Protótipo de Nariz Artificial

Motivação, planejamento, montagem e programação do protótipo

GRENDES LabGeo

@GRENDESLABGEO

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PERNAMBUCO

@IFPERECIFE

Eco Sniff

@ECOSNIFF

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Protótipo de nariz artificial [livro eletrônico] : motivação, planejamento, montagem e programação do protótipo / coordenação Aida Araújo Ferreira...[et al.]. -- 1. ed. -- Recife, PE : Ed. dos Autores, 2024.
PDF

Outros coordenadores: Gilmar G. de Brito, Ioná Maria B. R. Barbosa, Romero Barbosa de Assis, Ronaldo R. B. de Aquino, Vânia S. de Carvalho.
Vários colaboradores.
Bibliografia.
ISBN 978-65-01-06701-8

1. Compostagem 2. Efeito estufa (Atmosfera) 3. Engenharia 4. Inovação tecnológica 5. Meio ambiente 6. Prototipagem 7. Resíduos sólidos 8. Tecnologia I. Ferreira, Aida Araújo. II. Brito, Gilmar G. de. III. Barbosa, Ioná Maria B. R. IV. Assis, Romero Barbosa de. V. Aquino, Ronaldo R. B. de. VI. Carvalho, Vânia S. de.

24-213256 CDD-620.0042

Índices para catálogo sistemático:

1. Protótipos : Engenharia : Tecnologia 620.0042

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Coordenação

Prof. Dra. Aida Araújo Ferreira
Prof. Dr. Gilmar G. de Brito
Prof. Dra. Ioná Maria B. R. Barbosa
Prof. Dr. Romero Barbosa de Assis
Prof. Dr. Ronaldo R. B. de Aquino
Prof. Dra. Vânia S. de Carvalho

Equipe

Carla Maria da Silva
Isaque Domingos Santana Silva
Júlia Didra Bezerra de Assis
Niviane Cristina Alves da Silva
Rebeca Breatriz de Luna

Sumário

SOBRE O PROJETO

A empresa Lógica Ambiental, responsável pelo tratamento de efluentes e resíduos sólidos da Região Metropolitana do Recife, estava sendo alvo de reclamações da população circundante por conta do forte odor que exalava da estação, devido aos processos de compostagem e tratamento de efluentes domésticos. Em contrapartida, como alternativa para solucionar o problema, a empresa instalou biofiltros feitos de palha, responsáveis por purificar o ar expelido pelas leiras de compostagem.

A partir disso, a EcoSniff surgiu como auxiliadora no processo de validação da eficácia do biofiltro de palha. Para isso, foi desenvolvido um protótipo de nariz artificial para captação de gases oriundos da compostagem, antes e após a aplicação do filtro biológico. Foram escolhidos como material de trabalho, para elaboração do protótipo, equipamentos de baixo custo de aquisição e fácil manuseio.

Imagens aéreas da Lógica Ambiental

Fonte: https://logicaambiental.negocio.site/.

Leiras de Compostagem

Fonte: Próprio Autor

Coleta de resíduos líquidos das leiras

Fonte: Próprio Autor

INTRODUÇÃO

A **compostagem** e os malefícios da **exposição aos gases** expelidos

Devido ao crescimento populacional e geração excessiva de lixo, o descarte e armazenamento ambientalmente seguro de compostos orgânicos têm se caracterizado como um assunto importante a ser discutido e solucionado na sociedade hodierna. Os resíduos sólidos são categorizados como:

a) Resíduos sólidos urbanos, que incluem resíduos de cozinha, ou seja, orgânicos e facilmente degradáveis na natureza;

b) Resíduos recicláveis como papel, que podem ser reutilizados;

c) Resíduos hospitalares e resíduos tóxicos produzidos durante o processo de atividades industriais;

Para a sociedade moderna, é um desafio o equacionamento da geração excessiva e da disposição final confiável e segura dos resíduos sólidos, pois evidencia um problema ambiental que afeta a humanidade, indo na contramão do contexto da sustentabilidade.

COMPOSTAGEM

Como alternativa, o estudo e efetivação do processo de compostagem dos resíduos sólidos urbanos surge com urgência. Com isso, a Resolução nº 481 de 03 de outubro de 2017, do Conselho Nacional do Meio Ambiente – CONAMA, define como compostagem o processo de decomposição biológica controlada dos resíduos orgânicos, efetuado por uma população diversificada de organismos, em condições aeróbias e termofílicas, resultando em material estabilizado, com propriedades e características completamente diferentes daqueles que lhe deram origem; sendo denominado composto o produto estabilizado, oriundo do processo de compostagem, podendo ser caracterizado como fertilizante orgânico, condicionador de solo e outros produtos de uso agrícola.

CICLO DE VIDA DOS COMPOSTOS

Coleta de resíduos orgânicos urbanos

Produção de fertilizantes naturais

Agricultura ou recuperação de solos pobres

Colheita e produção de novos resíduos orgânicos

GASES ORIUNDOS DA COMPOSTAGEM

Sabe-se que alguns gases podem causar malefícios para a saúde do organismo humano, tendo em vista que os odores desagradáveis gerados pelas indústrias são uma mistura complexa de gases presentes em concentrações, poeiras e vapores mais elevados. As emissões de gases causam reações ofensivas nos indivíduos expostos e podem desencadear várias implicações para a saúde como problemas respiratórios, irritação nasal, além de deteriorar o ambiente circundante. Esses gases geralmente contêm poluentes orgânicos e inorgânicos.

Com isso, os odores são hoje, entre os poluentes atmosféricos, a maior causa das reclamações às autoridades locais. Nesse sentido, de acordo com a Resolução nº 481 de 03 de outubro de 2017, do Conselho Nacional do Meio Ambiente – CONAMA, as usinas de compostagem devem adotar medidas de controle ambiental para diminuição da emissão de odores.

Equipamentos utilizados:

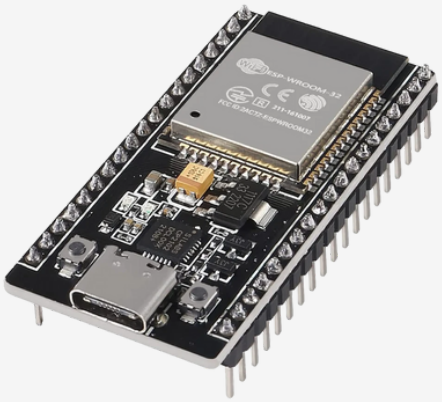
ESP32, Sensores MQ , Sensor DHT22, Placa Protoboard e Jumpers

Para a criação do protótipo de nariz artificial, foi utilizado um módulo ESP32, que é a junção entre o chip microcontrolador ESP32 e uma placa PCB (Printed Circuit Board ou Placas de Circuito Impresso) que possui Wi-fi embutido, sensor eletroquímico MQ135, o sensor de temperatura e umidade DHT22, uma placa protoboard e jumpers.

ESP32

O módulo ESP32 representa um avanço significativo no universo da Internet das Coisas (IoT), oferecendo uma plataforma versátil e útil para o desenvolvimento de dispositivos conectados. Este se destaca dentre as outras opções por sua combinação de potência de processamento, conectividade sem fio e uma abrangência de recursos. Devido suas pequenas dimensões, é ideal para interligação com outros dispositivos e integração em placas de PCB (Printed Circuit Board). Além disso, este se destaca dentre as outras opções por sua combinação de potência de processamento, conectividade sem fio e uma abrangência de recursos. A conectividade sem fio é um diferencial do ESP32. Essa capacidade de comunicação sem fio possibilita a integração de dispositivos IoT em redes locais e sua interação com outros dispositivos inteligentes.

Módulo ESP32



Fonte: https://www.ardurobotica.com.br

SENSORES MQ

A família MQ é composta por diversos sensores que vão do MQ-2 ao MQ-9, MQ-131 e MQ-135 ao MQ-138, possuem diversas especificações e são capazes de detectar diversos gases, dentre eles: ozônio, butano, metano, álcool, gás natural e entre outros. Os sensores são compostos por um micro tubo cerâmico com Óxido de Alumínio, uma camada sensitiva de Óxido de Estanho e um eletrodo para medida fixado junto a um aquecedor e a uma carcaça de plástico com um trançado feito de aço inoxidável.

CURIOSIDADE
AQUECIMENTO DOS SENSORES

O aquecedor do sensor desempenha um papel importante para uma calibração mais precisa dos sensores. De acordo com (KUMAR, 2021), o aquecedor possibilita que o sensor esteja em condições ideais para realizar a leitura de forma precisa, e para que a temperatura seja atingida de acordo com as recomendações do fabricante, é necessário que o aquecedor do sensor fique ligado pelo menos por vinte e quatro horas antes da realização da medida.

SENSOR MQ-135

São utilizados em equipamentos de controle da qualidade do ar para edifícios/escritórios e captam os seguintes gases: Amônia (NH3), Álcool, Benzeno (C6H6), Fumaça e Dióxido de Carbono (CO2). Os sensores MQ-135 possuem amplo escopo de detecção, vida estável e longa, resposta rápida, alta sensibilidade e circuito de acionamento simples.

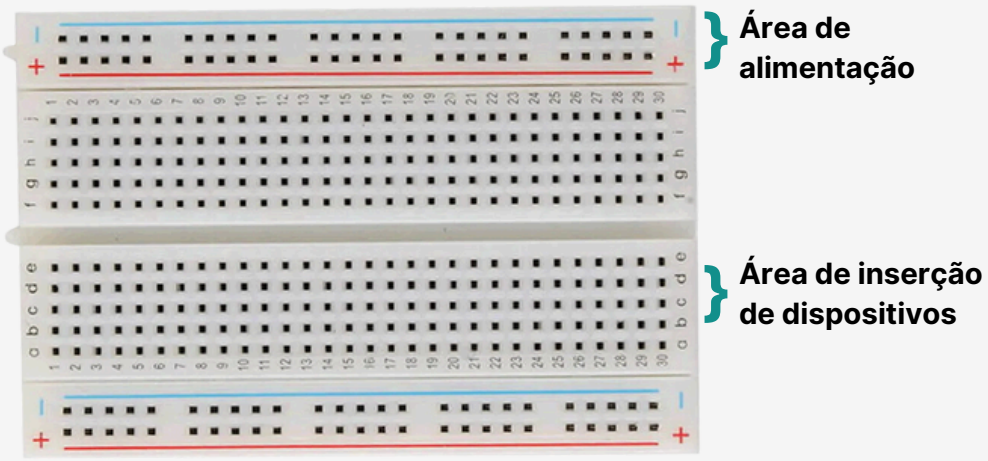
Sensor MQ-135

Fonte: https://datacombr.com.br

PLACA PROTOBOARD

É uma matriz de contato constituída por uma base plástica e conexões condutoras com furos para encaixe de componentes eletrônicos, utilizada para montagem de circuitos e testes.

Placa Protoboard



Fonte: adaptado de https://datacombr.com.br

De acordo com a imagem acima, na **área de alimentação**, todos os furos de uma mesma coluna estão internamente conectados. No entanto, os furos de uma coluna não estão internamente conectados com os furos de outra coluna.
Seguindo o mesmo raciocínio, na **área de inserção de dispositivos**, os furos de uma mesma linha estão internamente conectados, mas elas são independentes entre si.

CURIOSIDADE

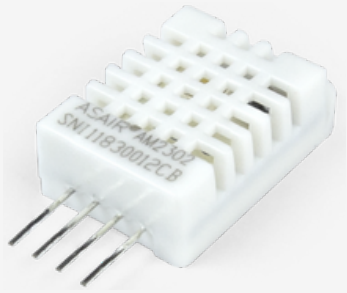
CONEXÕES DE ALIMENTAÇÃO

Na área de alimentação, temos a coluna sinalizada com um sinal de **positivo** e outra com um sinal de **negativo** (como mostrado na figura acima), ambas podem ser destinadas à conexão terra (GND) ou de energia (VCC). No entanto, é preferível seguir a dinâmica da placa, **positivo para energia e negativo para terra**.

SENSOR DHT22

É um sensor responsável por quantificar temperatura (em graus Celsius, que variam de -40° a 80°) e umidade (gramas por metros cúbicos, que variam de 0% a 100%).

Sensor DHT22

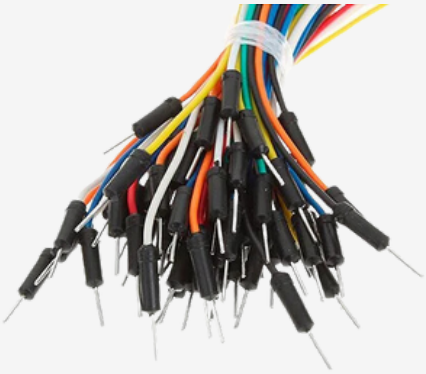


Fonte: https://grafenocomponentes.com.br

JUMPERS

Os jumpers são chaves elétricas utilizadas em placas e alguns dispositivos, utilizada para ativar, regular ou desativar funções específicas do sistema que não são acessíveis via software.

Jumpers



Fonte: https://cdn.awsli.com.br

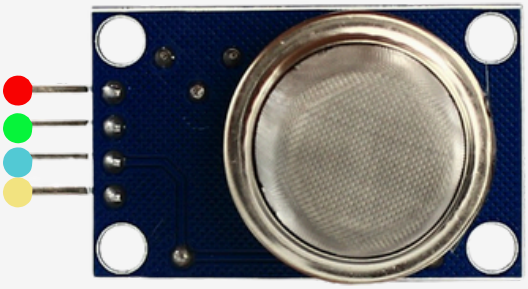
MÃOS À OBRA!

Agora que já conhecemos todos os equipamentos necessários para montagem do protótipo, vamos colocar a mão na massa!

MONTAGEM DO PROTÓTIPO

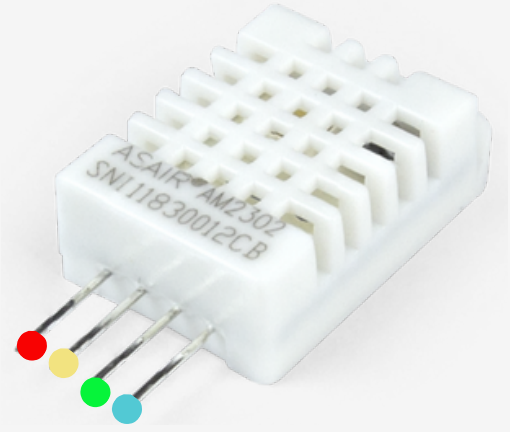
Para compreendermos melhor o processo de montagem do protótipo proposto, antes, necessitamos analisar e entender a pinagem dos sensores utilizados. A pinagem, diz respeito à funcionalidade de cada pino presente no sensor, bem como seus respectivos receptores na placa utilizada.

PINAGEM DO SENSOR MQ-135



- **VCC (energia):** fornece energia para o módulo.
  Deve ser conectado no pino 5V da placa de desenvolvimento ESP32;
- **GND (terra):** é o pino de aterramento e precisa ser conectado ao pino GND da placa;
- **DO:** fornece uma representação digital da presença de gases tóxicos;
- **AO:** fornece tensão de saída analógica proporcional à concentração de gases tóxicos.

PINAGEM DO SENSOR DHT22



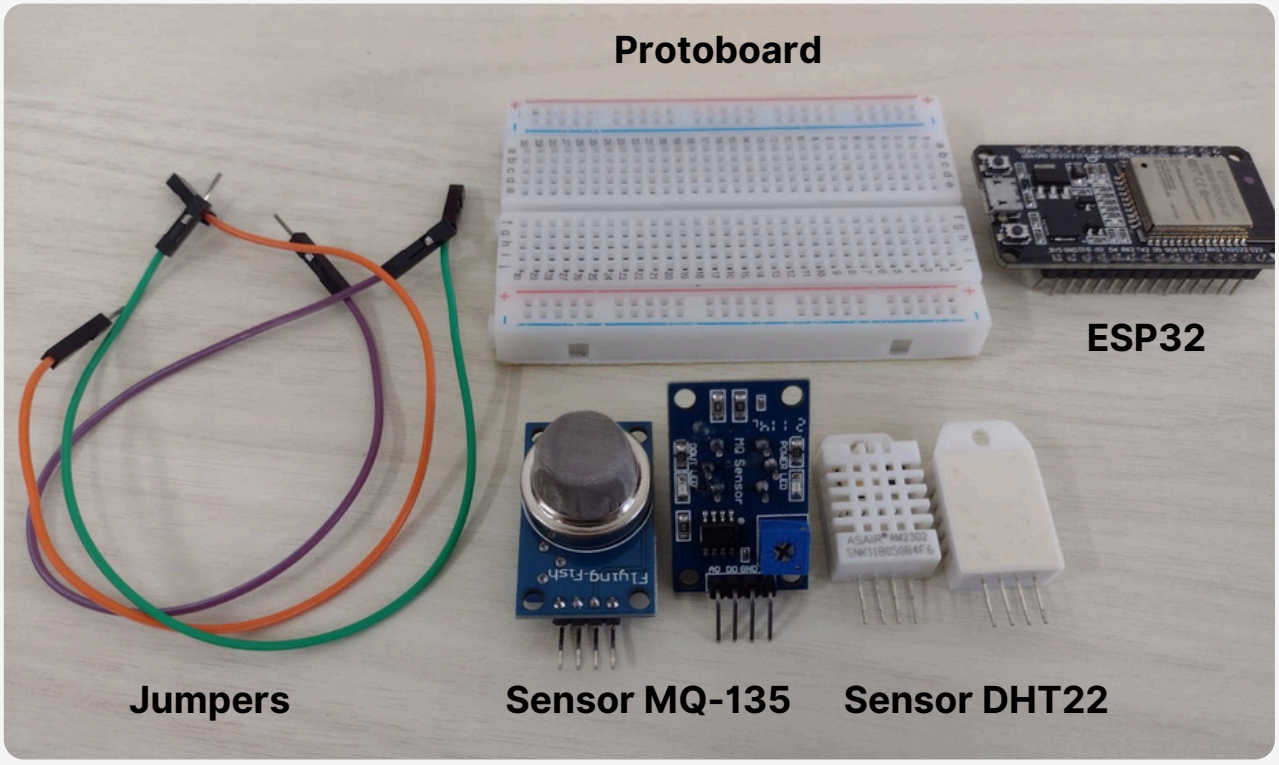
- **Alimentação:** 3,3 a 5,5 VCC
- **Saída Data**
- **Não é utilizado**
- **GND:** 0V

Montagem do **Protótipo**

Para essa etapa de montagem do protótipo, faremos um passo a passo para melhor compreensão do processo.

PASSO 1

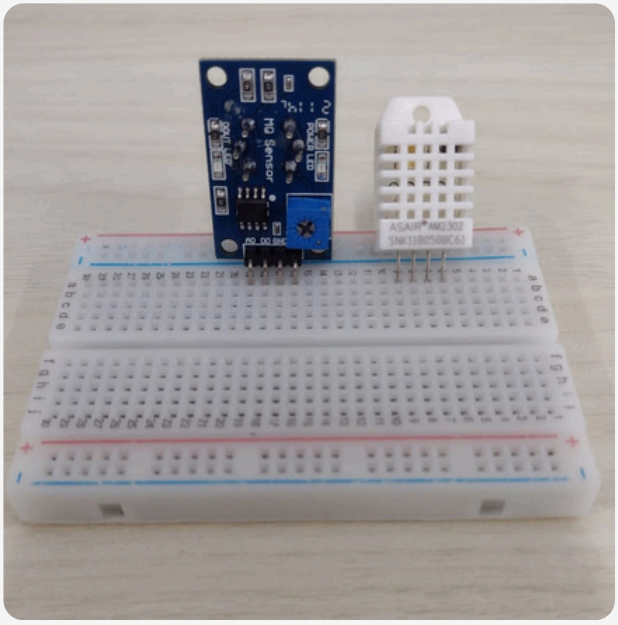
Separe todos os equipamentos que utilizaremos na montagem:



PASSO 2

Conecte os sensores (MQ-135 e DHT22) na Protoboard. Fique à vontade para escolher os furos que achar melhor, mas, para seguir um padrão, aconselhamos que coloquem o sensor DHT22 nos furos a20, a21, a22 e a23 e os sensores MQ-135 nos furos a12, a13, a14, a15 e a3, a4, a5,e a6 , respectivamente.

Observe a imagem ao lado.



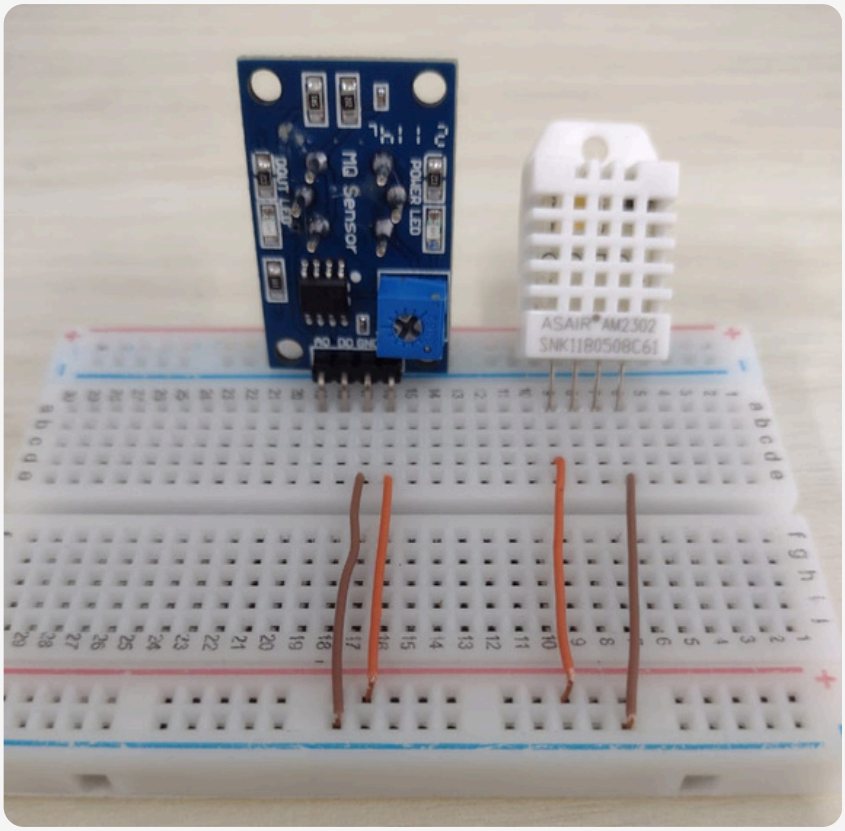
PASSO 3

Neste passo é feita a ligação entre a protoboard e os sensores através dos jumpers, nas seguintes posições:

Sensor DHT22
Pino 1 - VCC, ligado no VCC na protoboard.
Pino 2 - DATA ligado no pino D4 da ESP32.
Pino 3 - Não utilizar.
Pino 4 - GND, ligado na protoboard no GND.

MQ4 e MQ135:
Pino 1 - A0 ligado no D14 na ESP32.
Pino 2 - D0 não utilizar.
Pino 3 - GND ligado no GND na ESP32.
Pino 4 - VCC ligado no VCC na ESP32.

Observe a imagem abaixo.

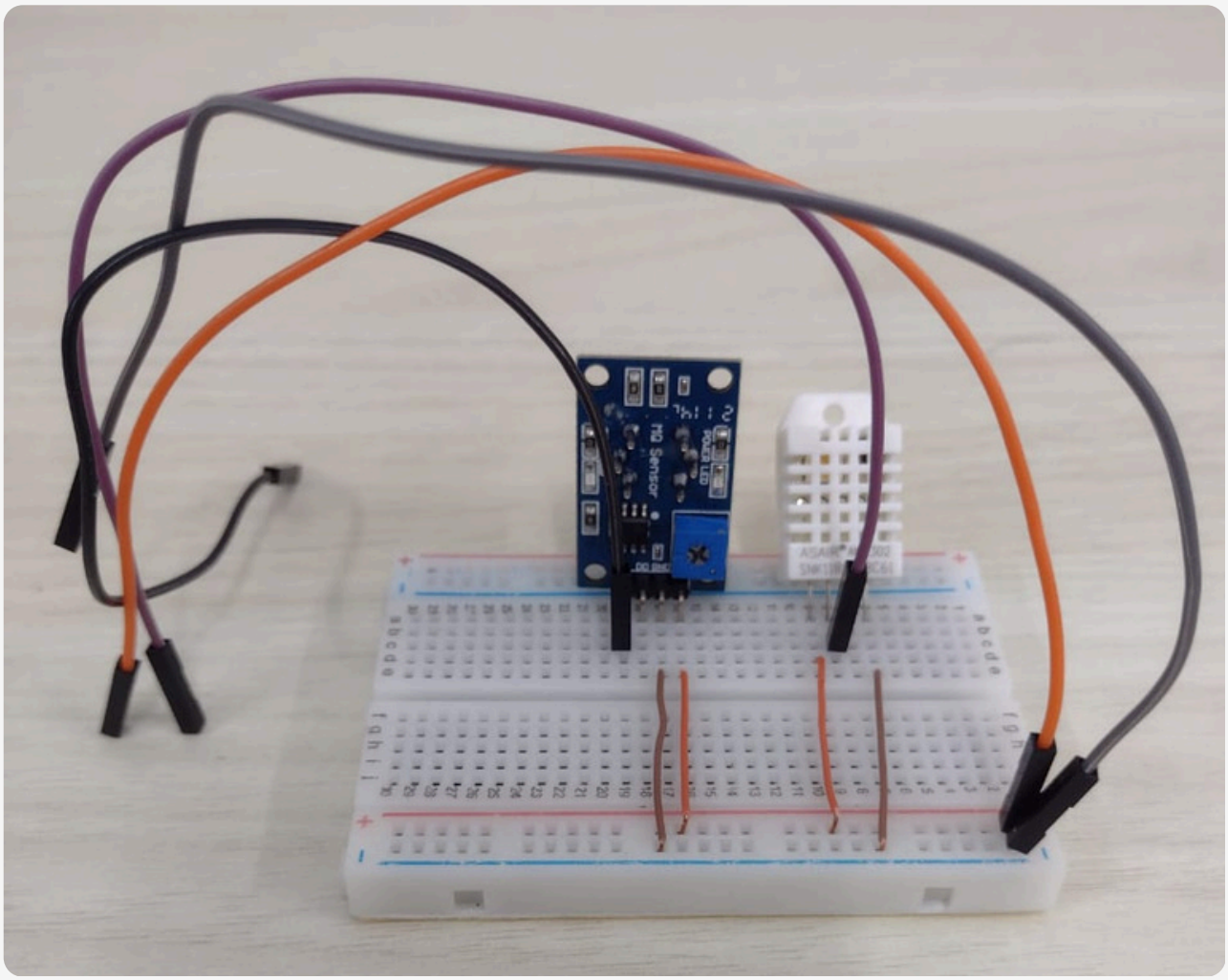


PASSO 4

Nesse passo são colocados os jumpers ligados na protoboard, nas posições:

- Jumper laranja - Ligado no VCC.
- Jumper cinza: Ligado no GND.
- Jumper roxo - Ligado no pino DATA do sensor DHT22.
- Jumper preto - Ligado no A0 do sensor MQ135.

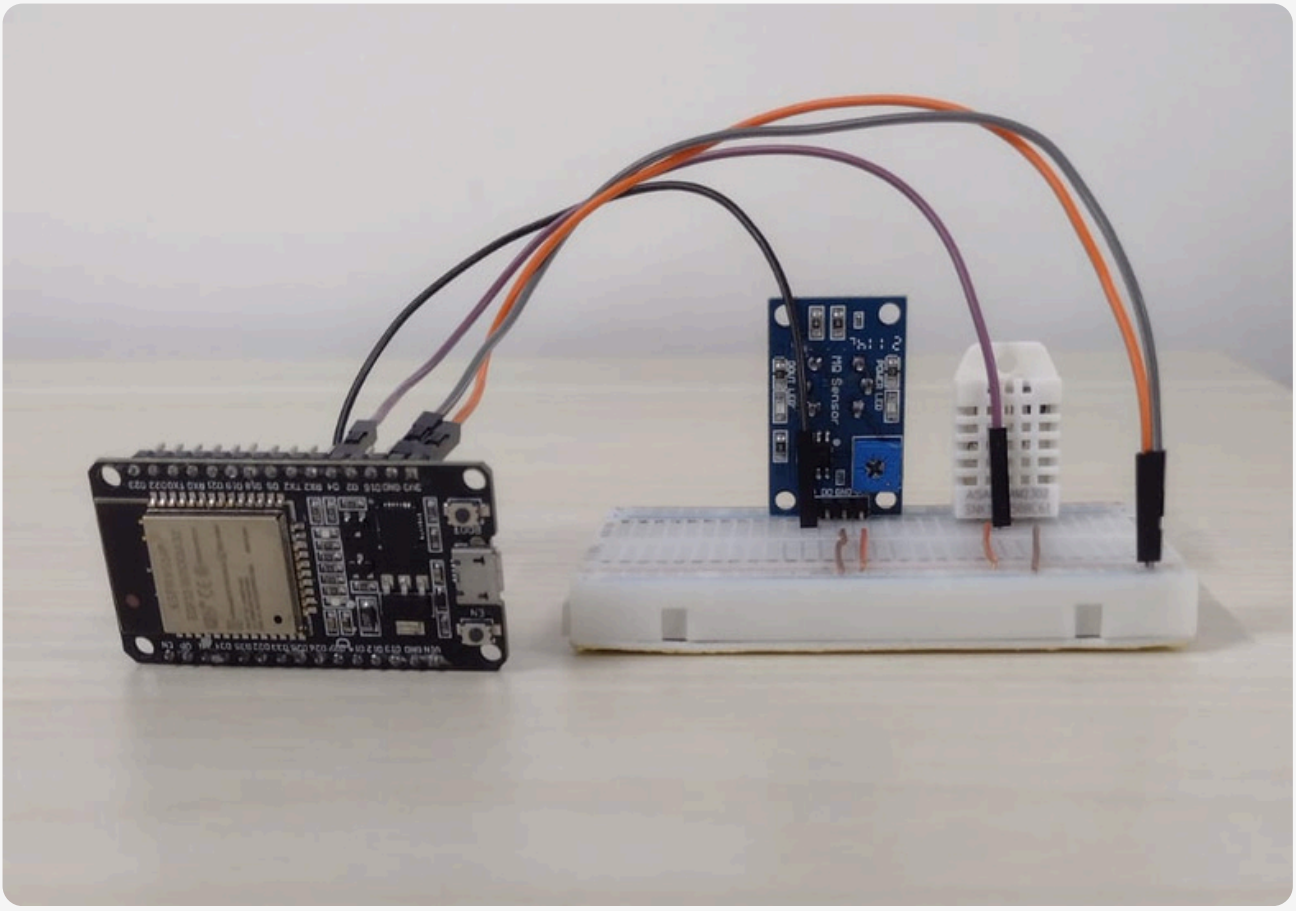
Observe a imagem a baixo



PASSO 5

Por fim, esta é a imagem do protótipo montado, utilizando todos os equipamentos descritos anteriormente.

Observe a imagem baixo.



Tecnologias utilizadas:

Arduino IDE e Blynk

Neste capítulo, apresentaremos as **principais tecnologias** utilizadas no projeto do nariz artificial para detecção de gases provenientes da compostagem. Essas tecnologias são fundamentais para o funcionamento e a integração dos componentes do sistema, permitindo o monitoramento em tempo real dos gases detectados.

ARDUINO IDE

A Arduino IDE, ou Integrated Development Environment (Ambiente de Desenvolvimento Integrado) Arduino, é um software utilizado para programar placas Arduino. Ela fornece uma interface gráfica simplificada e amigável, facilitando a criação de código para controle de dispositivos eletrônicos. A IDE inclui um editor de texto onde você escreve seu código, um compilador que transforma o código em linguagem de máquina compreensível pelo Arduino, e um carregador que envia o código compilado para a placa Arduino.
Além disso, a Arduino IDE possui uma vasta biblioteca de funções pré-definidas (chamadas de "sketches") que facilitam a interação com os componentes eletrônicos, como sensores, motores e LEDs. Essa facilidade de uso e a grande comunidade de desenvolvedores ao redor do mundo tornaram a Arduino IDE uma ferramenta popular para projetos de eletrônica e automação.
Ainda dentro do tópico Arduino IDE, as bibliotecas desempenham um papel crucial para o funcionamento adequado do projeto do nariz artificial. Abaixo, vou comentar sobre as principais bibliotecas utilizadas para cada componente específico:

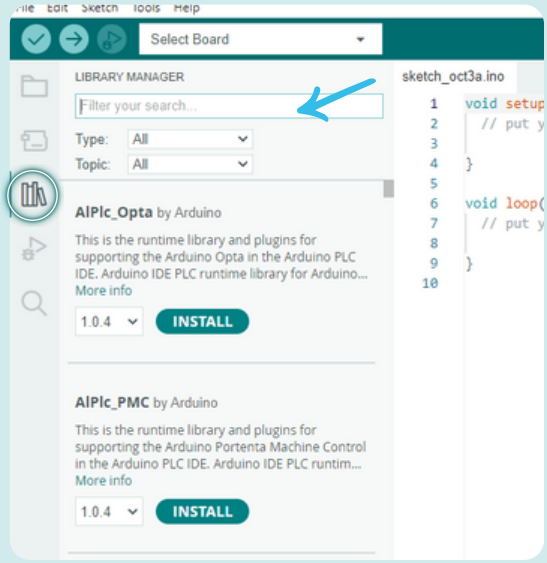
- **Placa ESP32**: A ESP32 é uma placa de desenvolvimento que possui conectividade Wi-Fi e Bluetooth. Para programar a ESP32 na Arduino IDE, é necessário instalar a placa ESP32 na IDE. Isso é feito adicionando uma URL específica nas preferências da IDE e, em seguida, instalando a placa através do Gerenciador de Placas.

- **Sensores da família MQ**: Os sensores da família MQ são utilizados para detectar gases. Para utilizar esses sensores, geralmente é necessário utilizar uma biblioteca específica para cada modelo de sensor MQ que você está utilizando. Uma biblioteca comum é a "MQSensor" que oferece funções para calibrar e ler os valores dos sensores MQ.
- **Sensor DHT22**: O sensor DHT22 é um sensor de umidade e temperatura. Para utilizá-lo, é necessário instalar a biblioteca "DHT sensor library" na Arduino IDE. Essa biblioteca fornece funções para ler os valores de umidade e temperatura do sensor DHT22.
- **Biblioteca do Blynk**: O Blynk é uma plataforma IoT que permite criar aplicativos para controlar e monitorar dispositivos eletrônicos pela Internet. Para integrar o Blynk ao projeto, é necessário instalar a biblioteca do Blynk na Arduino IDE. Essa biblioteca fornece funções para se conectar à plataforma Blynk e enviar e receber dados do aplicativo Blynk.

É importante lembrar que, para utilizar essas bibliotecas, é necessário instalá-las na Arduino IDE. Isso geralmente é feito através do Gerenciador de Bibliotecas da IDE, onde você pode pesquisar e instalar as bibliotecas necessárias para o seu projeto.

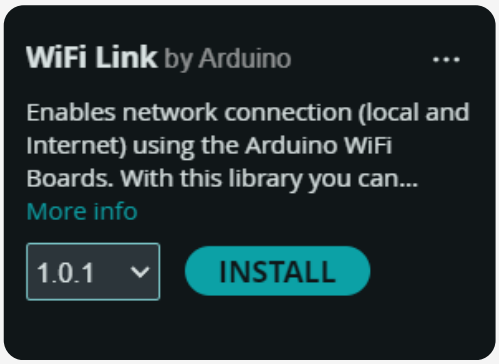
MÃOS À OBRA!

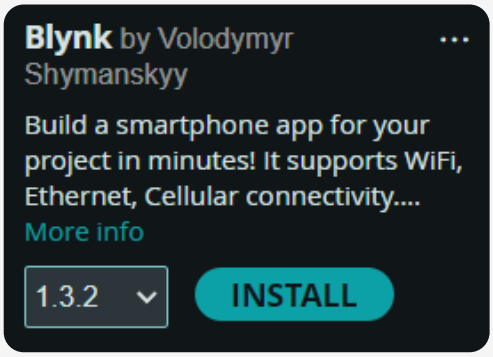
Agora mão na massa! Iremos instalar as bibliotecas necessárias para esse projeto no nosso Arduino IDE. Para isso, clique no icone de **library manager**. Após isso iremos procurar as **bibliotecas** necessárias para rodar o programa.

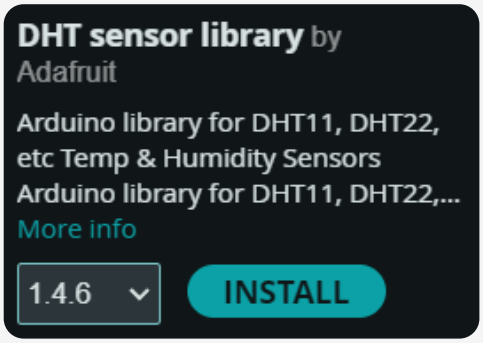


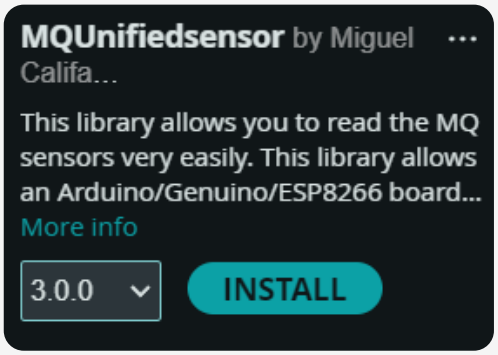
Instalando as bibliotecas da IDE Arduino

Procure pelas bibliotecas abaixo e as instale:







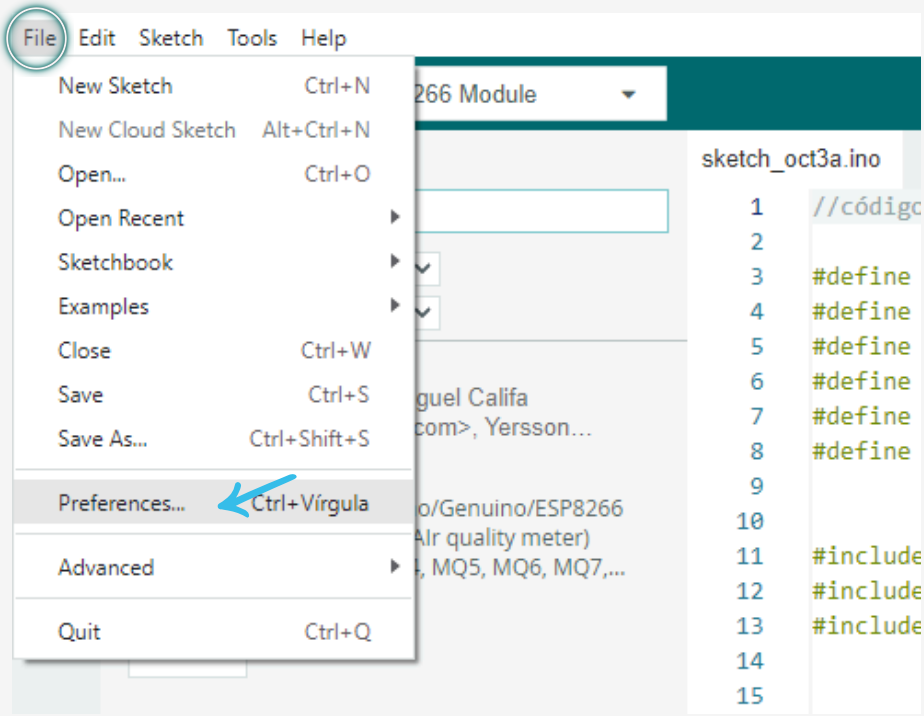


ATENÇÃO

Lembre-se de sempre instalar as versões mais recentes das bibliotecas! E tendo feito isso, as configurações do Arduino IDE estão finalizadas.

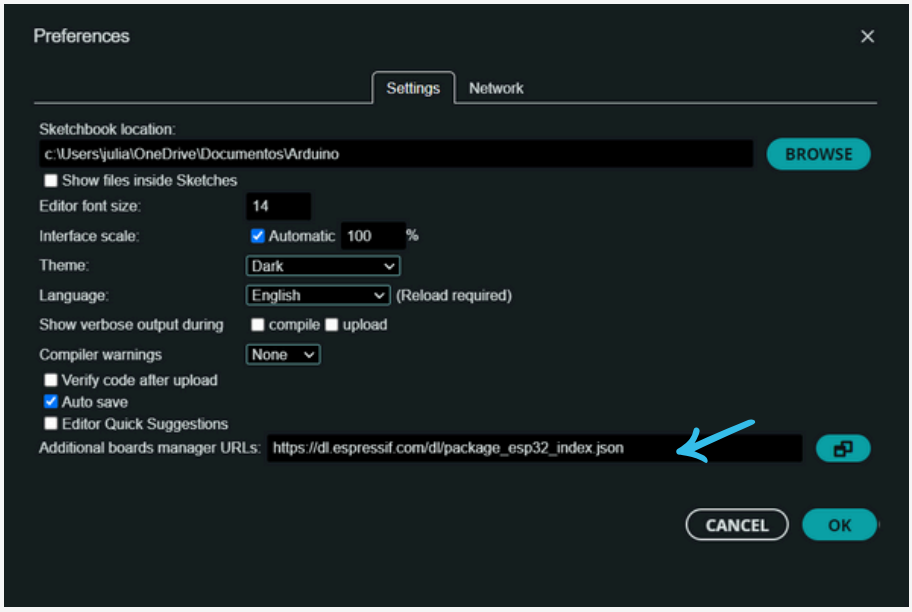
Instalando a placa ESP32 no Arduino IDE:

Para usar o ESP32 pela primeira vez no Arduino IDE é necessário configura-lo para suportar a placa. Abra o Arduino IDE, vá em Arquivo -> Preferências



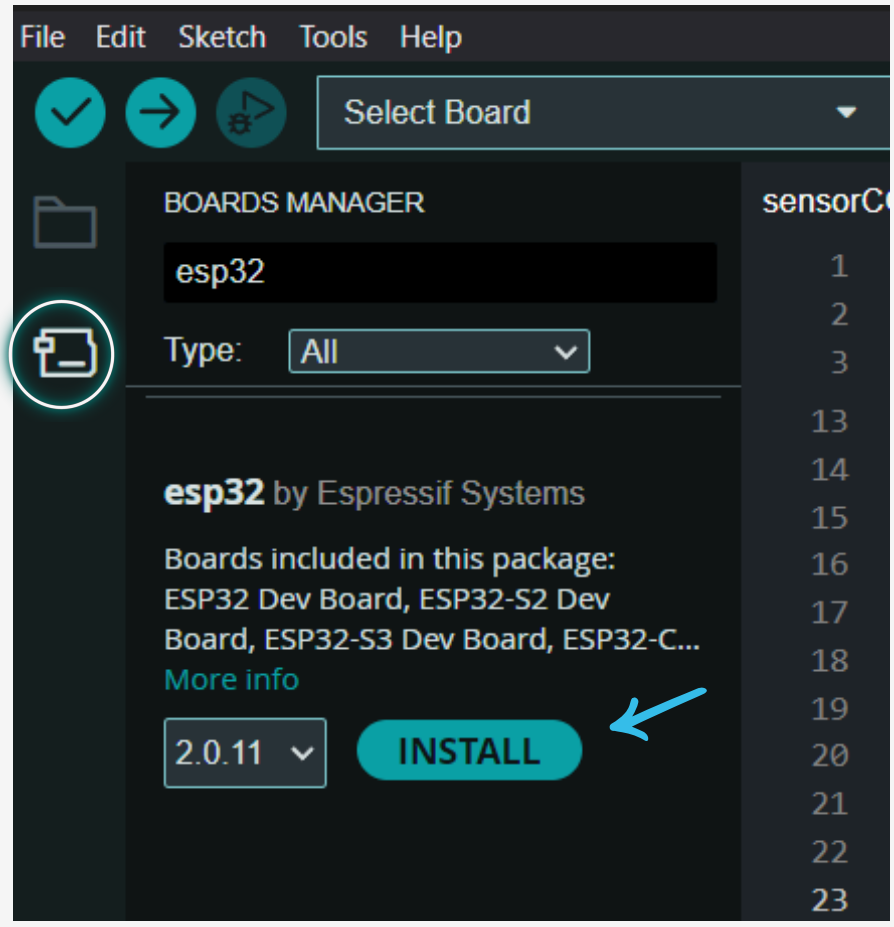
Procure pelo campo URLs adicionais de Gerenciadores de Placas: e insira o link abaixo:

https://dl.espressif.com/dl/package_esp32_index.json

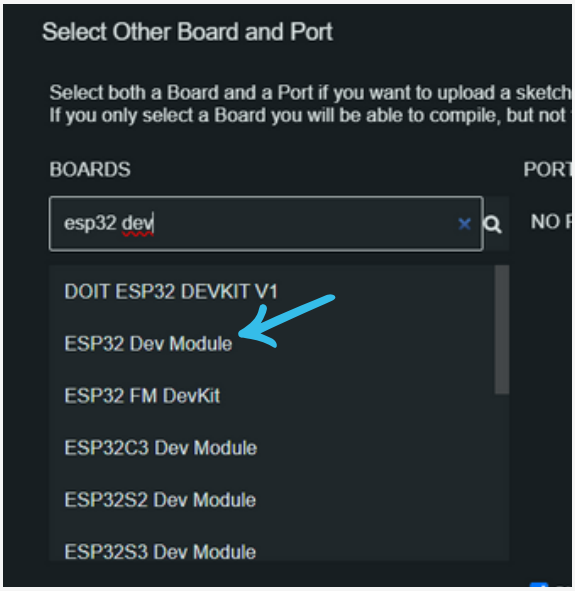


Instalando a placa ESP32 no Arduino IDE:

Em seguida, vá até o ícone de “boards” e procure pela placa ESP32 e instale-a.

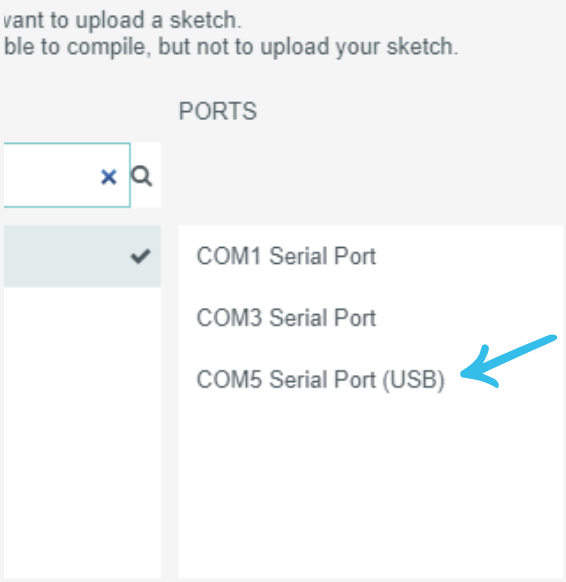


Após esse processo, vá na aba “select board”, busque pela placa ESP32 e selecione a opção “ESP32 DEV MODULE”.



Instalando a placa ESP32 no Arduino IDE:

Em seguida selecione a porta USB que sua placa está selecionada como no exemplo abaixo:



ATENÇÃO

Verifique se sua placa ESP32 ligou alguma luzinha ou led, isso indicará que sua placa está ligada e funcionando!

PLATAFORMA BLYNK

A plataforma Blynk é uma ferramenta poderosa para a criação de aplicativos IoT (Internet das Coisas) de forma rápida e fácil, graças a sua maneira intuitiva de criar interfaces de usuário para os projetos, tornando a interação com dispositivos eletrônicos pela Internet mais acessível e amigável. A ferramenta permite a criação de interfaces gráficas personalizadas, chamadas de "Widgets”, para controlar e monitorar dispositivos conectados à Internet.
No projeto do nariz artificial, a plataforma Blynk foi utilizada para criar uma interface de usuário que exibe em tempo real os dados de detecção dos gases, temperatura e umidade facilitando o acompanhamento do processo de compostagem. Sendo assim, é possível que o usuário realize o monitoramento dos processos de forma remota através do seu smartphone ou desktop.

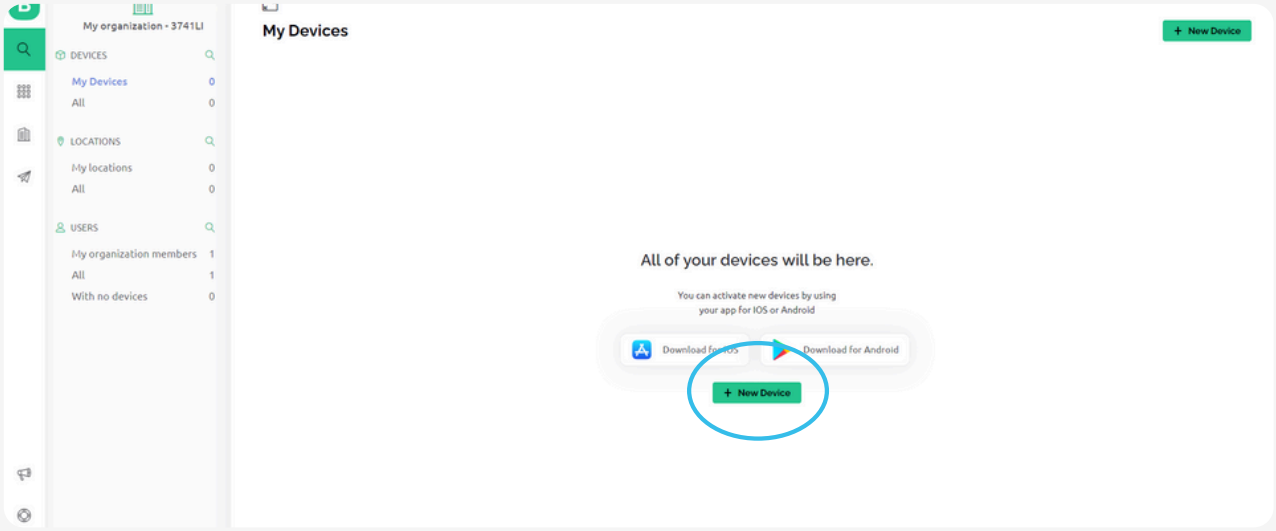
CURIOSIDADE
CONTA BLYNK

Para darmos inicio as configurações no Blynk é necessário criar uma conta na plataforma!

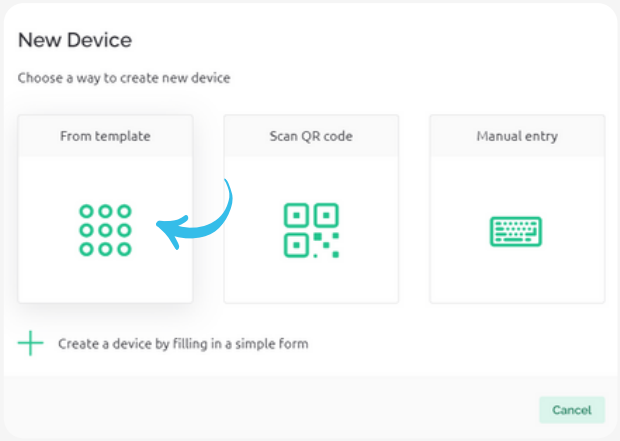
Acesse o link para o cadastro:
https://blynk.cloud/dashboard/register

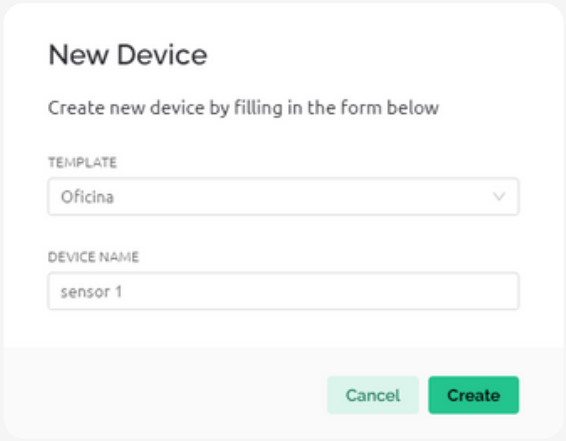
ADICIONANDO UM NOVO DEVICE

Após realizar o cadastro, vá para a aba “Devices” e clique em “New Device” para adicionar um novo dispositivo:

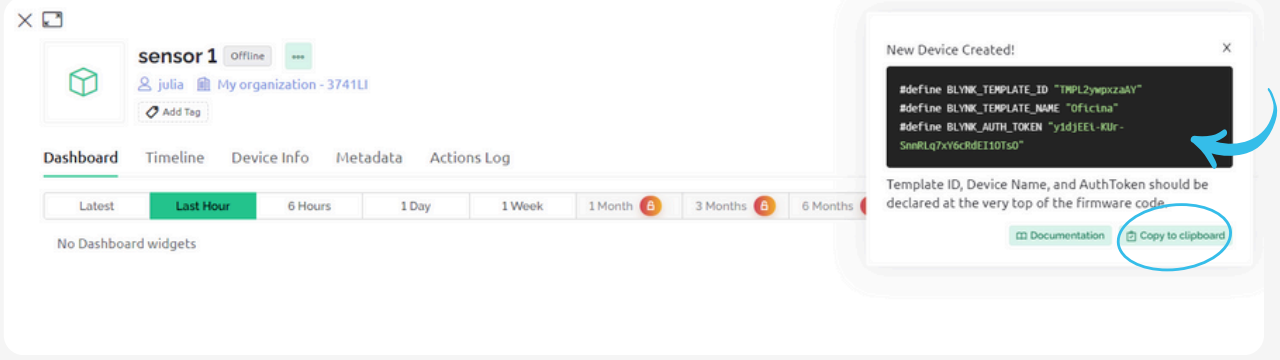


Em seguida, selecione a opção “From template” e dê um nome ao template e ao dispositivo que deseja criar:



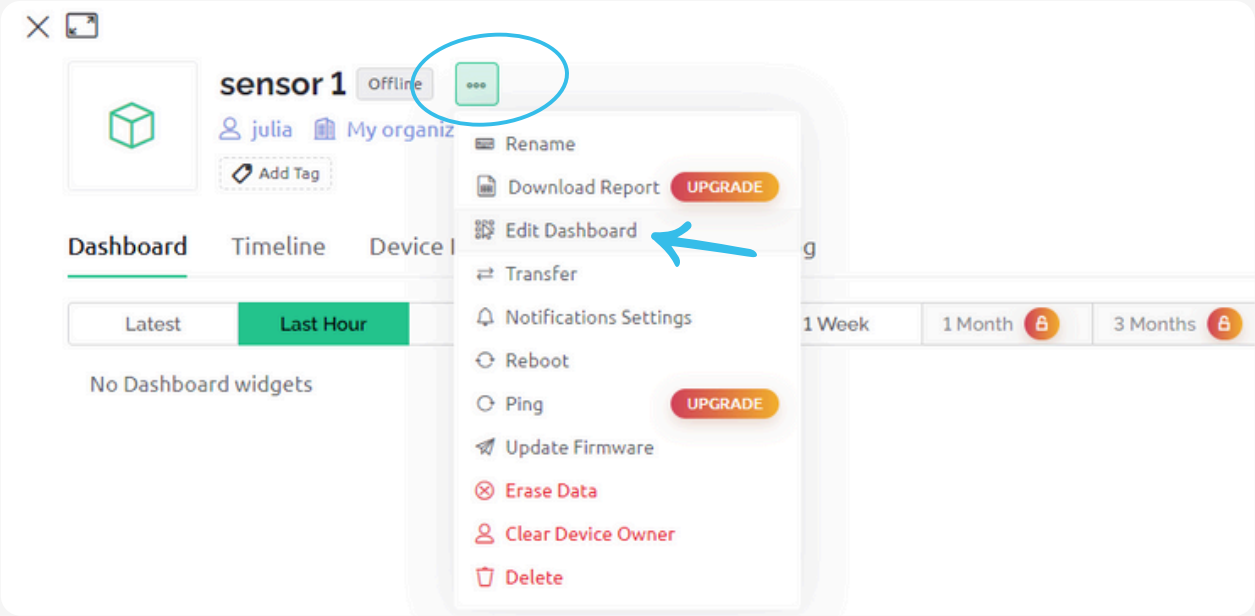


Pronto! Observe que a plataforma gera um ID e um Token para o seu dispositivo virtual. Clique em “Copy to clipboard” para copiar essas informações. Iremos usá-las posteriormente

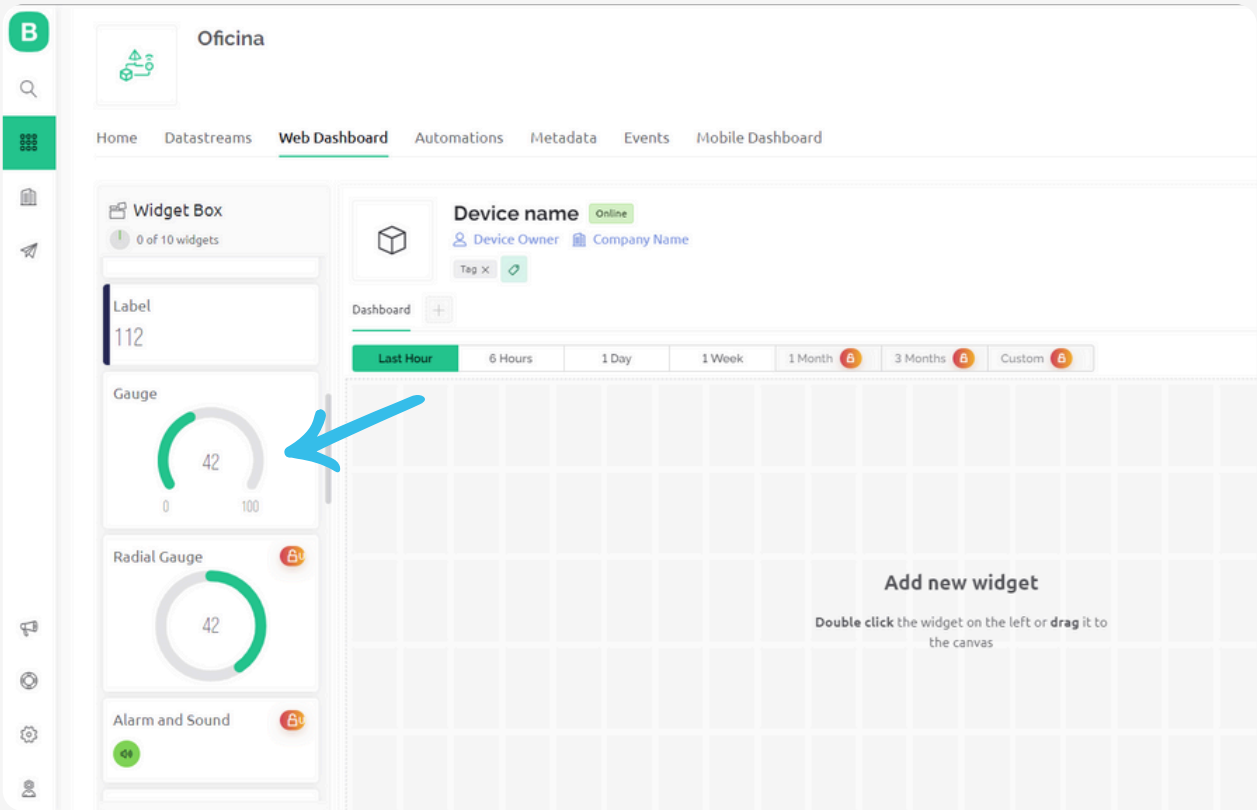


ADICIONANDO WIDGETS DOS SENSORES

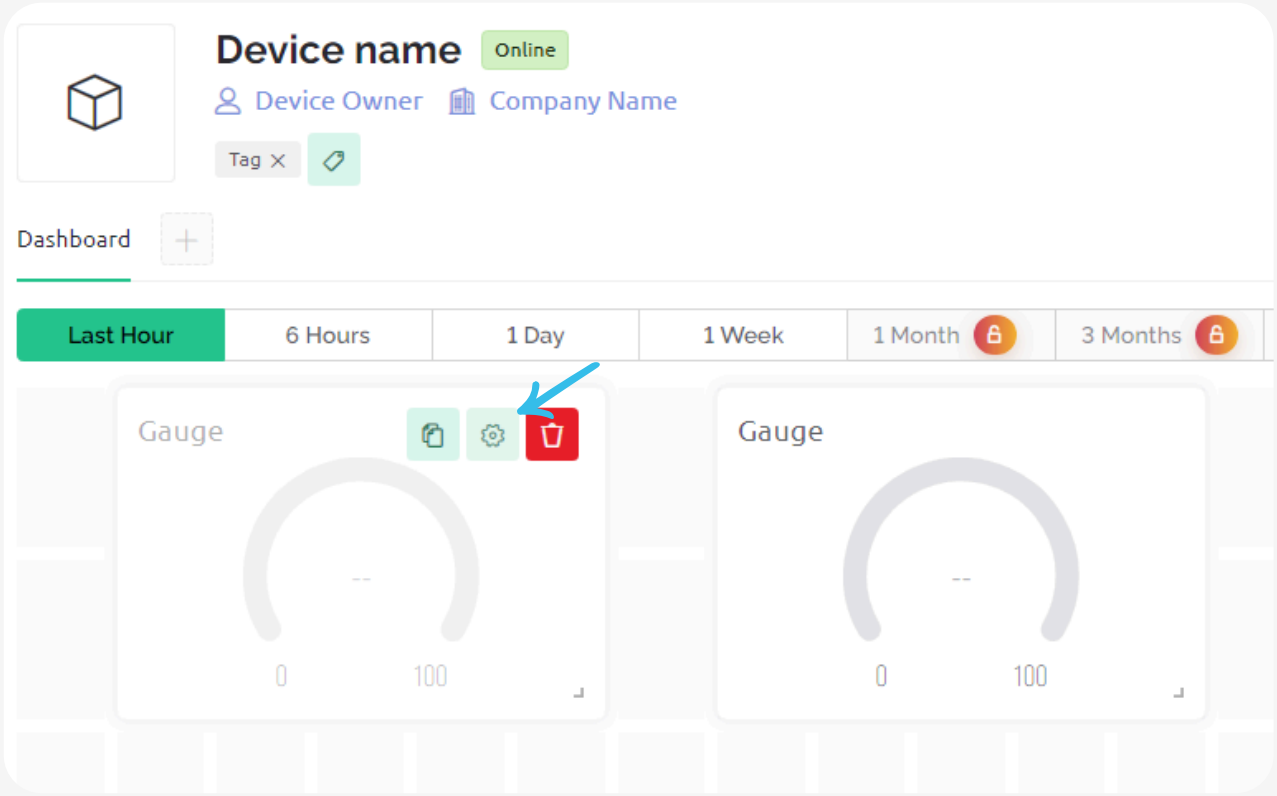
Clique nos 3 pontos e va em “Edit Dashboard”:



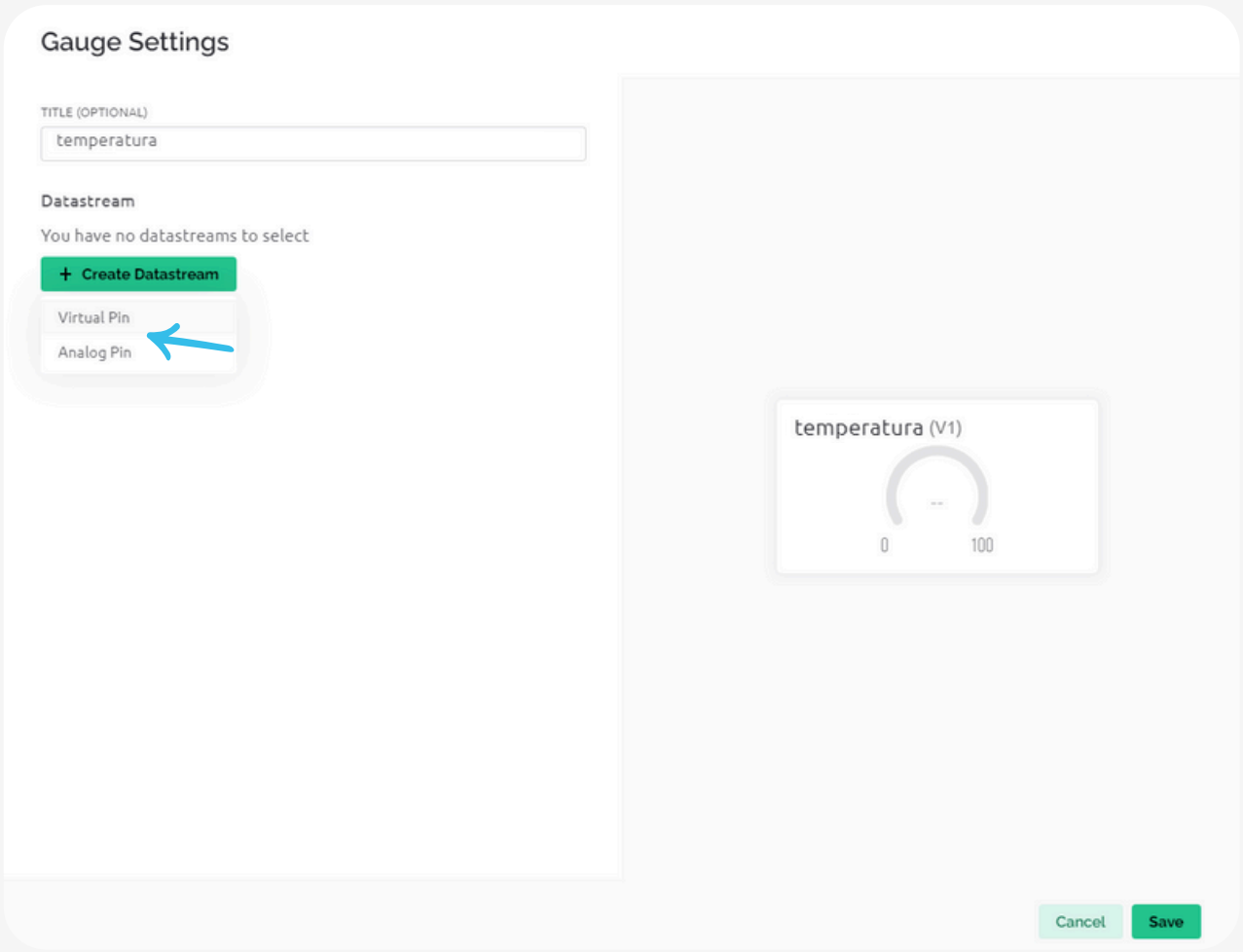
Procure por “Gauge”. Arraste 3 gauges para o dashboard:



Vá nas configurações (esse processo será feito em cada gauge):

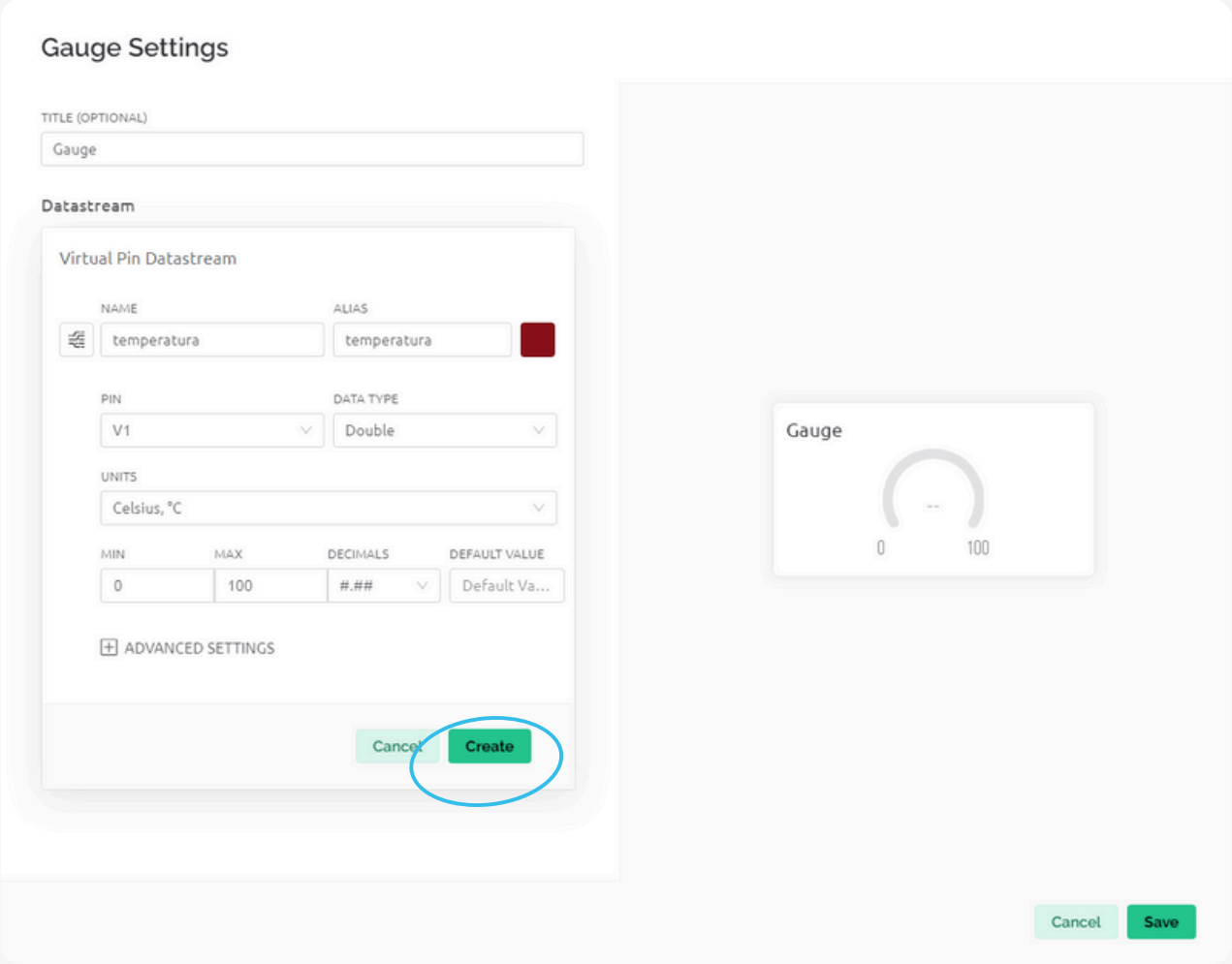


Insira o nome referente ao gauge e clique em “Create Datastream” e selecione a opção Virtual pin:



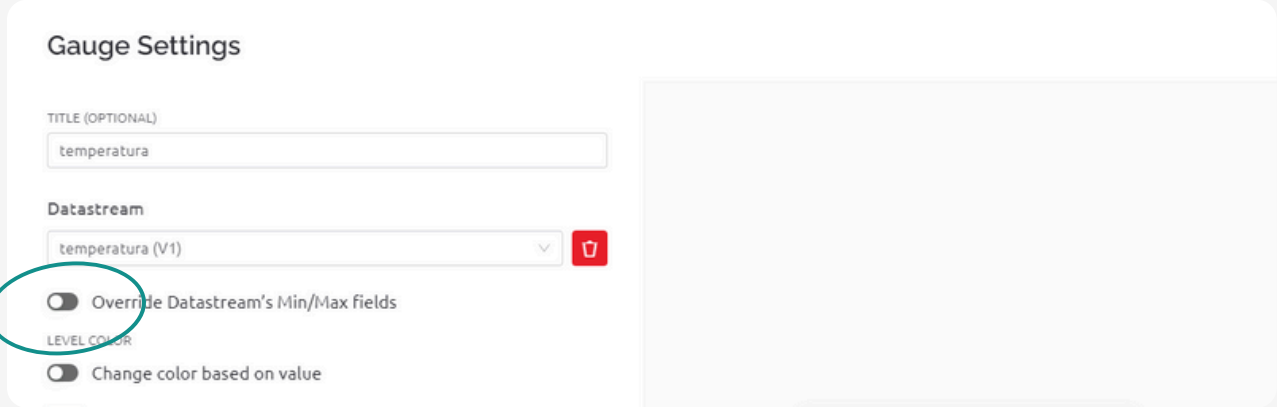
CONFIGURANDO GAUGES

Agora preencha com essas informações (esse gauge nos indicará o valor da temperatura) e depois clique em “Create”

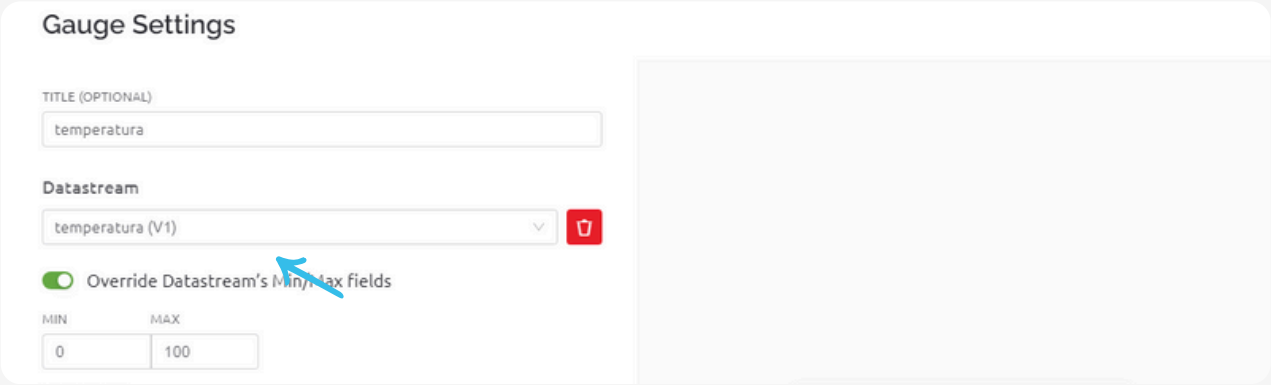


Iremos repetir os mesmo processo para os gauges de detecção de umidade e dos gases

Clique no botão indicado



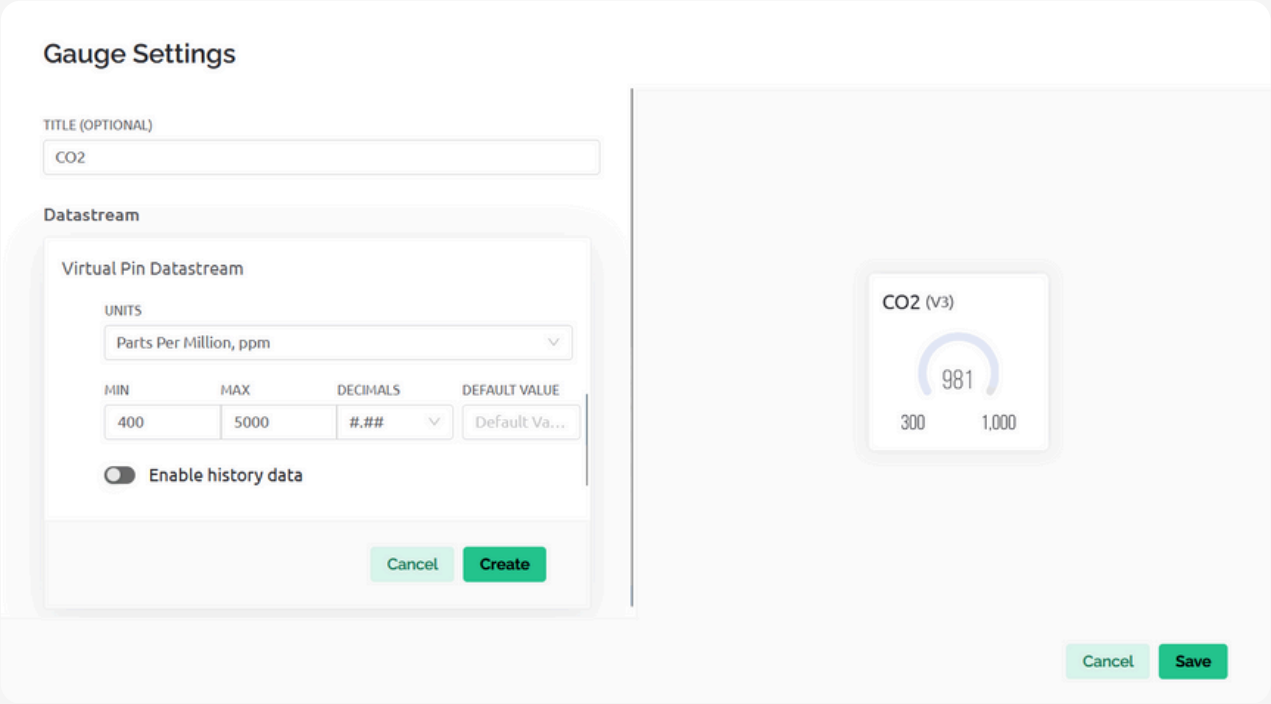
No campo “MIN” coloque o valor 0 e no campo “MAX” coloque o valor 100 (iremos considerar essa variação de temperatura, de 0 a 100C° graus)



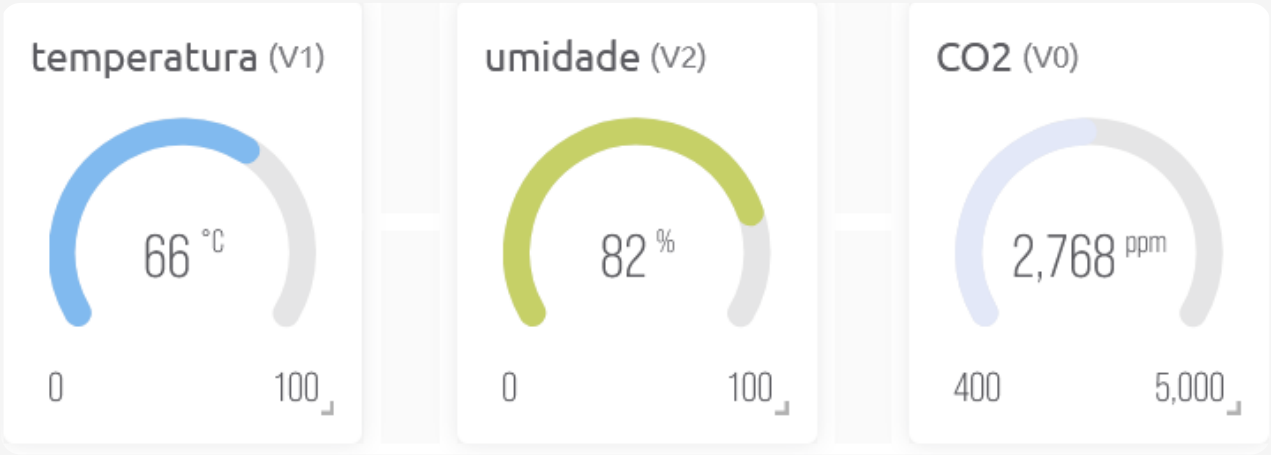
ATENÇÃO

Sinta-se livre para escolher os pinos virtuais que quiser para cada gauge (v1, v2, etc). Apenas fique ciente de quais pinos escolheu, pois posteriormente você irá inseri-los no código. E lembre-se, **cada gauge possui um pino virtual único**!

Por fim, iremos criar e configurar o gauge do gás que queremos detectar, no nosso caso o CO2, mas fique a vontade para identificar o gás que quiser, **lembre-se apenas de usar o sensor MQ correto**.
Na imagem abaixo configuramos o widget igualmente como fizemos nos passos anteriores, a diferença é que utilizamos a unidade PPM (Partes Por Milhão) que é comumente utilizada para fluidos gasosos. Além disso, atente-se à faixa de concentração do gás que deseja monitorar e insira essa faixa nos espaços “MIN” e “MAX”.



SIMULAÇÃO DOS SENSORES EM FUNCIONAMENTO



PROGRAMAÇÃO DO PROTÓTIPO:

INTEGRAÇÃO ARDUINO IDE COM A PLATAFORMA BLYNK

A integração da Arduino IDE com a plataforma Blynk foi fundamental para o sucesso do projeto. Através do uso de bibliotecas e códigos específicos, foi possível enviar os dados de detecção dos gases da placa ESP32 para a plataforma Blynk, onde eles são exibidos de forma clara e intuitiva para o usuário.

Essa integração permitiu o acompanhamento em tempo real do processo de compostagem, possibilitando a tomada de decisões baseada nos dados coletados. O processo usado para a integração das duas plataformas foi realizado através do uso do Token gerado pelo Blynk (é esse Token que faz a conexão do dispositivo IOT com o sistema em nuvem do app) e utilizar a biblioteca “Blynk” no Arduino IDE.

CÓDIGO

Agora vamos para o código! Desenvolvemos toda a programação com a linguagem C++. Você pode ter uma visão geral do código logo abaixo, mas não se preocupe! Iremos detalhar e explicar cada etapa desenvolvida na programação do protótipo.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <MQUnifiedsensor.h>
#include "DHTesp.h"

#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL2ywpxzaAY"
#define BLYNK_TEMPLATE_NAME "Oficina"
#define BLYNK_AUTH_TOKEN "y1djEEi-KUr-SnnRLq7xY6cRdEI1OTsO"

#define Board ("ESP-32")
#define Pin (32)
#define Type ("MQ-135")
#define Voltage_Resolution (3.3)
#define ADC_Bit_Resolution (12)
#define RatioMQ135CleanAir (3.6)

#define DHT_PIN 18

DHTesp dhtSensor;
MQUnifiedsensor MQ135(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin, Type);

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

void sendSensor()
{
  MQ135.update();
  float co2 = (MQ135.readSensor()) + 400;


  TempAndHumidity data = dhtSensor.getTempAndHumidity();
```

```
  Serial.println("Temp: " + String(data.temperature, 2) + "°C");
  Serial.println("Humidity: " + String(data.humidity, 2) + "%");
  Serial.println("CO2: " + String(co2) + " PPM");
  Serial.println("---");

  Blynk.virtualWrite(V1, data.temperature);
  Blynk.virtualWrite(V2, data.humidity);
  Blynk.virtualWrite(V3, co2);
}

void setup()
{
  Serial.begin(9600);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);

  MQ135.setRegressionMethod(1);
  MQ135.setA(110.47);
  MQ135.setB(-2.862);

  MQ135.init();

  float calcR0 = 0;
  for (int i = 1; i <= 10; i++)
  {
    MQ135.update();
    calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
  }
  MQ135.setR0(calcR0 / 10);

  MQ135.serialDebug(true);

  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  Blynk.run();
  sendSensor();
  delay(500);
}
```

PASSO A PASSO

O código está dividido em algumas etapas, iremos explicar o passo a passo de cada uma delas!

1.Inclusão das bibliotecas

```
#include <WiFi.h> //biblioteca que faz a conexão da esp32 com o wifi
#include <WiFiClient.h> //biblioteca que faz a conexão da esp32 com o wifi
#include <BlynkSimpleEsp32.h> //biblioteca do Blynk especifica para a esp32
#include <MQUnifiedsensor.h> //biblioteca que captura e trata os dados obtidos no MQ135
#include "DHTesp.h" // biblioteca que captura e trata os dados obtidos no DHT22
```

O código começa incluindo as bibliotecas necessárias para o funcionamento do projeto. Estas bibliotecas fornecem funções específicas que serão utilizadas ao longo do programa para realizar determinadas tarefas.

2. Definição das Variáveis

```
7   #define BLYNK_PRINT Serial
8   #define BLYNK_TEMPLATE_ID "id do seu template"
9   #define BLYNK_TEMPLATE_NAME "nome do seu template"
10  #define BLYNK_AUTH_TOKEN "seu token"
11
12  #define Board ("ESP-32")
13  #define Pin (32)
14  #define Type ("MQ-135")
15  #define Voltage_Resolution (3.3)
16  #define ADC_Bit_Resolution (12)
17  #define RatioMQ135CleanAir (3.6)
18
19  #define DHT_PIN 18
```

Imagine que as variáveis são nomes que irão armazenar o conteudo de alguma coisa. Por exemplo: x = 10, podemos dizer que a variável de nome “x” armazena o valor 10. Pensando nisso, o código acima indica as variáveis utilizadas ao longo do programa. Para definir o nome da variável faz-se uso da palavra reservada “#define” + o nome da sua variável + o conteudo da sua variavel.

Agora vamos ao significado das variáveis acima:

```
7   #define BLYNK_PRINT Serial
```

1. define que a comunicação de impressão (para debug) do Blynk será feita através da porta serial (Serial)

```
8   #define BLYNK_TEMPLATE_ID "id do seu template"
```

2. Define o ID do modelo do Blynk. Este ID é usado para identificar o modelo específico no aplicativo Blynk. Você deve inserir o seu ID do Blynk.

```
9   #define BLYNK_TEMPLATE_NAME "nome do seu template"
```

3. Define o nome do modelo do Blynk. Este nome é exibido no aplicativo Blynk para identificar o modelo.

```
10    #define BLYNK_AUTH_TOKEN "seu token"
```

4.Define o token de autenticação do Blynk. Este token é necessário para autenticar o dispositivo com o servidor Blynk.

```
12    #define Board ("ESP-32")
13    #define Pin (32)
14    #define Type ("MQ-135")
```

5. Na linha 12 define o tipo de placa utilizada, no caso, ESP-32. Já na linha 13 define o pino ao qual o sensor MQ-135 está conectado, no caso, pino 32. Por último, na linha 14 define o tipo de sensor, no caso, MQ-135.

```
15    #define Voltage_Resolution (3.3)
16    #define ADC_Bit_Resolution (12)
17    #define RatioMQ135CleanAir (3.6)
```

6. Na linha 15 define a resolução de voltagem, no caso, 3.3V. Já na linha 16 define a resolução de bits do conversor analógico-digital (ADC), no caso, 12 bits. Por fim, na linha 17 define a relação de ar limpo do sensor MQ-135, que é utilizada no cálculo da concentração de CO2.

```
19    #define DHT_PIN 18
```

7. Finalizando a etapa de definição de varáveis temos na linha 19 a definição do pino ao qual o sensor DHT22 está conectado, no caso, pino 18.

2. Configuração dos Sensores e Variáveis Globais:

```
21  DHTesp dhtSensor;
22  MQUnifiedsensor MQ135(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin, Type);
23
24  char auth[] = BLYNK_AUTH_TOKEN;
25  char ssid[] = "Wokwi-GUEST";
26  char pass[] = "";
27
```

O trecho de código acima é responsável por configurar e inicializar os objetos dos sensores DHT22 e MQ-135, bem como as variáveis necessárias para a conexão com a rede Wi-Fi e o servidor Blynk. Essas variáveis são chamadas de variáveis globais pois poderão ser acessadas por diversas partes do código.

Explicação do código:

1. Instância do Objeto do Sensor DHT22:
Nessa linha cria-se um objeto dhtSensor do tipo DHTesp (referente a biblioteca relacionada ao DHT22 instalamos na etapa anterior), que será usado para interagir com o sensor DHT22 de umidade e temperatura.

```
22  MQUnifiedsensor MQ135(Board, Voltage_Resolution, ADC_Bit_Resolution, Pin, Type);
```

2. Instância do Objeto do Sensor MQ-135:
Na linha 27 é criado um objeto MQ135 do tipo MQUnifiedsensor (que se refere a biblioteca que instalamos, mostrada na etapa anterior), que será usado para interagir com o sensor MQ-135 de CO2. Os parâmetros passados são as definições previamente feitas para a placa (Board), resolução de voltagem (Voltage_Resolution), resolução de bits do ADC (ADC_Bit_Resolution), pino ao qual o sensor está conectado (Pin) e o tipo de sensor (Type).

```
24    char auth[] = BLYNK_AUTH_TOKEN;
```

3. Definição do Token de Autenticação do Blynk:
Na linha acima define-se a variável auth como uma string contendo o token de autenticação do Blynk. Esse token é necessário para conectar o dispositivo ao servidor Blynk. Essa variável irá armazenar a variável “BLYNK_AUTH_TOKEN“ que contém o token do blynk.

```
25    char ssid[] = "nome da sua rede Wifi";
26    char pass[] = "senha da sua rede Wifi";
```

2. Definição do Nome e senha da Rede Wi-Fi:
Na linha 25 é definida a variável “ssid” como uma string contendo o nome da rede Wi-Fi, e na linha 26 define-se a variável “pass” como uma string contendo a senha da rede Wi-Fi à qual o dispositivo irá se conectar.

3. Função sendSensor():

```
28  void sendSensor()
29  {
30    MQ135.update();
31    float co2 = (MQ135.readSensor()) + 400;
32
33
34    TempAndHumidity data = dhtSensor.getTempAndHumidity();
35
36    Serial.println("Temp: " + String(data.temperature, 2) + "°C");
37    Serial.println("Humidity: " + String(data.humidity, 2) + "%");
38    Serial.println("CO2: " + String(co2) + " PPM");
39    Serial.println("---");
40
41    Blynk.virtualWrite(V1, data.temperature);
42    Blynk.virtualWrite(V2, data.humidity);
43    Blynk.virtualWrite(V3, co2);
44  }
45
```

A função sendSensor() tem o objetivo de ler os valores dos sensores de temperatura, umidade e CO2 e enviar esses valores para o aplicativo Blynk. Para isso, a função realiza as seguintes etapas:

Explicação do código

```
30    MQ135.update();
```

1. Atualização do Sensor MQ-135:
Atualiza o sensor MQ-135 para obter a leitura mais recente do sensor.

```
31    float co2 = (MQ135.readSensor()) + 400;
32    TempAndHumidity data = dhtSensor.getTempAndHumidity();
```

2. Cálculo da Concentração de CO2 e Leitura da Temperatura e Umidade:
Na linha 31 lê-se a concentração de CO2 do sensor MQ-135 e soma essa concentração em 400 unidades devido à calibração, pois o valor mínimo de CO2 na atmosfera é de 400 ppm (Partes Por Milhão). Na linha 32 utilizamos o objeto “dhtSensor“ para obter os valores de temperatura e umidade do sensor DHT22.

```
33
34    Serial.println("Temp: " + String(data.temperature, 2) + "°C");
35    Serial.println("Humidity: " + String(data.humidity, 2) + "%");
36    Serial.println("CO2: " + String(co2) + " PPM");
37    Serial.println("---");
38
```

3. Impressão dos Valores no Monitor Serial:
As linhas acima imprimem os valores de temperatura, umidade e CO2 no monitor serial, para fins de debug e monitoramento.

```
39    Blynk.virtualWrite(V1, data.temperature);
40    Blynk.virtualWrite(V2, data.humidity);
41    Blynk.virtualWrite(V3, co2);
```

4. Envio dos Valores para o Blynk:
Essas linhas enviam os valores de temperatura, umidade e CO2 para o aplicativo Blynk através dos pinos virtuais V1, V2 e V3, respectivamente. Esses valores serão exibidos no aplicativo Blynk para monitoramento em tempo real.

3. Função setup():

```
43
44  void setup()
45  {
46    Serial.begin(9600);
47    dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
48
49    MQ135.setRegressionMethod(1);
50    MQ135.setA(110.47);
51    MQ135.setB(-2.862);
52
53    MQ135.init();
54
55    float calcR0 = 0;
56    for (int i = 1; i <= 10; i++)
57    {
58      MQ135.update();
59      calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
60    }
61    MQ135.setR0(calcR0 / 10);
62
63    MQ135.serialDebug(true);
64
65    Blynk.begin(auth, ssid, pass);
66  }
```

A função setup() acima é responsável pela configuração inicial do sistema, onde são inicializados os sensores, calibrados os valores do sensor MQ-135, e estabelecida a conexão com o servidor Blynk para enviar os dados dos sensores.

Explicação do código

```
46    Serial.begin(9600);
```

1. Inicialização da Comunicação Serial:
A linha acima Inicia a comunicação serial com a velocidade de 9600 baud. Isso significa que você está definindo uma taxa de transmissão de 9600 bits por segundo entre o dispositivo e o computador.

```
47    dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
```

2. Configuração do Sensor DHT22:
Configura o sensor DHT22 para utilizar o pino especificado (armazenado na variável “DHT_PIN”) e o tipo de sensor DHT22.

```
48
49    MQ135.setRegressionMethod(1);
50    MQ135.setA(110.47);
51    MQ135.setB(-2.862);
52
```

3. Configuração do Sensor MQ-135:
As linhas acima configuram o método de regressão para calcular a concentração de CO2, bem como os valores dos coeficientes a e b para o sensor MQ-135 (esses métodos de regressão são realizados pela biblioteca “MQUnifiedSensor”, bem como os valores de “a” e “b” são obtidos através da biblioteca)

```
53  MQ135.init();
54
55  float calcR0 = 0;
56  for (int i = 1; i <= 10; i++)
57  {
58    MQ135.update();
59    calcR0 += MQ135.calibrate(RatioMQ135CleanAir);
60  }
61  MQ135.setR0(calcR0 / 10);
62
```

4. Inicialização e Calibração do Sensor MQ-135:
Na linha 53 ocorre a Inicialização do sensor MQ-135 para leitura. No outro trecho de código é realizada a calibração do sensor MQ-135. O loop é executado 10 vezes para obter a média dos valores de calibração. Esse passo precisa ocorrer para que os dados obtidos pelo sensor sejam mais confiáveis.

```
63  MQ135.serialDebug(true);
```

5. Ativação do Modo de Depuração Serial do Sensor MQ-135:
A linha acima ativa o modo de depuração serial do sensor MQ-135 para exibir informações de depuração.

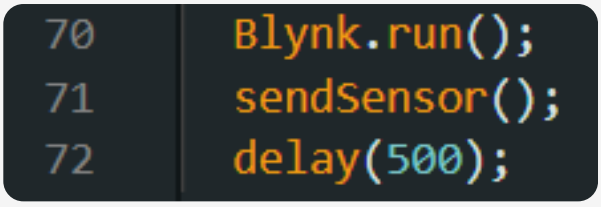
```
65  Blynk.begin(auth, ssid, pass);
```

6. Conexão com o Servidor Blynk:
Inicia a conexão com o servidor Blynk utilizando o token de autenticação, nome da rede Wi-Fi e senha fornecidos (variavéis citadas nos passos anteriores e que armazenam os valores citados).

4. Função loop():

A função acima representa o loop principal (loop()) do código e é responsável por executar continuamente as operações necessárias para o funcionamento do sistema. Ele verifica e processa os eventos do servidor Blynk e realiza a leitura dos sensores periodicamente, enviando os dados para o aplicativo Blynk em um intervalo de 500 milisegundos.

Explicação do código



O código acima pode ser dividido em 3 etapas:
1. Execução das Operações do Servidor Blynk:
Na linha 70 é realizada a verificação e o processamento dos eventos do servidor Blynk, permitindo a comunicação bidirecional entre o dispositivo e o aplicativo Blynk.

2. Chamada da Função sendSensor():
Já a linha 71 chama a função sendSensor() para realizar a leitura dos sensores e enviar os dados para o aplicativo Blynk.

3. Atraso para Evitar Sobrecarga do Servidor Blynk:
A linha 72 adiciona um atraso de 500 milissegundos (0.5 segundos) entre as iterações do loop, evitando sobrecarga no servidor Blynk e permitindo um intervalo regular de leitura dos sensores.

AGORA... MÃOS À OBRA!

Chegou a hora de programar! Agora que você já entendeu o código e suas funcionalidades, abra o seu Arduino IDE e replique o código ensinado! Fique à vontade para realizar as alterações e realizar os testes que quiser. Boa prática!

AGRADECIMENTOS

Prezados parceiros,

Gostaríamos de expressar nosso sincero agradecimento pela valiosa parceria no desenvolvimento do nosso projeto. A colaboração entre a Facepe, o IFPE e a Lógica Ambiental foi essencial para o sucesso de nossas iniciativas.
A Facepe proporcionou suporte financeiro e incentivo à pesquisa e inovação. O IFPE, com sua estrutura acadêmica e corpo docente dedicados, criou um ambiente propício para o desenvolvimento de ideias e capacitação. A Lógica Ambiental trouxe uma perspectiva prática e sustentável, aplicando soluções em contextos reais e promovendo práticas ambientalmente responsáveis.
Agradecemos a todos pelo empenho, dedicação e profissionalismo ao longo do projeto. Esta sinergia resultou em avanços significativos e fortaleceu nossos laços, criando um modelo exemplar de cooperação.
Esperamos continuar esta parceria próspera e enfrentar novos desafios juntos, promovendo o desenvolvimento científico, tecnológico e ambiental de nossa sociedade.

Com gratidão e apreço,
Equipe EcoSniff.