arquivo Fonte Git Att...	1 KB
.gitignore	2023 15:08 2023 08:30 Arquivos: README.md	Arquivo Fonte Git Ig...	1 KB
README.md	30/10/2023 15:08	Arquivo Fonte Markd...	4 KB

, a qual está dentro da pasta “firmware”

Nome	Data de modificação	Tipo	Tamanho
.vscode	21/11/2023 15:28	Pasta de arquivos	
firmware	21/11/2023 08:30	Pasta de arquivos	
readme.md	Data da criação: 21/11/2023 08:30 Tamanho: 225 KB Pastas: include, libraries, main	Arquivo Fonte Markd...	1 KB

que por sua vez estava dentro da pasta “main”.

Nome	Data de modificação	Tipo	Tamanho
include	26/11/2023 21:43	Pasta de arquivos	
libraries	21/11/2023 17:28	Pasta de arquivos	
main	30/11/2023 09:49	Pasta de arquivos	

Data da criação: 21/11/2023 08:30
Tamanho: 8,21 KB



Por fim, acesse o arquivo “main.ino” pelo arduino IDE.

Nome	Data de modificação	Tipo	Tamanho
ConfigMQTT.cpp	30/11/2023 09:49	Arquivo Fonte C++	2 KB
InterfacelCD.cpp	25/11/2023 16:36	Arquivo Fonte C++	1 KB
LEDs.cpp	26/11/2023 21:43	Arquivo Fonte C++	2 KB
main.ino	26/11/2023 21:43	Arquivo INO	2 KB
Operad...	26/11/2023 21:43	Arquivo Fonte C++	3 KB



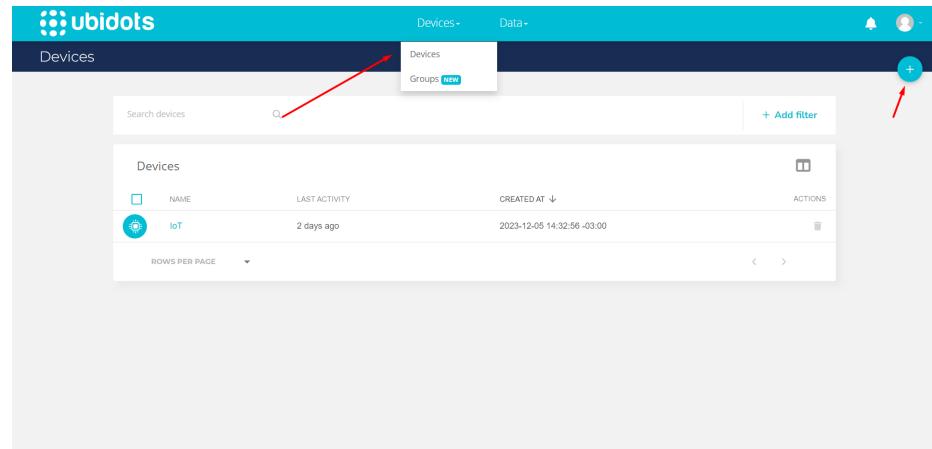
3.2. Alteração do código:

No arquivo “main.ino”, deverá ser alterado ,na linha 10, a variável “ssid” pelo nome da rede local, e, na linha 11, a variável “password” pela senha da WiFi da rede local.

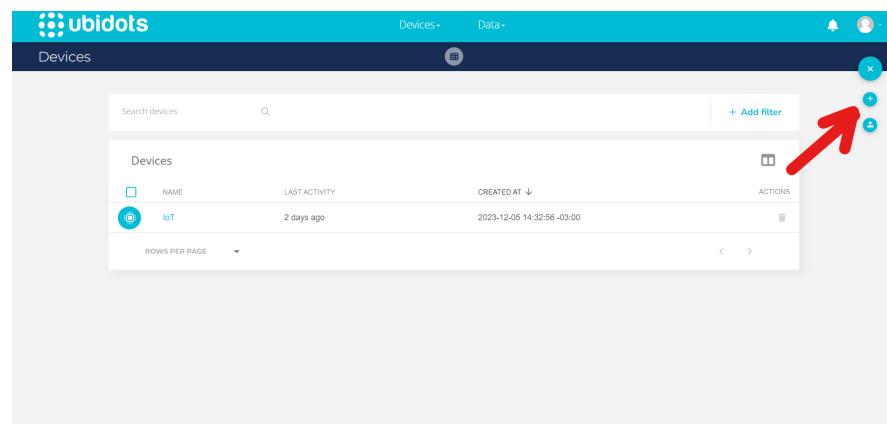
```
#include "../include/Operador.h"
#include "../include/ConfiguracaoOperador.h"

ConfiguracaoOperador config = {
    .limitePeso = 4, // Constante para o limite de peso

    // Configurações para o MQTT
    .server = WiFiServer(80), // Servidor WiFi
    .ssid = "", // SSID da rede WiFi
    .password = "", // Senha da rede WiFi
    .token = "BBUS-QJeJwahdOURDJQrU7RFv4danfkQx9d", // Token do Ubidots
    .deviceLabel = "esp32", // Label do dispositivo no Ubidots
}
```



Após o clique, será mostrado outro botão com um símbolo de adição, o qual deve ser clicado.

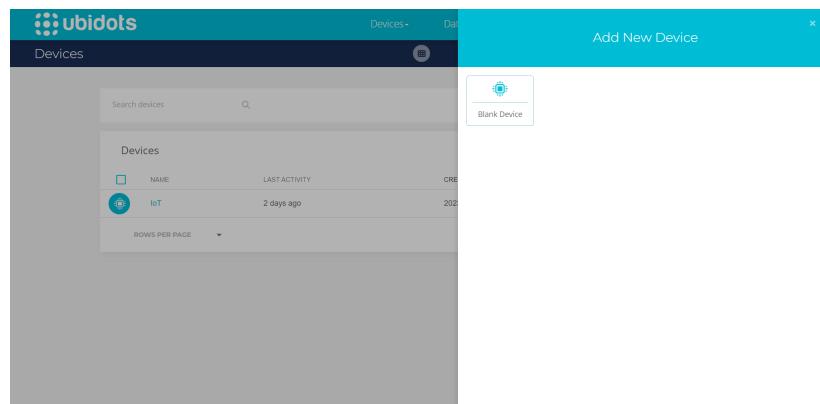


PASSO 4: Configuração do Ubidots

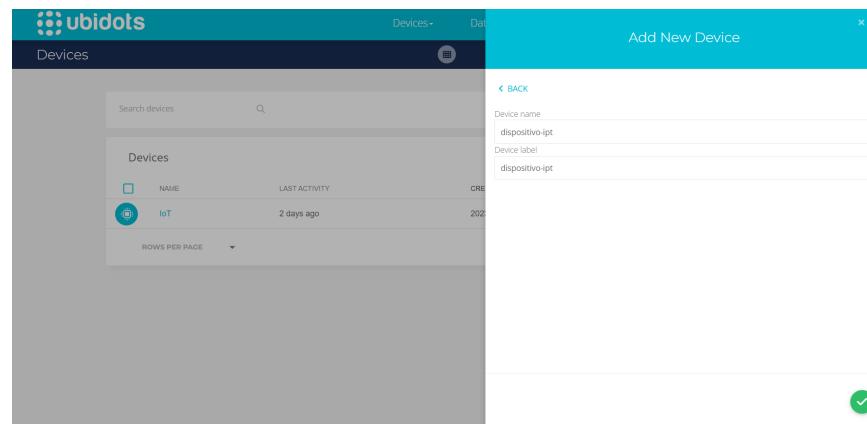
4.1 Criar um dispositivo no Ubidots:

Entre na página de “devices”, como indica a seta presente no meio da captura de tela, após realizar o login no ubidots. Com isso, clique no botão de mais que está sendo mostrado na imagem.

Após isso, aparecerá uma tela de escolha de tipos de device, na qual deve ser selecionado “Blank Device”

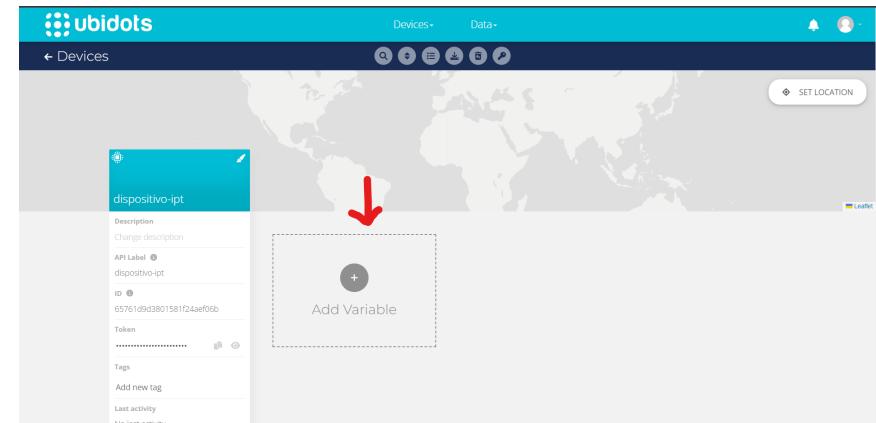


Por fim, deve-se escolher o nome do dispositivo.

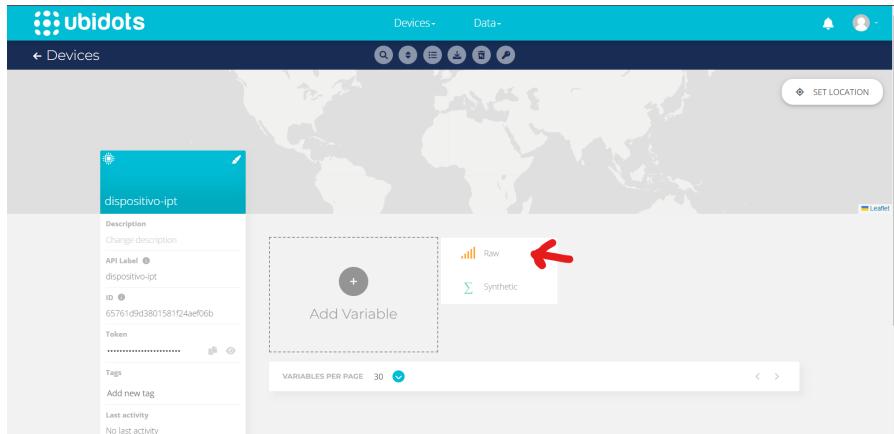


4.2 Adicionando variáveis

Clique no dispositivo adicionado. O usuário será redirecionado para uma página na qual terá escrito “Adicionar variável”, o qual deve ser clicado.



Por fim, criará as variáveis necessárias para a transmissão, do tipo “RAW”, que são peso e temperatura, as quais podem ser dadas o nome desejado pelo usuário.



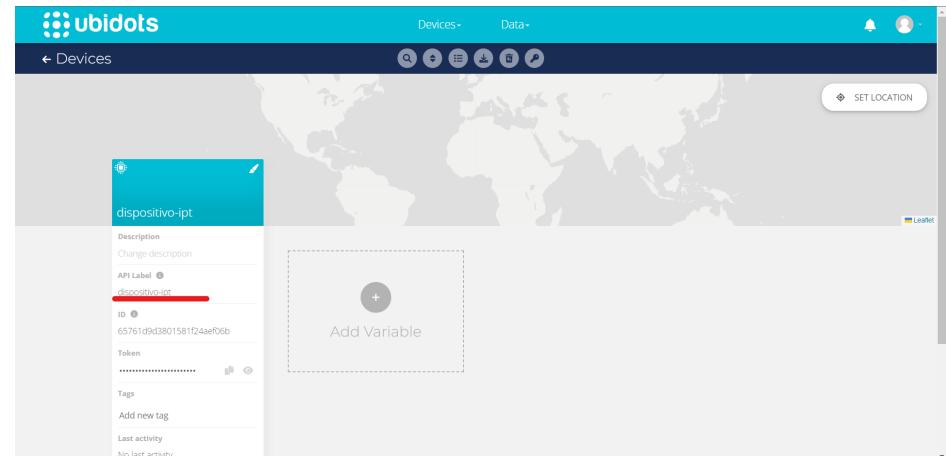
4.3 Trocando as variáveis no código:

Para que sejam enviadas as informações, é necessário alterar as informações, no mesmo arquivo do passo 2. Desta vez será alterada a variável “**.deviceLabel**” para o que está escrito na página de seu dispositivo da aplicação Ubidots.

```

LEDs.cpp ConfigMQTT.cpp InterfaceLCD.cpp ConfigMQTT.h ConfiguracaoOperador.h InterfaceLCD.h LEDs.h Operador.h SDcard.cpp SDcard.h main.ino Operador.cpp
1 #include "Operador.h"
2 #include "ConfiguracaoOperador.h"
3 #include "SDcard.h";
4
5
6 ConfiguracaoOperador config = {
7     .limitePeso      = 4,                                // Constante para o limite de peso
8
9     // Configurações para o MQTT
10    .server          = WiFiServer(80),                  // Servidor WiFi
11    .ssid            = "Inteli-COLLEGE",                // SSID da rede WiFi
12    .password        = "Qazwsx@123",                   // Senha da rede WiFi
13    .token           = "BBUS-EwV8icKtOB8sJOCSzz1UpyRs0s24Lv", // Token do Ubidots
14    .deviceLabel     = "iot",                           // Label do dispositivo no Ubidots
15
16    // Definição dos pinos
17    .DT              = 4,                                // Pino DT
18    .SCK             = 2,                                // Pino SCK
19    .BUTTON_PIN      = 25,                               // Pino onde o botão está conectado
20    .SDCARD_PIN      = 5
  
```

Aqui pode-se verificar no Ubidots o nome do “API Label” e verificar se é o mesmo do **.deviceLabel**



Após essa mudança, no arquivo Operador.cpp, na linha 122, há uma linha de código escrita “configMQTT.publicar(“Massa”, “Temperatura”, dados);”, devem ser alterados as informações “Massa” e “Temperatura” para os nomes das variáveis dadas na plataforma ubidots, conservando as aspas. Seguem imagens de como realizar o passo a passo, desde entrar no arquivo até mudar as informações.

Primeiramente acesse a pasta “src” do projeto

Nome	Data de modificação	Tipo	Tamanho
.git	01/12/2023 15:52	Pasta de arquivos	
assets	30/11/2023 09:49	Pasta de arquivos	
document	30/11/2023 09:49	Pasta de arquivos	
outros	21/11/2023 08:30	Pasta de arquivos	
src	21/11/2023 15:23	Pasta de arquivos	
.gitattributes	Data da criação: 30/10/2023 15:08 Tamanho: 245 KB Pastas: vscode, firmware Arquivos: README.md	Arquivo Fonte Git Att...	1 KB
.gitignore	2023 15:08 2023 08:30 Arquivos: README.md	Arquivo Fonte Git Ig...	1 KB
README.md	30/10/2023 15:08	Arquivo Fonte Markd...	4 KB

Assim acesse a pasta “firmware”,

Nome	Data de modificação	Tipo	Tamanho
.vscode	21/11/2023 15:28	Pasta de arquivos	
firmware	21/11/2023 08:30	Pasta de arquivos	
readme.md	Data da criação: 21/11/2023 08:30 Tamanho: 225 KB Pastas: include, libraries, main	Arquivo Fonte Markd...	1 KB

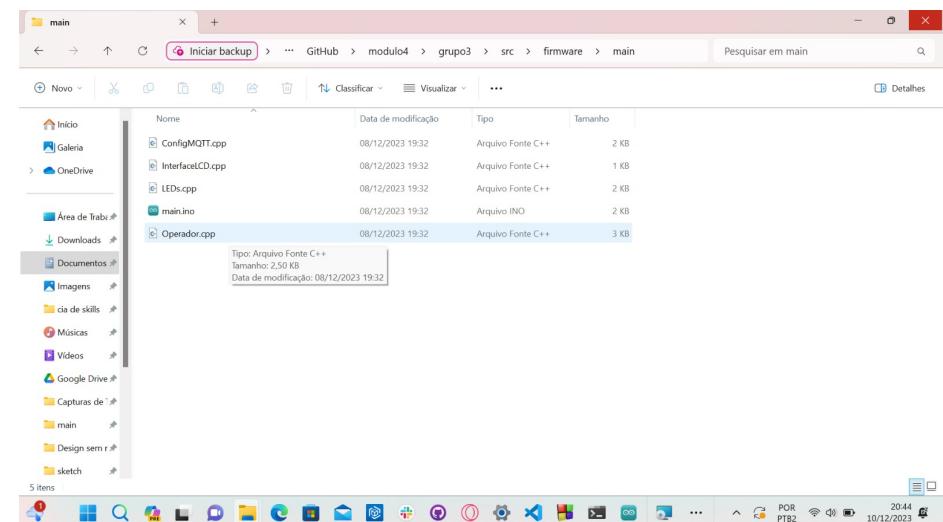
Então acesse a pasta “main”

Nome	Data de modificação	Tipo	Tamanho
include	26/11/2023 21:43	Pasta de arquivos	
libraries	21/11/2023 17:28	Pasta de arquivos	
main	30/11/2023 09:49	Pasta de arquivos	

Data da criação: 21/11/2023 08:30
Tamanho: 8,21 KB



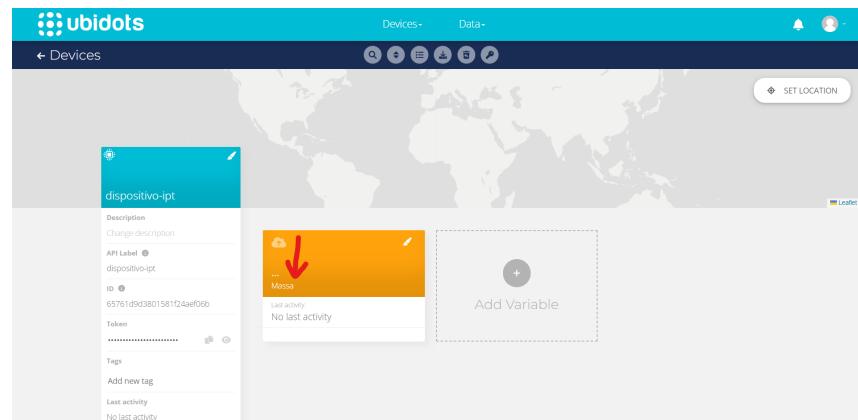
Por fim, acesse o arquivo “Operador.cpp”.



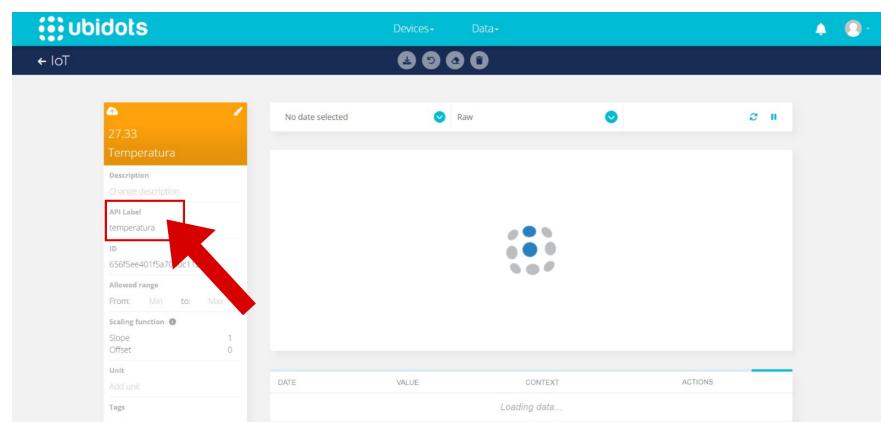
Operador.cpp

Arquivo Fonte C++
Tamanho: 2,50 KB
Data de modificação: 08/12/2023 19:32

Na plataforma Ubidots, após selecionar o dispositivo, aparecerá as variáveis já existentes. Com isso, deve-se selecionar as respectivas variáveis para “massa” e “temperatura” e copiar o api-label, assim como mostra na imagem.



Assim, é possível acessar o “API Label”.

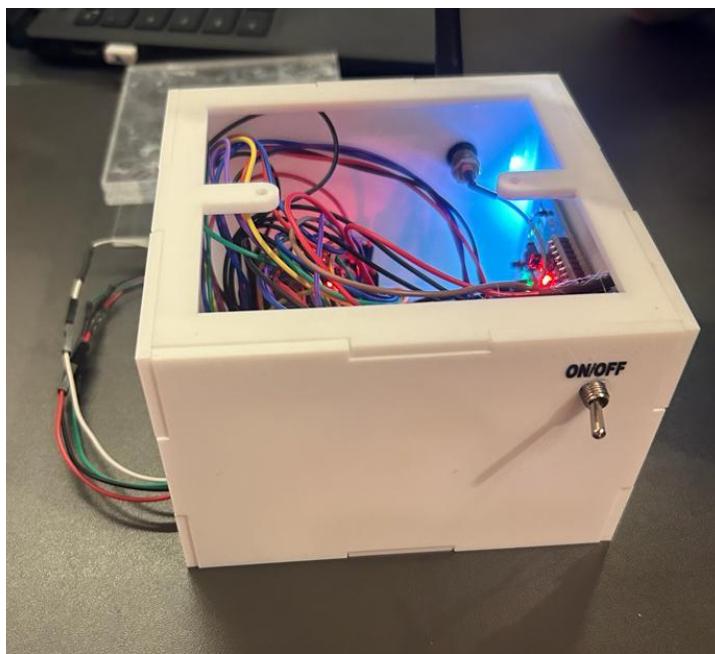


Por fim, deve ser alterado esses dois textos que estão no código para o **api-label** copiado no passo anterior, é importante que se mantenha a ordem de massa e temperatura. OBS: Copie o **api-label** dentro das aspas.

```
LEDs.cpp ConfigMQTT.cpp InterfaceLCD.cpp ConfigMQTT.h ConfiguracaoOperador.h InterfaceLCD.h LEDs.h Operador.h SDcard.cpp SDcard.h main.ino Operador.cpp ...
...
113     ...
114     char temperature[10];
115     sprintf(temperature, "%,.2f", dados.second);
116     sprintf(mass, "%,.2f", dados.first);
117     cartaosd.appendfile(SD, "/fernando.txt", mass);
118     cartaosd.appendfile(SD, "/fernando.txt", " kg");
119     cartaosd.appendfile(SD, "/fernando.txt", " temperature");
120     cartaosd.appendfile(SD, "/fernando.txt", " °C");
121
122     // Manda para o Ubidots e publica as informações sobre os labels: Massa e Temperatura
123     configMQTT.publicar("Massa", "Temperatura", dados);
124     // Caso o peso estiver acima do limite definido pelos engenheiros
125     if (dados.first > limitePeso) {
126         // Led vermelho é ligado, junto ao buzzer com uma intensidade 50.
127         leds.ligarRedLED();
128         analogWrite(SPEAKER_PIN, 50);
129     } else {
130         analogWrite(SPEAKER_PIN, 0);
131         leds.desligarLED();
132     }
133 }
```

PASSO 5: Inicializar a chave HH

Deve virar a chave HH para ON para inicializar o protótipo.



Após o wifi ser conectado, como mostra na imagem abaixo:



Deve-se apertar o botão que está escrito “Leitura”.



PASSO 6: Clicar no botão de iniciar leitura

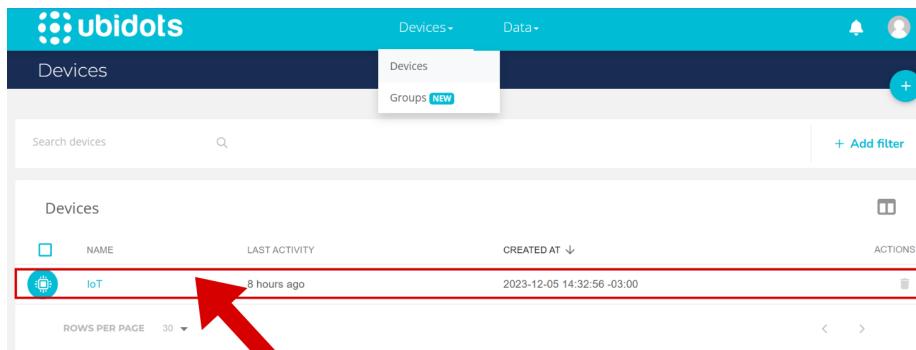
PASSO 7: Seleção da obra e o sensor dentro da plataforma

Uma vez que o dispositivo é iniciado, o usuário deve selecionar dentro da plataforma o dispositivo da obra que ele gostaria de visualizar.

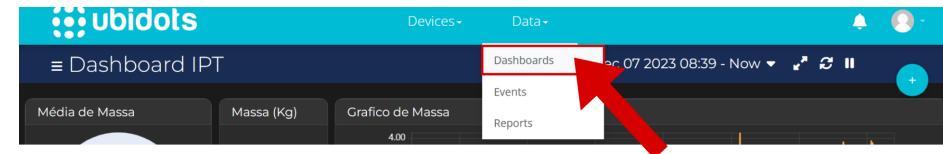
Para selecionar o dispositivo: Dentro do ubidots clique em devices → escolha o device.



Então selecione o dispositivo desejado,



Para visualizar o dashboard: Dentro do ubidots clique em devices → escolha o device.



Assim, é possível visualizar todos os dashboards deste “device”



5. Guia de Operação

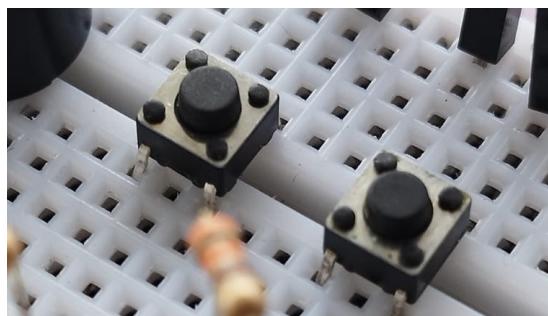
Operação Dispositivo Físico

1. Chave HH (Bateria)



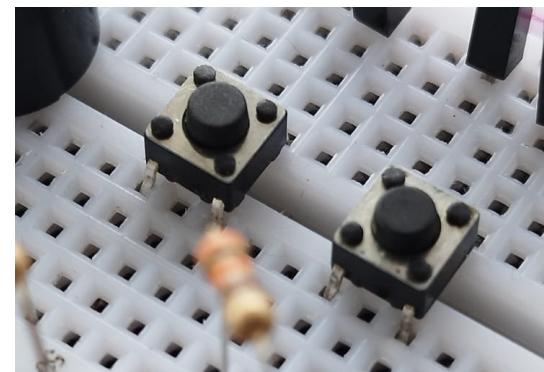
Para começar, ative a Chave HH que liga/desliga a bateria. Assim, o protótipo terá energia para funcionar.

2. Botão Inicializar Célula de Carga



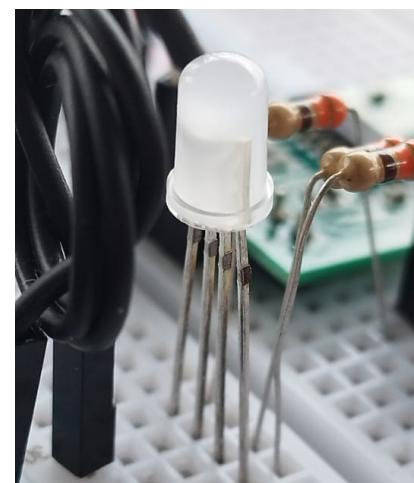
Clique no primeiro botão, da esquerda para a direita, o mesmo inicializa a célula de carga.

3. Botão da Tara



Clique no segundo botão, para zerar a medição da Tara.

4. LED RGB



A partir daqui, o **LED RGB** conversa com você sobre o que está acontecendo no sistema, fornecendo **feedback imediato** relativo ao funcionamento do mesmo e eventuais problemas.

Funcionamento perfeito do sistema: **LED Verde**

Sem internet: **LED Amarelo**

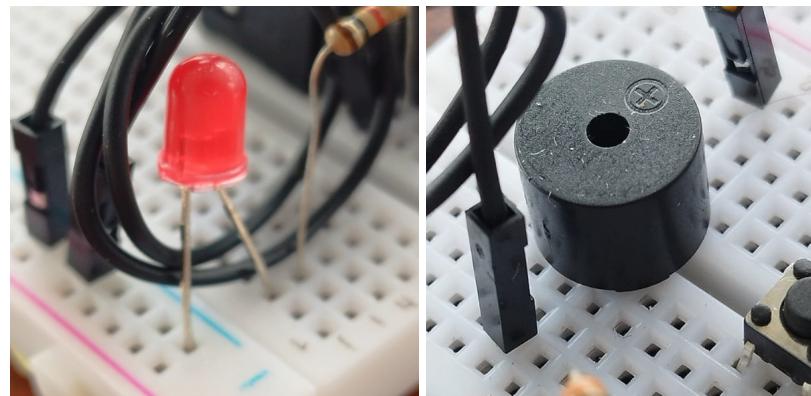
Célula desconectada: **LED Vermelho**

Sensor não encontrado: **LED Roxo**

LCD não encontrado: **LED Branco**

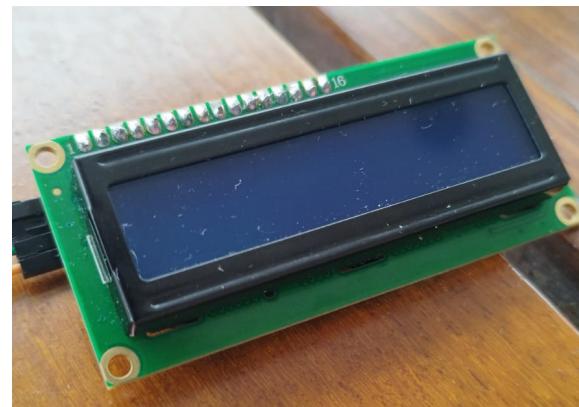
Obs: Na **seção 6, Troubleshooting**, estão listadas possíveis soluções para **todos** os problemas acima.

5. LED Vermelho e Buzzer



Atenção! O LED Vermelho e o Buzzer indicam carga acima do limite de 4kg, portanto, se o LED acender e o Buzzer apitar, fique atento à célula de carga!

6. Display LCD



O display mostrará as informações de peso e temperatura, o botão que inicializa a célula de carga, descrito no ponto 2, também liga o display.

7. Célula de Carga



Agora a célula de carga estará ligada, para garantir e testar seu funcionamento, coloque sobre ela um peso conhecido e veja se o processo de medição ocorre sem erros, ou seja, se ela retorna o valor correto do peso conhecido.

Operação Interface Ubidots

Para realizar o login no sistema e testar a célula de carga, verifique a seção 4, lá, está descrito passo a passo como você pode entrar na plataforma.

6. Troubleshooting

#	Problema	Possível solução
1	Falha de conexão WiFi	Implementar um mecanismo de reconexão automática. O ESP32 pode ser programado para tentar se reconectar à rede Wi-Fi em intervalos regulares em caso de falha. Além disso, o indicador visual (LED RGB) sinaliza quando o dispositivo está ou não conectado à rede.
2	Falha Protocolo MQTT	Incorporar um sistema de verificação de integridade do protocolo MQTT. O ESP32 pode ser programado para monitorar ativamente a comunicação MQTT e reconectar-se em caso de falhas. Também pode ser útil implementar logs de eventos para ajudar na identificação e resolução de problemas relacionados ao protocolo.
3	Sensor não encontrado	Implementar um mecanismo de verificação dos sensores

		durante a inicialização. O código do ESP32 pode verificar se os sensores, como o BME280 e o HX711, estão disponíveis e funcionando corretamente. Em caso de falha na detecção, o sistema pode registrar o evento e, se possível, tentar reinicializar os sensores.
4	LCD não encontrado	Similar à verificação de sensores, durante a inicialização, o código pode verificar a presença do LCD. Se o LCD não for detectado, o sistema pode registrar essa falha e, dependendo da criticidade, pode continuar operando sem a interface do LCD ou entrar em um modo de segurança.
5	Célula de Carga Desconectada	Implementar um sistema de monitoramento contínuo para a célula de carga. O ESP32 pode verificar regularmente a conexão da célula de carga e registrar eventos

se a desconexão for detectada. Além disso, pode-se incluir um mecanismo de alerta, como um LED piscante, para indicar a desconexão da célula de carga.