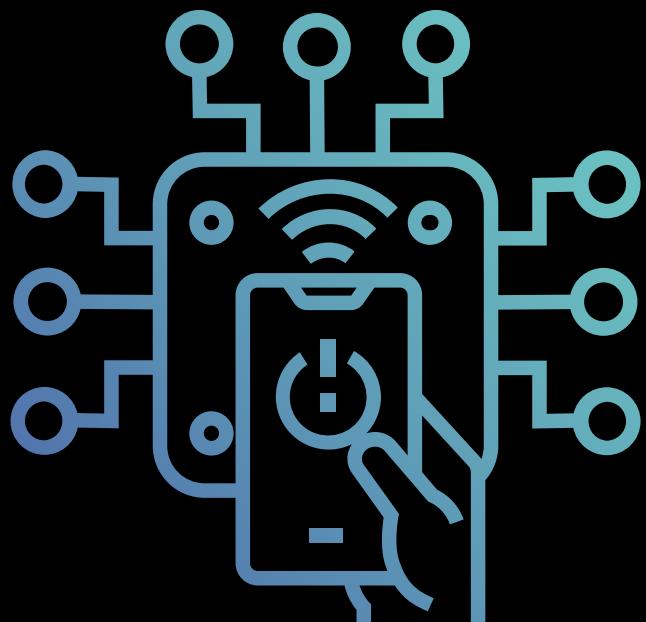


E-BOOK INTERNET DAS COISAS PARA INICIANTES COM ESP-32



ELETROÔNICA OMEGA

Sumário

1 INTRODUÇÃO	2
2 CONCEITOS IMPORTANTES	3
3 COMPOSIÇÃO DO ESP-32	12
4 BAIXANDO A IDE DO ARDUINO.....	15
5 CONFIGURANDO O ESP-32	19
6 CONHECENDO O BLYNK.....	21
7 SEQUENCIAL DE LEDS	24
8 LIGANDO LÂMPADA E VENTILADOR	45
9 MONITORAMENTO DE TEMPERATURA	57
10 ALERTA DE CHUVA	67
11 CONTROLANDO DISPLAY LCD	77
12 CONTROLE DE ACESSO COM RFID	82
13 IRRIGAÇÃO AUTOMÁTICA	94
14 FINALIZAÇÃO.....	105

1

Introdução

Esse E-book foi desenvolvido pela **Arduino Ômega** e destina-se aos makers iniciantes que querem aprender sobre Internet das Coisas (IoT) utilizando a placa **ESP-32** de forma prática e divertida! Aqui você vai aprender como desenvolver projetos com os materiais disponíveis no **Kit Internet das Coisas com ESP-32**. Além disso, será apresentado qual a função dos principais componentes, conceitos de eletrônica, elétrica e programação. Ao finalizar este e-book você será capaz de construir os seus próprios projetos !!

Kit Internet das Coisas (IoT)

Este é o Kit ideal para o maker que está em busca de projetos de **Internet das Coisas (IoT)** e **automação residencial**. Com este kit você poderá fazer projetos para apagar e acender lâmpadas por Wi-Fi, automatizar a irrigação do seu jardim, controlar relés pela internet, monitorar sua casa com os dados de temperatura, umidade de solo, pressão atmosférica, chuva, entre outros.



Você pode adquirir esse Kit
[Clicando Aqui](#) !!



Conceitos Importantes

Antes de darmos início em nossos projetos, é importante conhecer alguns **Conceitos** que serão utilizados no decorrer do E-book.

Internet of Things (IoT)

Internet of Things, traduzido para a língua portuguesa torna-se Internet das Coisas, descreve objetos físicos incorporados a **sensores, software e outras tecnologias** com o objetivo de **conectar e trocar dados com outros dispositivos e sistemas pela internet**. Esses dispositivos variam de objetos domésticos comuns a ferramentas industriais sofisticadas, como por exemplo sensores.

De acordo com o **Instituto Gartner** o número de dispositivos conectados à IoT deve chegar em **50 bilhões** até o ano de **2022!**



O que é ESP-32 ??

Em nossos projetos vamos utilizar o **ESP-32**, ele nada mais é do que uma placa de desenvolvimento de código aberto que utiliza o chip **ESP32**. É muito semelhante ao **Arduino Uno**, visto que ambos são **microcontroladores**. A vantagem de se utilizar o ESP-32 é ele possuir **Wi-Fi** e **Bluetooth** nativo, dessa forma essa placa é ideal para projetos de automação residencial e internet das coisas.



O **ESP-32** possui 36 pinos digitais e 16 podem ser utilizados como saída PWM (controle de dispositivos variando a intensidade).

Outro diferencial é a possibilidade de fazer a **programação** da placa via **OTA** (Over The Air), assim, através do **Wi-Fi** é possível **programar** o **ESP-32**.

Além disso, é possível programar a placa com as linguagens de programação: **Lua**, **Python**, **JavaScript** ou até mesmo com a **IDE do Arduino**.

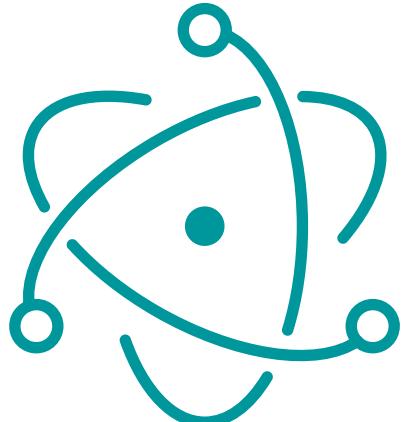


Entradas e Saídas

Assim como o **Arduino**, o **ESP-32** possui entradas e saídas.

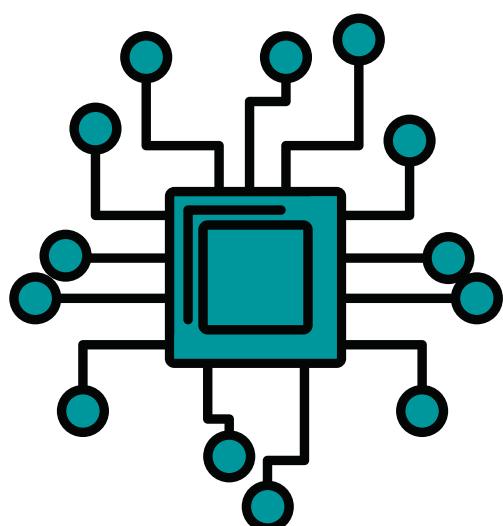
De forma resumida as entradas são por exemplo apertar um botão, e assim, quando ele é acionado é enviado um sinal para o **microcontrolador** do **ESP-32** que pode ativar algo, como por exemplo um **LED** ou um **motor**.

Já as saídas são exatamente o que é ativado pelo Arduino ou ESP-32, como um **LED**, **motor**, **buzzer**, entre outros.



Microcontroladores

Os **microcontroladores** são uma junção de Software e Hardware que através da programação conseguimos controlá-los para desempenhar tarefas. Ele é um tipo especial de circuito integrado, visto que conseguimos programar os **microcontroladores** para realizar tarefas específicas. Em breve vamos descobrir diversas coisas que podemos fazer utilizando **microcontroladores**. No caso do **ESP** o microcontrolador utilizado é o **ESP-32**.

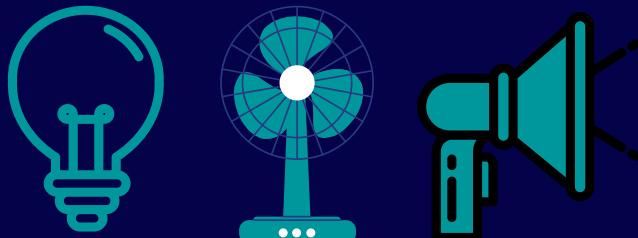


Sinais Digitais

Como visto anteriormente, o **ESP-32** possui 13 pinos digitais, mas afinal, você sabe o que é um **Sinal Digital**?

O **Sinal Digital** possui uma quantidade limitada, geralmente é representado por dois níveis (Alto ou Baixo). Assim é definido para instantes de tempo e o conjunto de valores que podem assumir é limitada. Podemos usar como **exemplos** de **Saídas Digitais**:

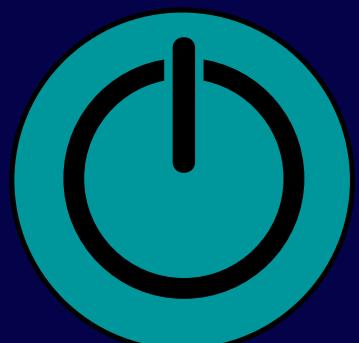
- Lâmpadas;
- Buzinas;
- Ventiladores.



Observe que todos esses equipamentos estão ou ligados ou desligados, não há uma variação contínua.

E também existem as Entradas Digitais, por exemplo:

- Chaves seletoras;
- Fins de Curso;
- Botões.



Sinais Analógicos

Além de portas digitais, o **ESP-32** também possui uma **Porta Analógica**. O **Sinal Analógico** é um sinal que pode assumir infinitos valores em um intervalo de tempo. **Exemplos** de **Sinais Analógicos** são:

- [Sensores de Temperatura](#);
- [Sensores de Umidade](#);
- [Sensores de Pressão](#);
- [Potenciômetros](#).



Simplificando...

Para simplificar, pense no interruptor e um dimmer de uma lâmpada...

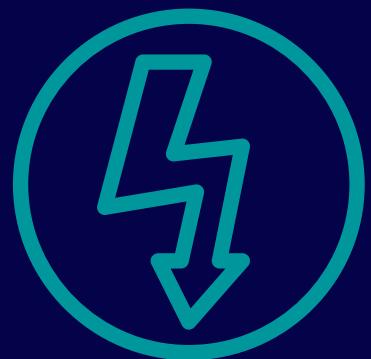
Ou o interruptor está **acionado** e a lâmpada está **acesa** ou está **desacionado** e a lâmpada **apagada**, não existe outro estado. Dessa forma o interruptor é considerado um Sinal Digital.

Já o **dimmer** controla a intensidade da lâmpada, girando o potenciômetro conseguimos regular a intensidade de brilho. Esse é considerado o sinal analógico, pois existem infinitos valores.



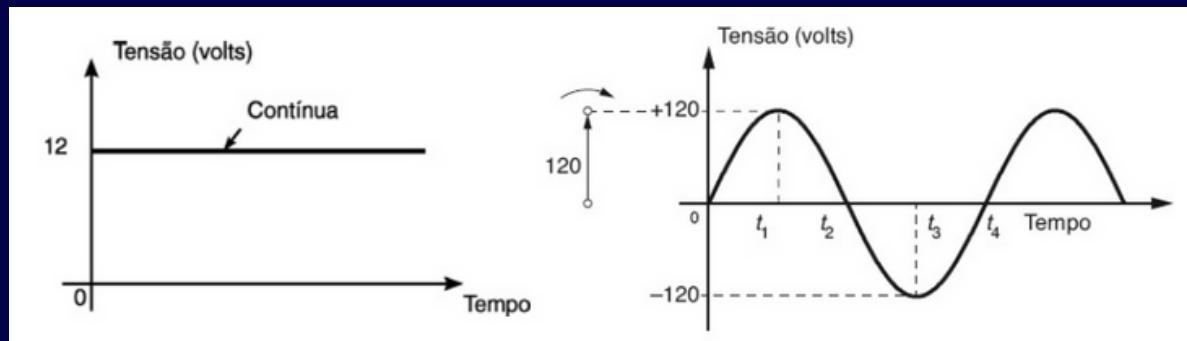
Tensão Elétrica

Também conhecida como **Diferença de Potencial**, a **tensão** é a grandeza que mede a diferença de potencial elétrico entre dois pontos. É a força necessária para mover os elétrons e criar assim uma **Corrente Elétrica**. O equipamento utilizado para medir a tensão é conhecido como **voltímetro**, no **Sistema Internacional** a unidade de medida é **Volts**, onde o símbolo é **V**.



Tipos de Tensão

A tensão pode ser **Contínua** ou **Alternada**. Na contínua ela não muda de polaridade com o passar do tempo, já na alternada é ao contrário, ou seja, muda de polaridade com o passar do tempo.



Exemplos

- A **Contínua** não muda de polaridade com o passar do tempo. A tensão das pilhas é contínua.
- Já a tensão **Alternada** a polaridade será alternada de acordo com a frequência, no caso de uma tomada no Brasil, a frequência normal é de 60Hz, o que quer dizer que a polaridade desta tensão vai alternar 60 vezes por segundo.



Fórmulas

Podemos calcular a **Tensão Elétrica** utilizando a **Lei de Ohm**, onde a tensão é igual a resistência vezes a corrente elétrica.

Assim, temos a seguinte fórmula:

Tensão = Resistência x Corrente



Resumindo: $U = R \times I$

OBS: U = Tensão
 R = Resistência
 I = Corrente

Também é possível calcular a tensão se caso soubermos o valor da **Corrente** e da **Potência** elétrica, onde tensão elétrica é igual a potência dividida pela corrente.

Assim, temos a seguinte fórmula:

Tensão = $\frac{\text{Potência}}{\text{Corrente}}$



Resumindo: $U = \frac{P}{I}$

OBS: U = Tensão
 P = Potência
 I = Corrente

OBS: O termo **Voltagem** apesar de ser muito falado não existe, é apenas um termo popular.

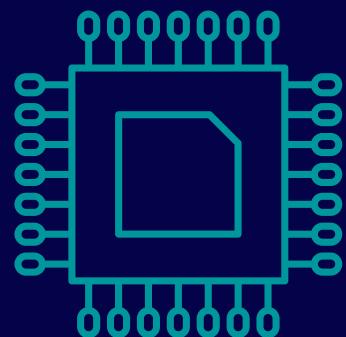
Corrente Elétrica

A **Corrente Elétrica** é um fenômeno físico onde ocorre o movimento de cargas elétricas, como os elétrons, que acontece no interior de diferentes materiais em razão da aplicação de uma diferença de potencial elétrico. A corrente elétrica é uma grandeza escalar, sua unidade de medida de acordo com o Sistema Internacional de Unidades, é o **Ampère**, cujo símbolo é **A**. Essa unidade mede o módulo da carga elétrica que atravessa a secção transversal de um condutor a cada segundo e por isso também pode ser escrita como **Coulombs Por Segundo**, onde o símbolo é **C/s**.

Tipos de Corrente

Assim como a **Tensão Elétrica**, existem dois tipos de corrente, a **Corrente Alternada (CA)** e a **Corrente Contínua (CC)**.

- **Corrente Contínua:** Não tem variação ao longo do tempo e se mantém praticamente constante, os elétrons são forçados a deslocar-se em sentido único. Esse tipo de corrente é comum em dispositivos que utilizam baixas tensões, como eletrônicos em geral.
- **Corrente Alternada:** A corrente alternada varia ao longo do tempo. Nesse tipo de corrente, uma rápida inversão de polaridade do potencial elétrico faz com que os elétrons se desloquem em vai e vem em torno de uma posição fixa. Corrente elétrica alternada é utilizada principalmente em motores elétricos e na transmissão de eletricidade, a corrente que chega às nossas casas é uma corrente elétrica alternada.



Fórmulas

Para se saber qual a intensidade da **Corrente Elétrica** em um condutor, é possível utilizar a equação de carga elétrica pelo tempo.

Assim, temos a seguinte fórmula:

$$\text{Intensidade de Corrente} = \frac{\text{Carga elétrica}}{\text{Intervalo de tempo}}$$



Resumindo: $I = \frac{Q}{\Delta t}$

OBS: Q = Carga elétrica
 Δt = Intervalo de tempo
 I = Corrente

Utilizando a **Tensão Elétrica** e a **Resistência** também é possível calcularmos a corrente, onde corrente é igual tensão elétrica dividido pela resistência.

Assim, temos a seguinte fórmula:

$$\text{Corrente} = \frac{\text{Tensão}}{\text{Resistência}}$$



Resumindo: $I = \frac{U}{R}$

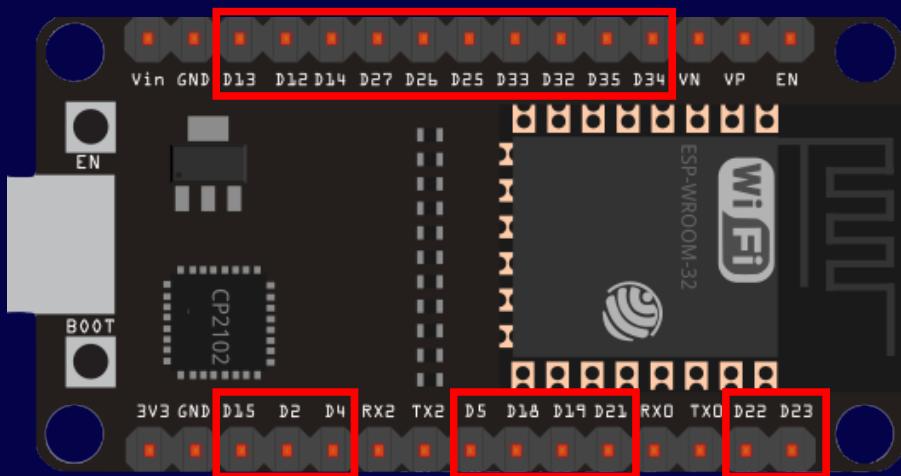
OBS: U = Tensão
 R = Resistência
 I = Corrente



3 Composição do ESP-32

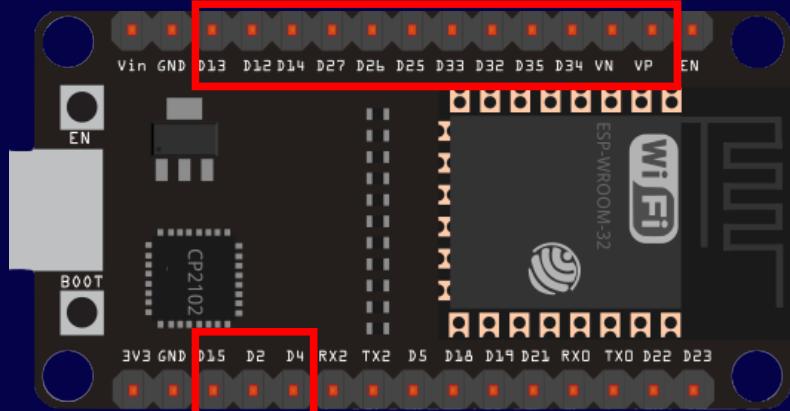
Bem, agora que já conhecemos os principais conceitos que vão ser utilizados, vamos conhecer um pouco mais sobre o **NodeMCU**.

Pinos Digitais



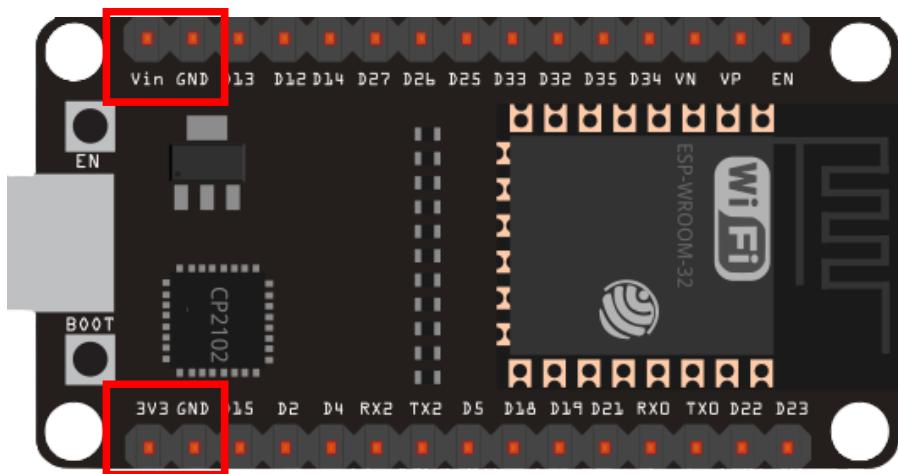
As portas **D2** a **D35** são os pinos de entradas e saídas digitais, usadas para conexões de **módulos** e **sensores**.

Pino Analógico



Os pinos **D13** a **VP** e **D15** a **D4** funcionam como pinos analógicos

Pinos para Alimentação

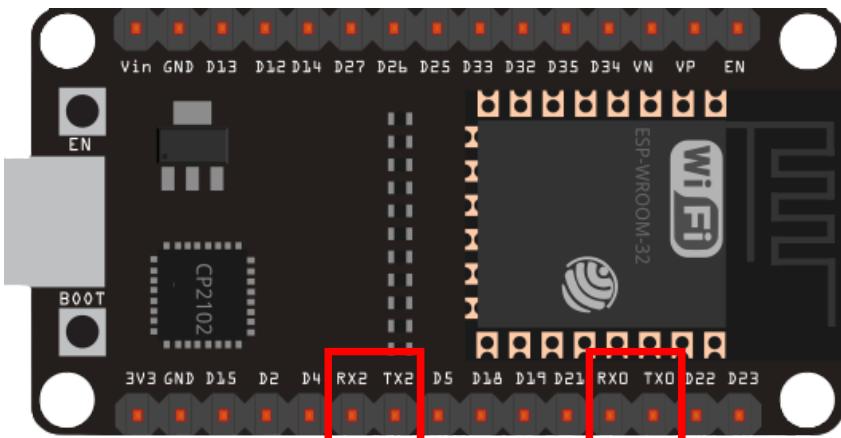


Pinos GND: É o terra que funciona como o negativo para todos os circuitos e dispositivos externos. O GND é comum a todas as portas, ou seja, ele é compartilhado, diferente das portas digitais.

Pino 3V3: Pino que fornece 3,3 volts para a alimentação de dispositivos externos.

Pino Vin: A porta Vin fornece a mesma tensão que é recebida pelo pino de alimentação externo.

Pinos para Comunicação

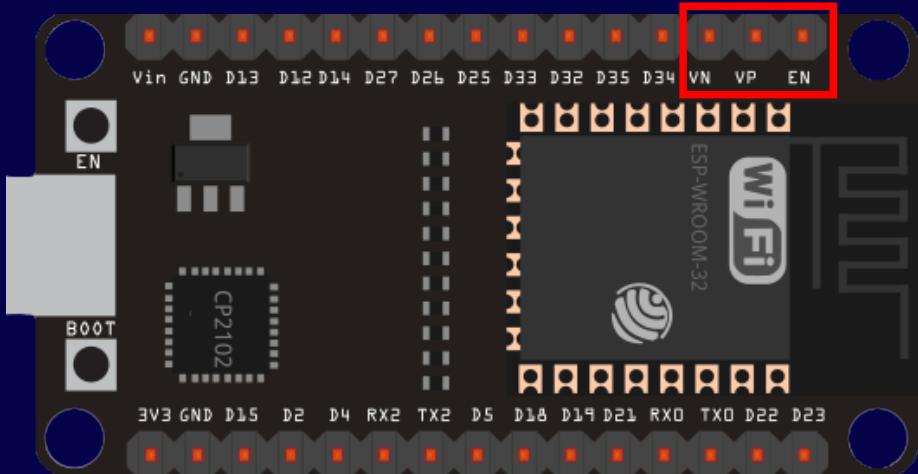


RX: Receiver, recebe bits.

TX: Transmitter, envia bits.

São as portas utilizada para fazer a **Comunicação Serial**.

Outros Pinos



Pino EN: O chip ESP32 é habilitado quando o pino EN está em nível lógico **alto**. Quando está em nível lógico **baixo**, o chip funciona com a potência mínima.

Pino VP: Sensor VP.

Pino VN: Sensor VN.

EN: É usado para reiniciar o NodeMCU.

Entrada Micro Usb: Usada para a alimentação da placa e para enviar a programação.

BOOT: permite a gravação do programa no ESP32.

4

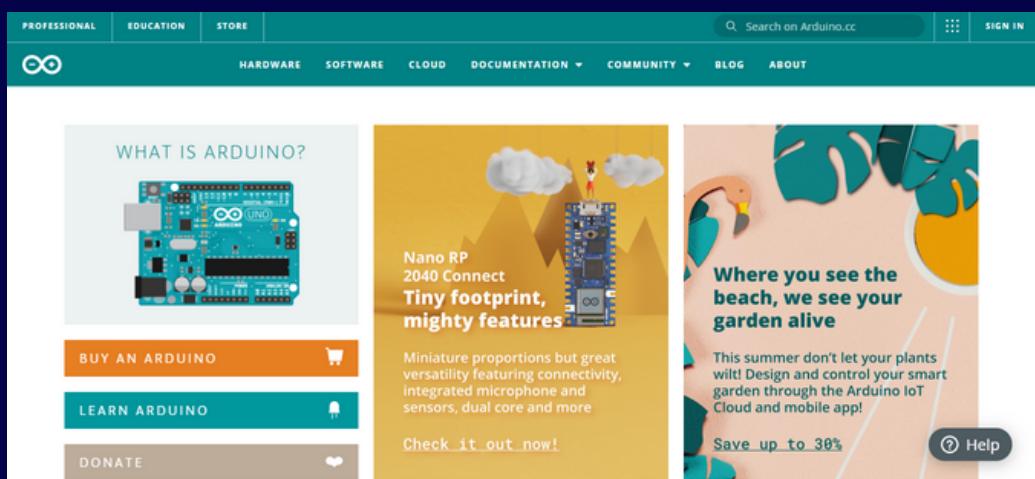
Baixando a IDE do Arduino

Isso mesmo !! Para **programarmos** nossa placa vamos utilizar a **IDE do Arduino** !

Clique Aqui para baixar a **IDE do Arduino**. Caso tenha dúvidas, siga o passo a passo abaixo.

1

Acesse o site oficial do Arduino [Clicando aqui](#).



2

Clique em **Software**.



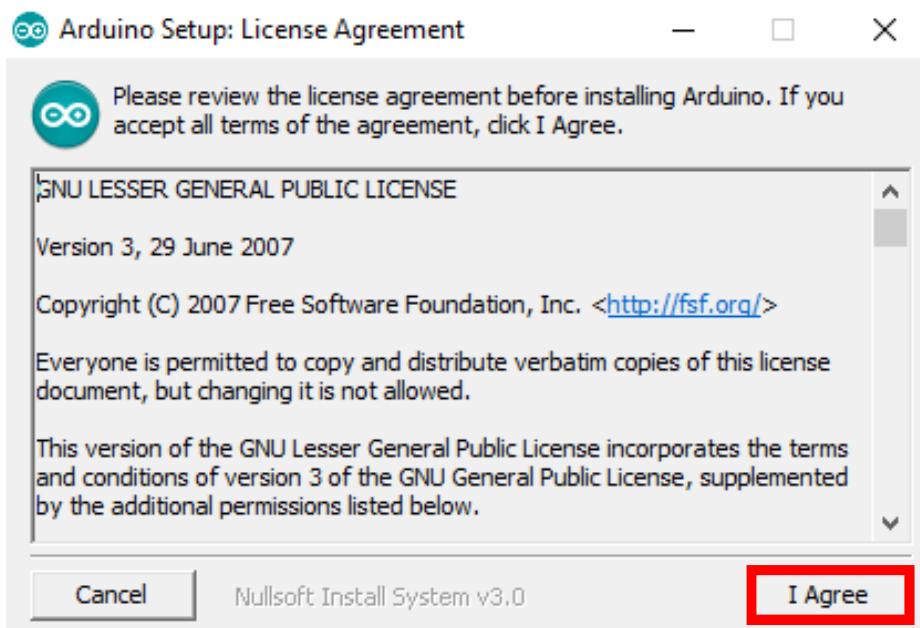
Aqui é onde vamos fazer o Download da **IDE do Arduino**. Selecione qual sistema operacional você utiliza.

Caso possua **Windows 7** ou uma **versão superior**, selecione a primeira opção: **Windows Win 7 and newer**.

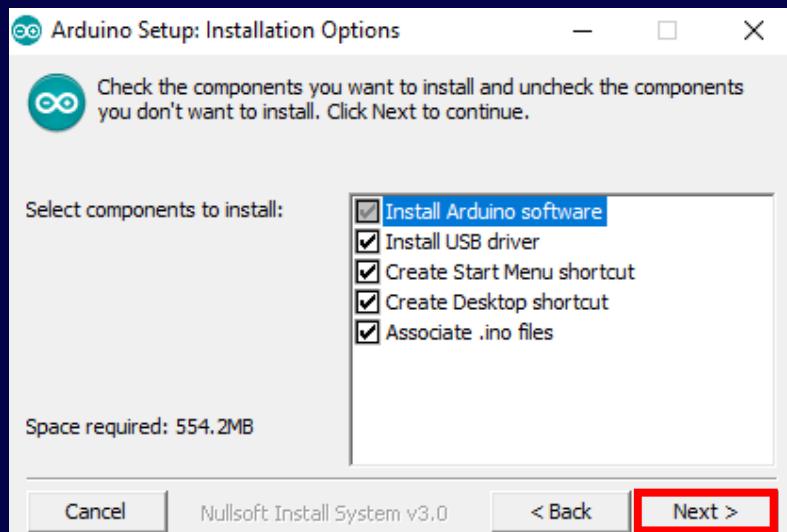
The screenshot shows the Arduino IDE download page. At the top, there's a navigation bar with links for VARE, SOFTWARE, CLOUD, DOCUMENTATION, COMMUNITY, and BLOG. Below the navigation bar, a section titled "Support the Arduino IDE" displays statistics: "Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **53.359.212** times — impressive! Help its development with a donation." Below this, there are several donation buttons with amounts: \$3, \$5, \$10, \$25, \$50, and Other. Two buttons are highlighted: "JUST DOWNLOAD" (in red) and "CONTRIBUTE & DOWNLOAD" (in green). Below the buttons is a small illustration of a computer monitor with a circuit board and two stylized characters.

Se quiser, você pode fazer uma doação para ajudar o desenvolvimento da **Arduino IDE**. Caso contrário, basta clicar em **Just Download** e o programa começará a baixar.

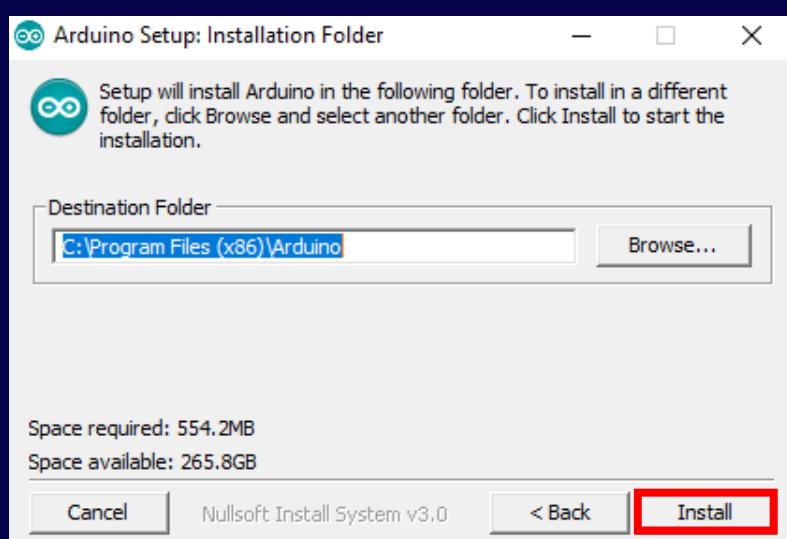
3 Após o Download do instalador da IDE terminar, **clique duas vezes sobre o ícone** gerado para abri-lo.



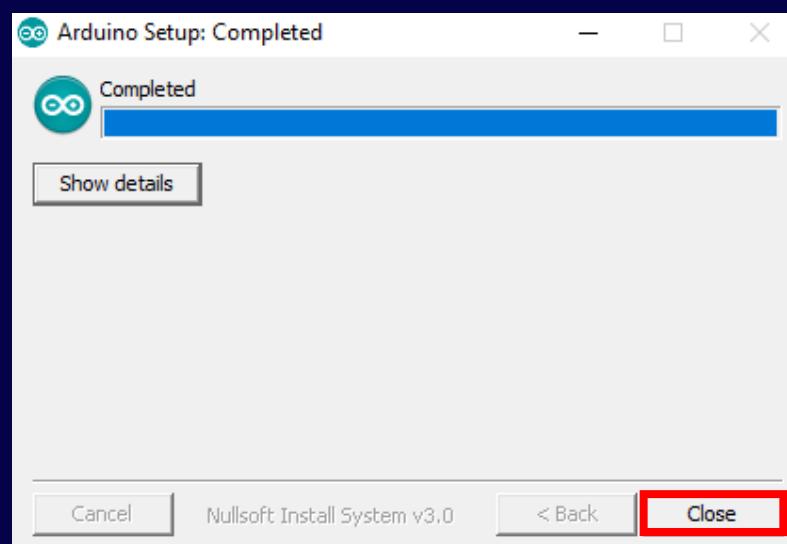
Clique em **I Agree**.



Selecione todas as caixas e clique em **Next**.



Escolha o local onde a IDE vai ser instalada em seu computador e clique em **Install**. A instalação começará.



Quando a instalação for concluída clique em **Close**.

Pronto, agora é só programar !

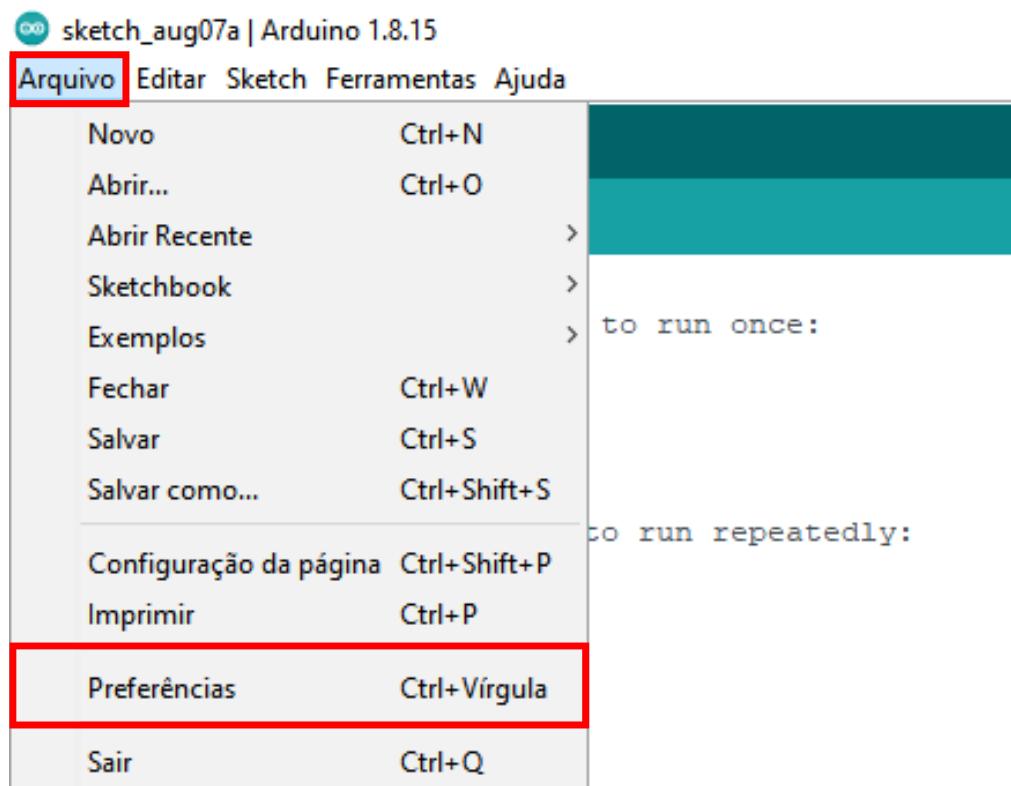
5

Configurando o ESP-32 na IDE

Precisamos configurar a **IDE do Arduino** para conseguirmos programar o **ESP-32**.

1

Abra a **IDE do Arduino** e clique em **Arquivo** e depois em **Preferências**.

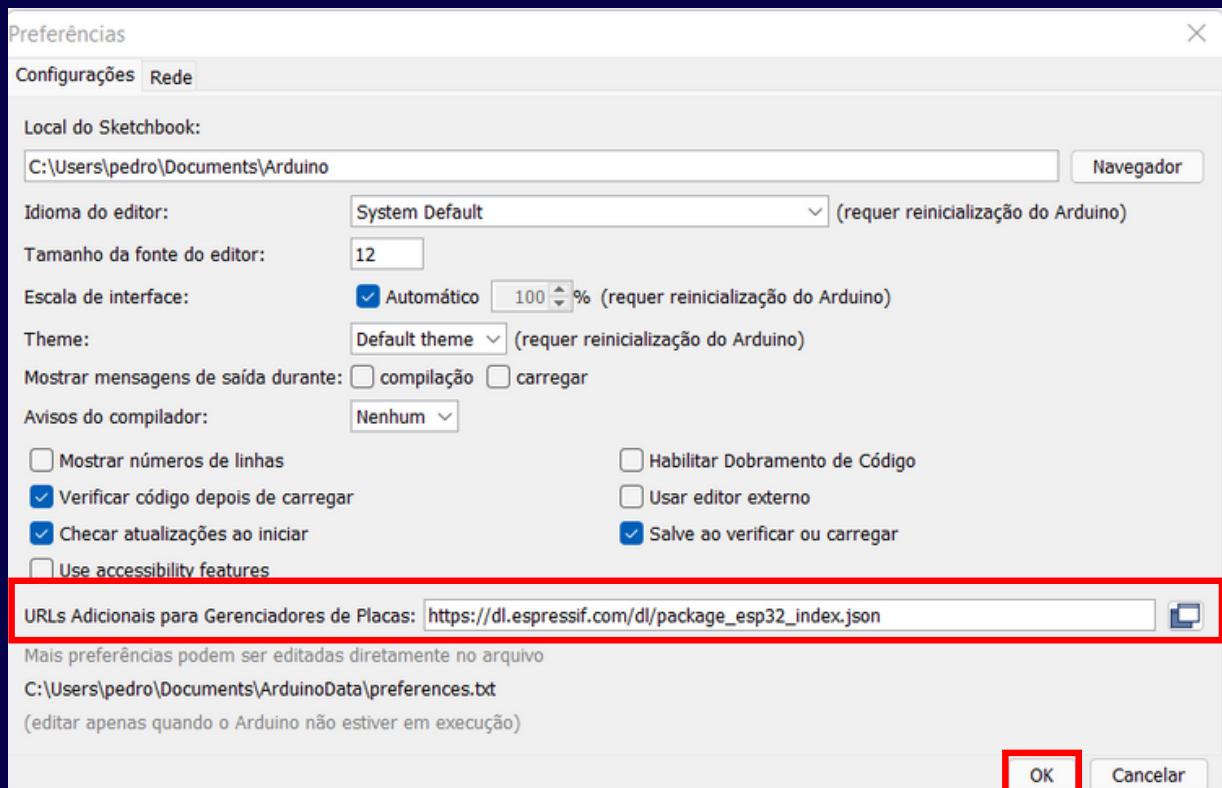


2

Copie o link:

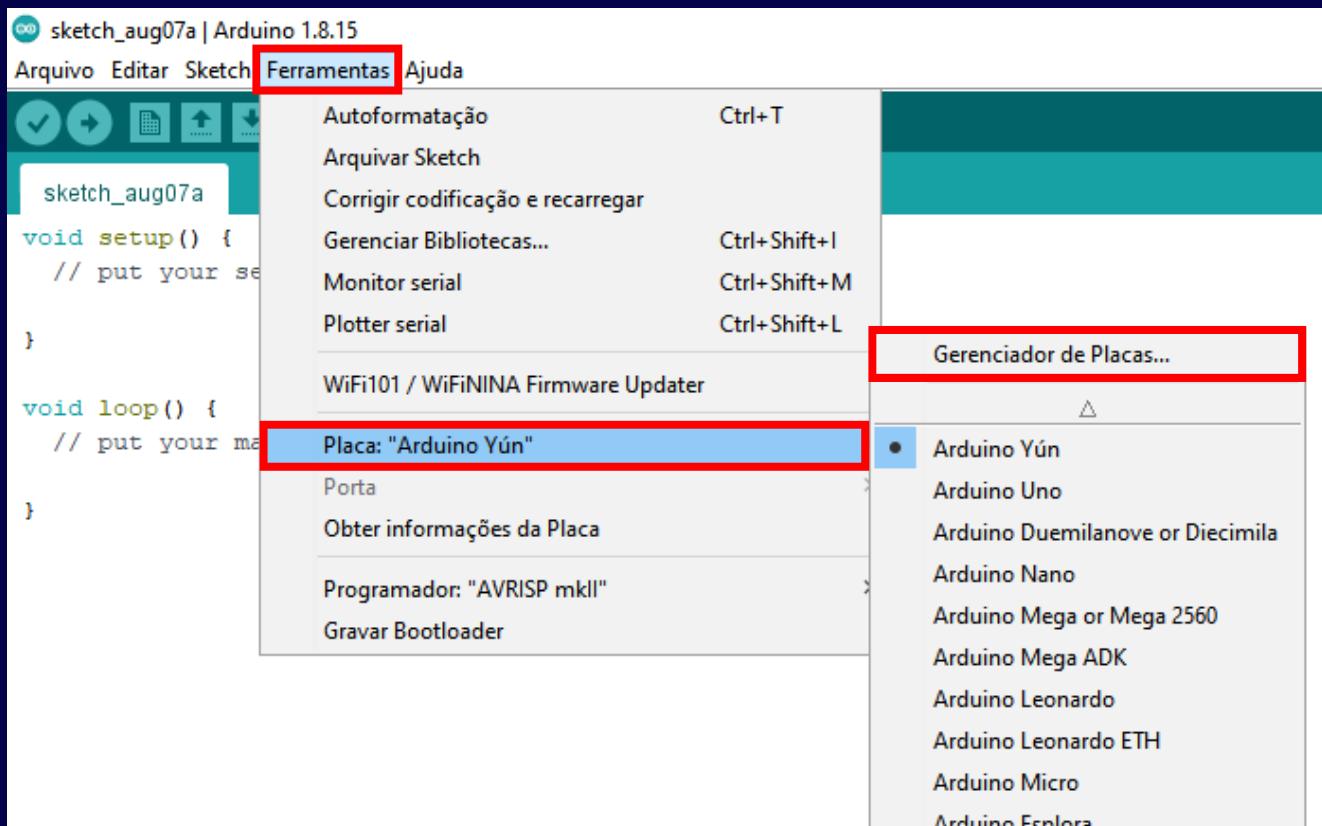
https://dl.espressif.com/dl/package_esp32_index.json

Cole no campo **URLs adicionais de Gerenciadores de Placas**, depois clique em **ok**.



3

Clique em **Ferramenta**, depois em **Placa** e em **Gerenciador de Placas**.



4

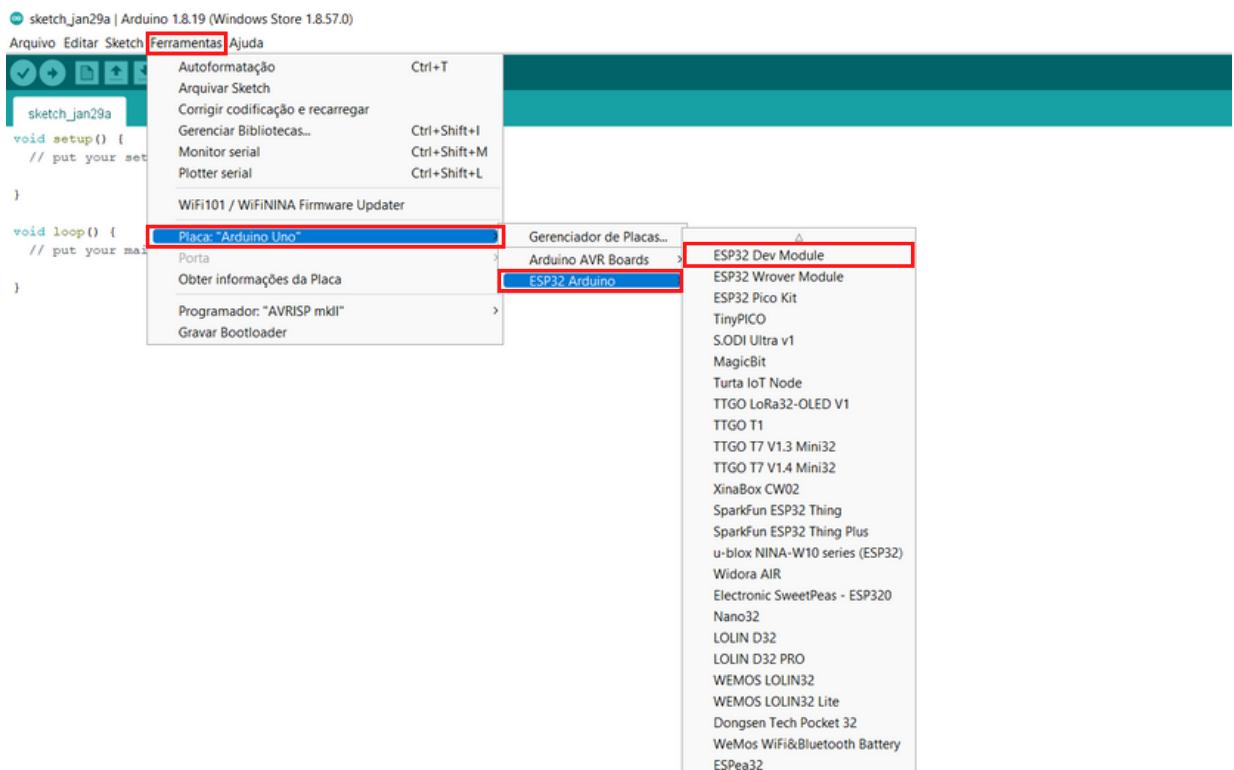
Busque na **barra de pesquisa** por **ESP32**.



Clique em **Instalar**.

5

Clique em **Ferramentas**, depois em **Placa**, selecione **ESP32 Arduino**, clique em **ESP32 Dev Module**.



6

Conhecendo o Blynk

Em muitos projetos nesta apostila vamos utilizar o aplicativo **Blynk**. O **Blynk** foi desenvolvido para ser utilizado em **projetos IoT**, com ele conseguimos comunicar através do celular com nossa placa **ESP-32** e controlá-la via **Wi-fi** ou **Bluetooth**.

É possível até mesmo **programarmos** nosso **microcontrolador** sem escrever uma linha de códigos se quer !!!



[Clique Aqui](#) para baixar o aplicativo para **Android** !!

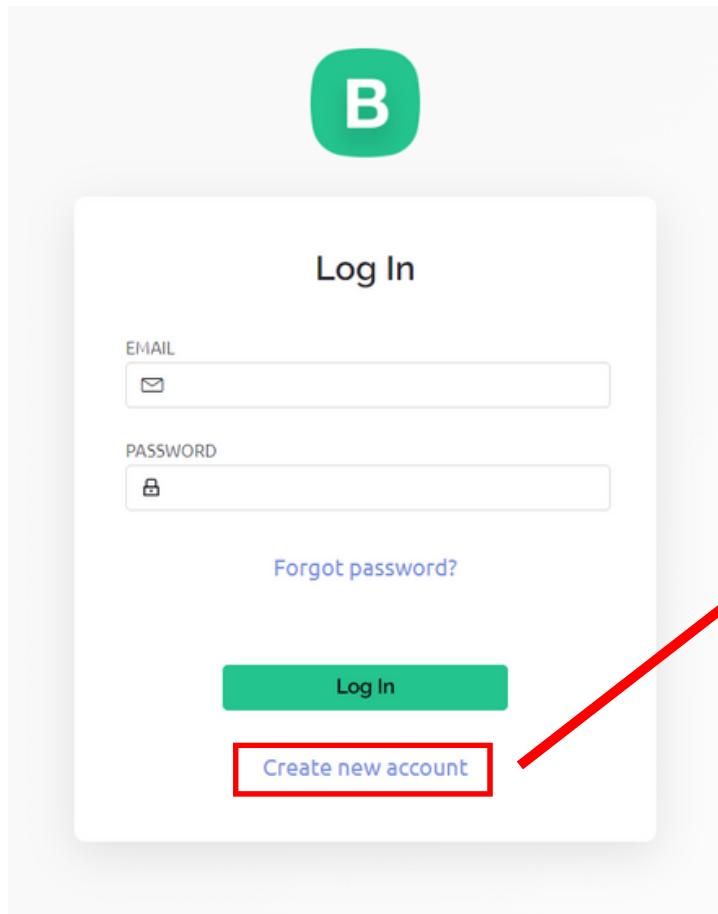
Caso utilize **IOS**, [Clique Aqui](#) para baixar.

Criando uma Conta

Para utilizar o **Blynk**, é preciso criar uma **conta**. É possível cria-la pelo site da **Blynk**, [clique aqui](#) para acessar.

OBS: É muito importante inserir um **e-mail** que você tem acesso !!

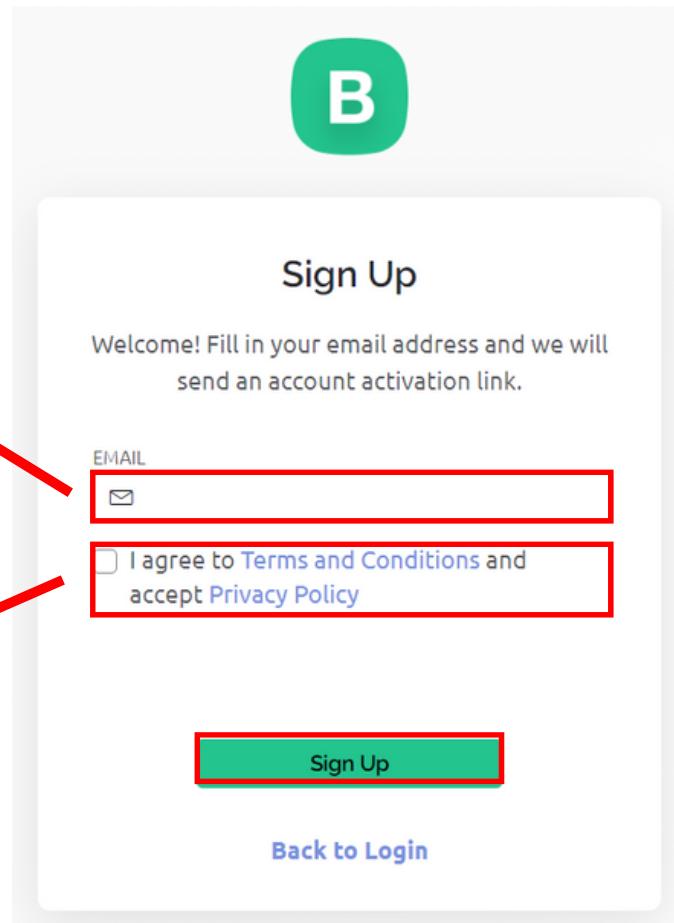




Clique em **Create new account**.

Você deve inserir seu e-mail, o **Blynk** enviara uma mensagem nesse **e-mail** para confirmar que ele é realmente seu. Logo após a confirmação, será necessário criar um senha.

Não se esqueça de **aceitar os Termos** e **Condições** de uso do site !

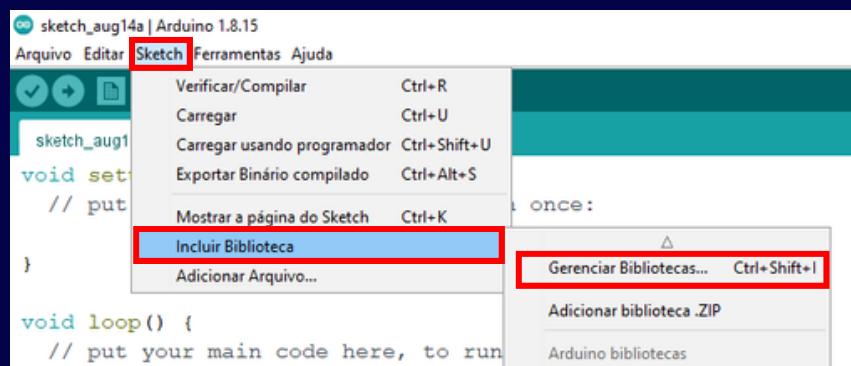


Configurando o Blynk na IDE

É necessário instalar a **biblioteca** do **Blynk** na IDE do Arduino para as programações funcionarem **corretamente**.

1

Para instalar a biblioteca clique em **Sketch**, depois em **Incluir Biblioteca** e **Gerenciar Bibliotecas**.



2

Na barra de pesquisa busque por **Blynk**, irá aparecer um botão para instalar a **biblioteca**, o botão só é habilitado após a escolha da versão da biblioteca, selecione a versão feita por Volodymyt Shymankyy e **clique no botão instalar**.



OBS: Instale a biblioteca feita por **Volodymyt Shymankyy** e **seleccione a versão mais recente**.

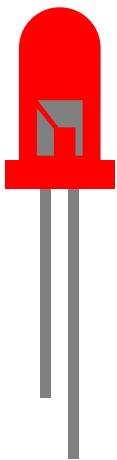
7

Sequencial de LEDs

Finalmente chegamos em nosso **primeiro projeto** !!! Nós vamos aprender a controlar o acendimento de três **LEDs** utilizando o aplicativo **Blynk** através do Bluetooth do smartphone. Mas antes é preciso conhecer um pouco mais sobre os componentes que serão utilizados no projeto.

LED

O **LED** é um diodo emissor de luz, ele é um componente capaz de emitir luz visível transformando a energia elétrica em energia luminosa.

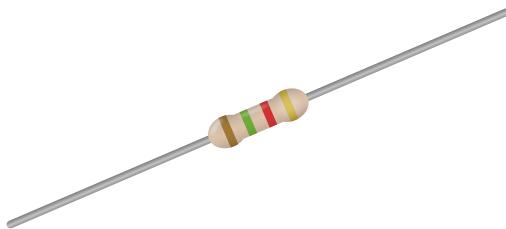


O maior terminal do LED é a parte **Positiva** e o menor **Negativa** !!

- **Ânodo:** Positivo
- **Cátodo:** Negativo

Resistor

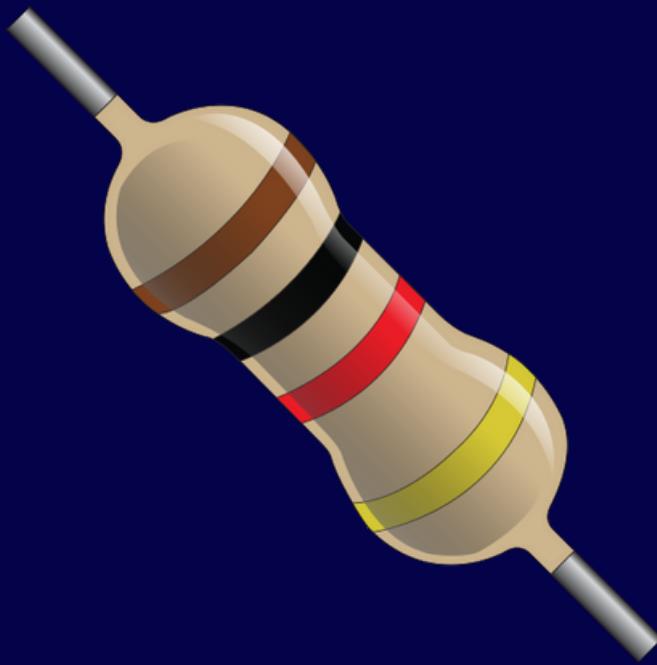
Resistores são dispositivos muito utilizados para controlar a passagem de corrente elétrica em circuitos elétricos por meio do **Efeito Joule**, que converte **energia elétrica** em **energia térmica**.



Vamos utilizar ele pois se ligarmos um **LED** direto no **5 volts** fornecido pelo Arduino, ele irá **queimar** !!

Código de Cores

Você deve ter notado que os **resistores** possuem algumas **faixas coloridas** em seu corpo, essas listras são **códigos** que **mostram o valor** do resistor, vamos aprender a como ler esses códigos !!



- 1-** As faixas coloridas são lidas a partir da que está mais próxima de uma extremidade.
- 2-** A primeira faixa colorida representa o **primeiro algarismo** do valor da resistência.
- 3-** A segunda faixa colorida indica o **segundo algarismo**.
- 4-** A terceira faixa representa a **potência de dez** pela qual devemos multiplicar os dois algarismos.
- 5-** A quarta faixa, que é opcional, indica **imprecisão** no valor da resistência. Prateado indica **10%** de imprecisão, dourado indica **5%** e sem nenhuma faixa representa imprecisão de **20%**.

Valor das Cores

COR	VALOR
Preto	0
Marrom	1
Vermelho	2
Laranja	3
Amarelo	4
Verde	5
Azul	6
Violeta	7
Cinza	8
Branco	9
Dourado	5% impre.
Prateado	10% impre.

Como saber qual a resistência correta ??

Existe uma **fórmula** para saber qual a **resistência** deve ser utilizada em um **LED**. Mas antes é preciso saber qual é a **Tensão** e a **Corrente** de cada **LED**.

LEDs		
Cor do LED	Tensão em Volts (V)	Corrente em Miliamperes (mA)
Vermelho	1,8 - 2,0V	20 mA
Amarelo	1,8 - 2,0V	20 mA
Laranja	1,8 - 2,0V	20 mA
Verde	2,0 - 2,5V	20 mA
Azul	2,5 - 3,0V	20 mA
Branco	2,5 - 3,0V	20 mA

Cálculos

Para se calcular o **resistor** adequado para o **LED** utilizaremos a seguinte fórmula:

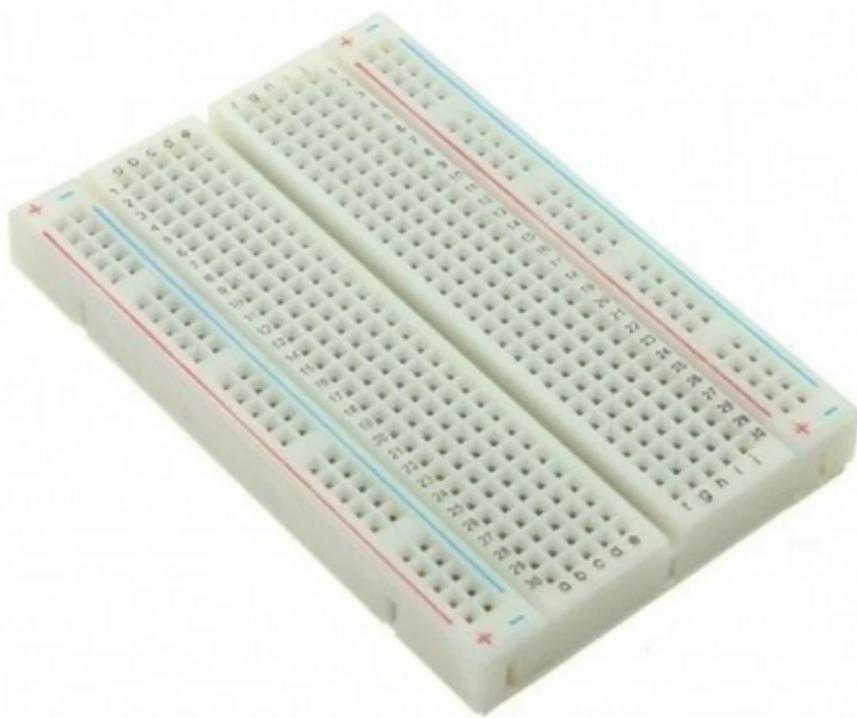
$$\text{Resistência do LED} = \frac{\text{Tensão de Alimentação} - \text{Tensão do LED}}{\text{Corrente do LED}}$$

OBS: O valor do resistor pode ser acima do resultado, porém nunca abaixo, por exemplo, se o valor for igual a **150 Ω**, podemos usar um resistor de valor mais alto, como **200 Ω**, ou até mesmo **1KΩ**, que são **1000 Ω**.

Lembre-se: O símbolo **Ω** significa **Ohms**, que é a nossa unidade de resistência elétrica.

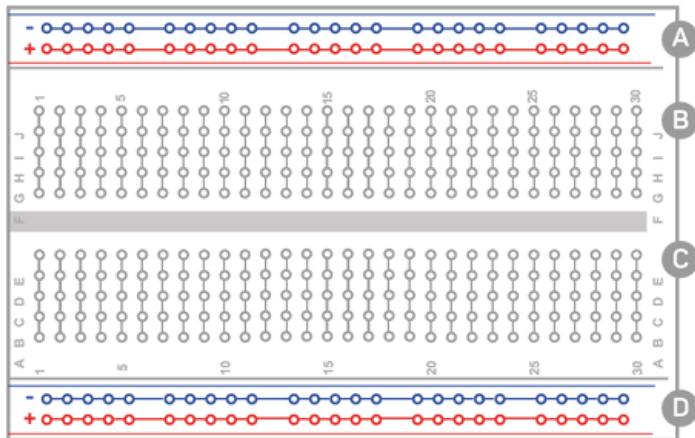
Protoboard

O próximo componente é a **Protoboard**, que nada mais é do que uma placa com furos e conexões condutoras utilizada para a montagem de protótipos e projetos que estão na fase inicial. A vantagem de se utilizar uma **Protoboard** é que podemos montar os componentes direto nele, assim eliminamos a necessidade de utilizar a solda.



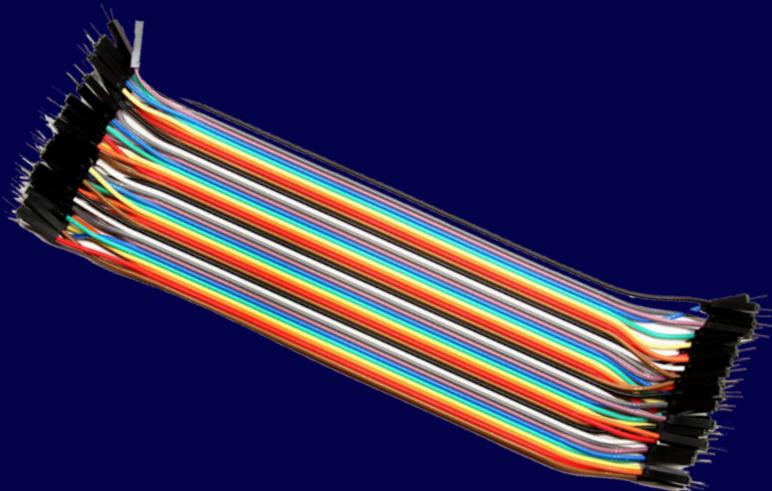
Conexões

As conexões da **Protoboard** são dessa forma por dentro:



Jumper

Os **Jumpers** nada mais são do que condutores para ligar um ponto a outro.



Unindo Protoboards

Nossa placa não cabe na **protoboard**, dessa forma será necessário juntarmos duas protoboard. Caso haja dúvidas de como isso deve ser feito, siga o tutorial abaixo:

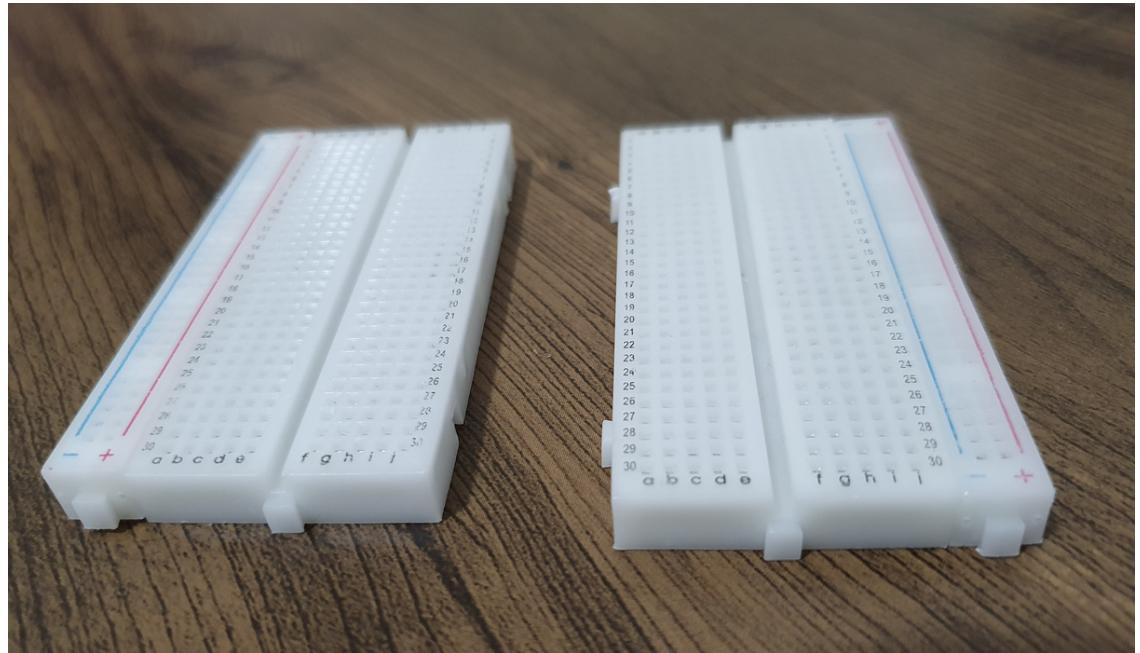


Primeiro é necessário remover os **adesivos** da parte traseira das protoboards.

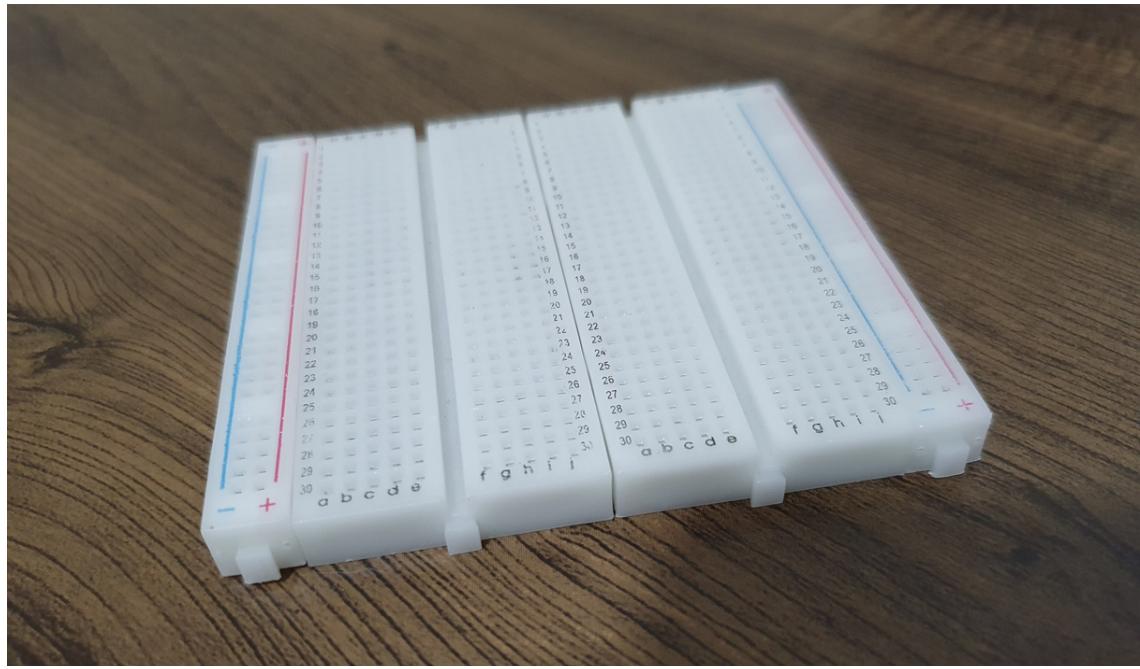




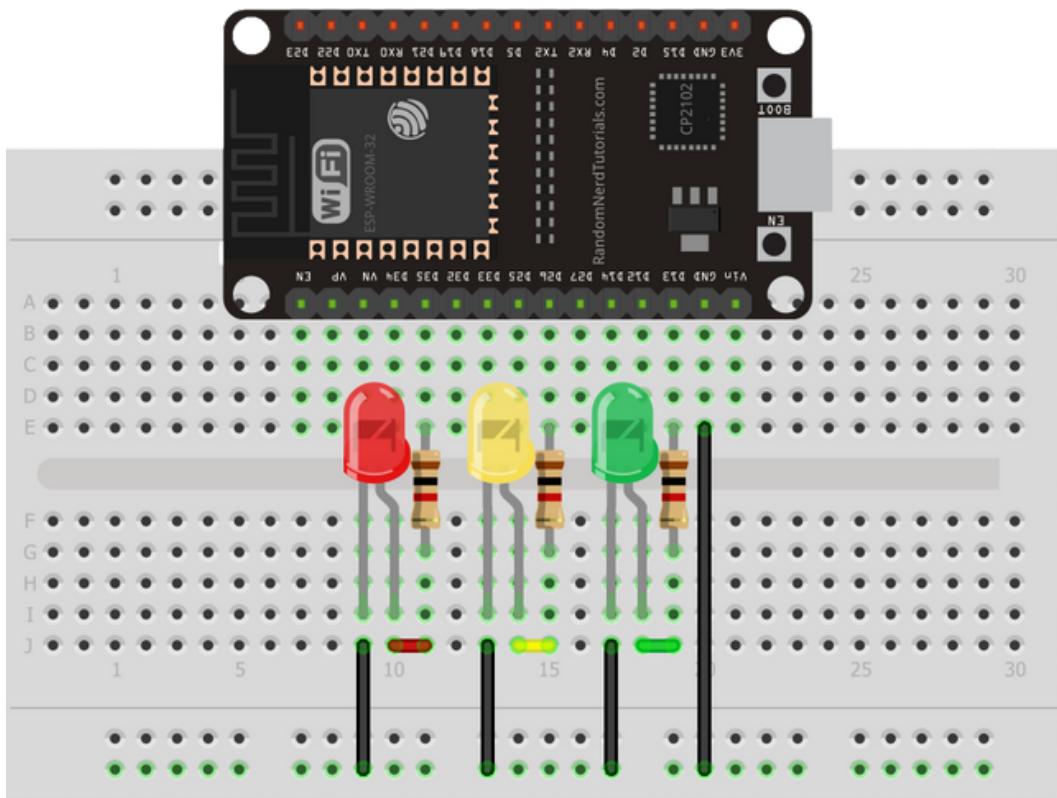
Depois, é necessário remover um dos lados para alimentação (**positivo e negativo**) em ambas as **protoboard**.



Por fim, é necessário unir as duas **protoboard**.



Círcuito



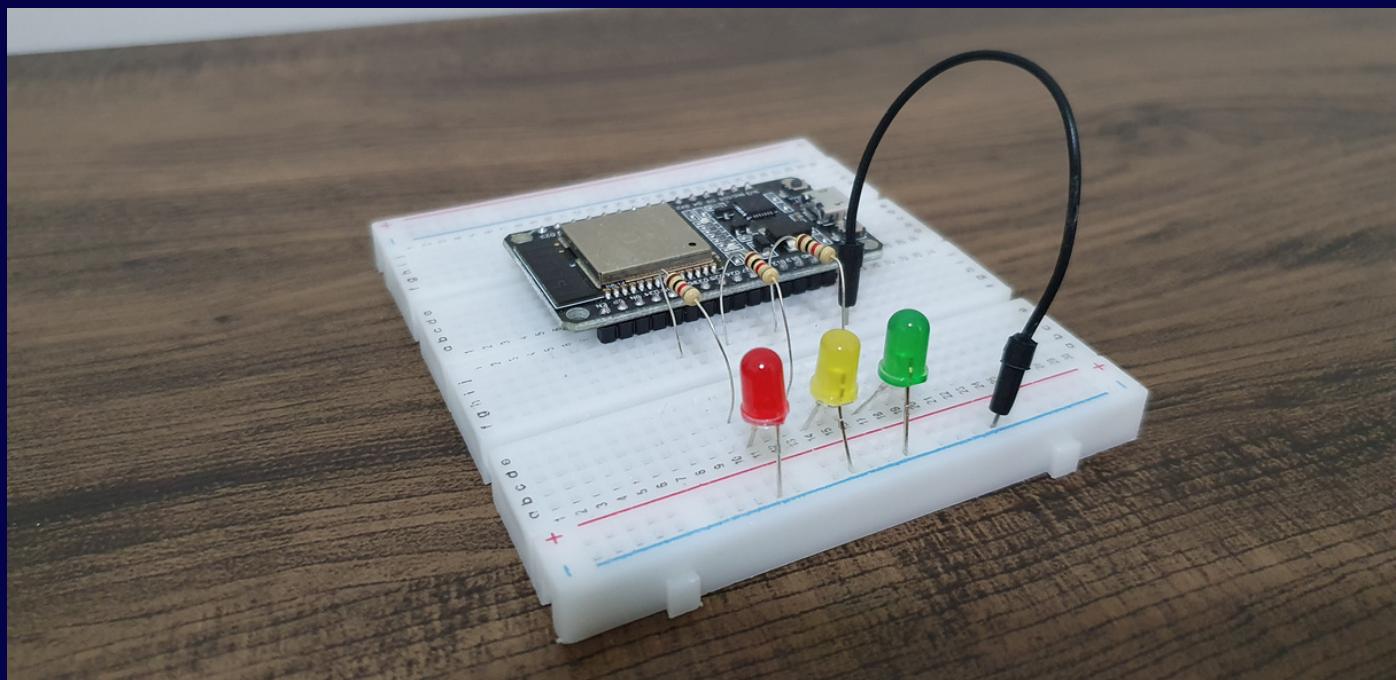
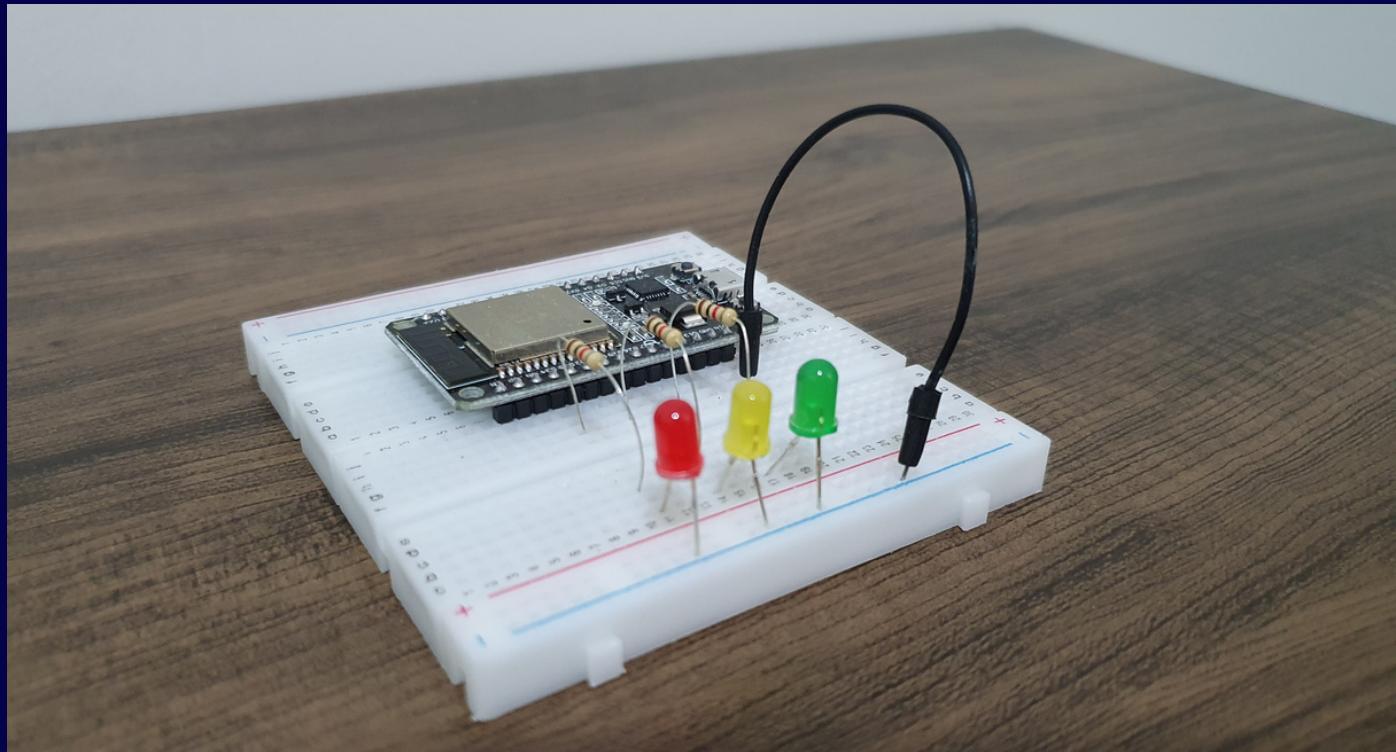
Primeiro é necessário energizar a protoboard com o **GND** da nossa placa.

O maior terminal do **LED vermelho** deve ser conectado a um **resistor de $1\text{K}\Omega$** e na **porta digital 32**. O menor terminal deve ser conectado no **negativo** do circuito.

O maior terminal do **LED amarelo** deve ser conectado a um **resistor de $1\text{K}\Omega$** e na **porta digital 26** (**obs:** a numeração da placa pode estar diferente do esquema acima). O menor terminal deve ser conectado no **negativo** do circuito.

O maior terminal do **LED verde** deve ser conectado a um **resistor de $1\text{K}\Omega$** e na **porta digital 13**. O menor terminal deve ser conectado no **negativo** do circuito.

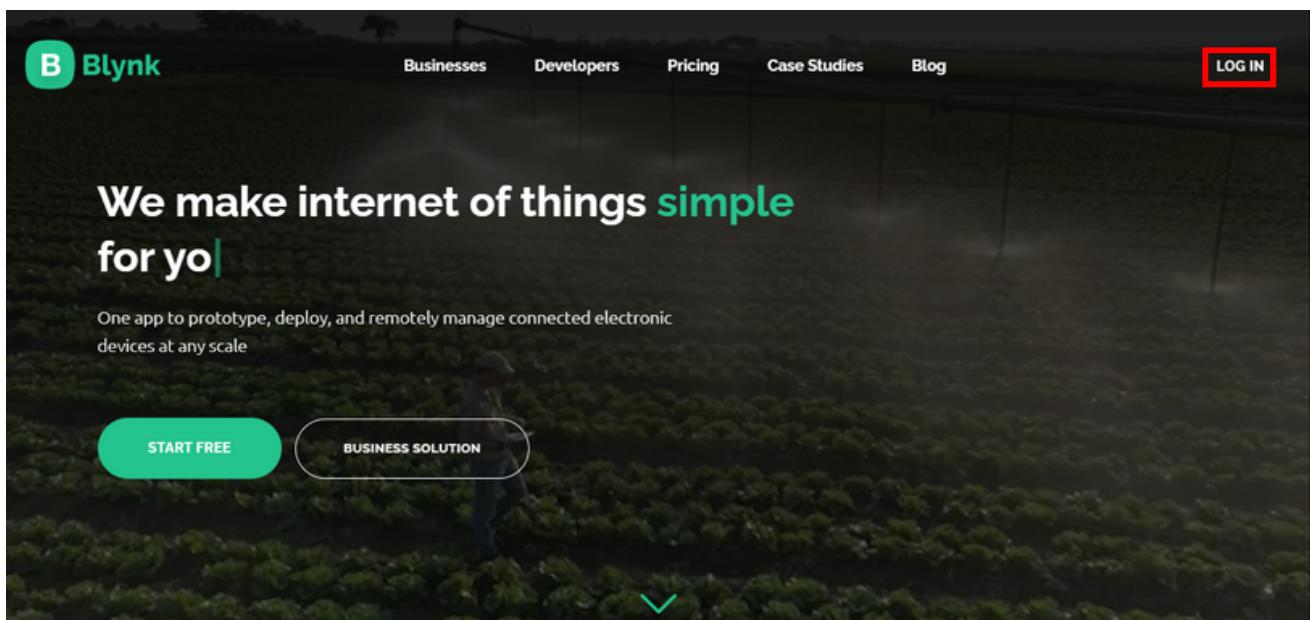
Círcuito na Prática



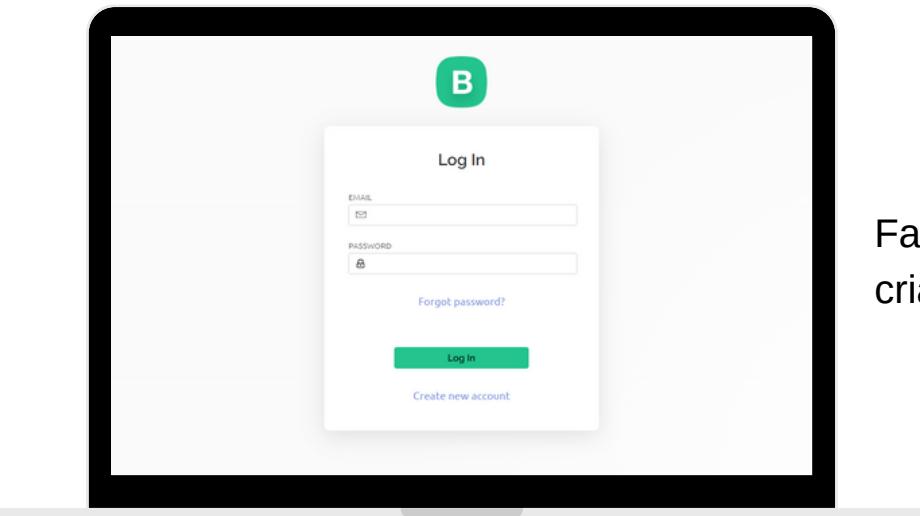
Configurando o Blynk



Vamos acessar o site do [Blynk](#) e acessar nossa conta que criamos anteriormente. [Clique aqui](#) para acessar.



Clique sobre **LOG IN**.



Faça o login com o **e-mail** e a **senha** criada anteriormente.



Clique sobre **New Templates**.

The screenshot shows the Blynk platform's main dashboard. On the left is a sidebar with icons for Devices, Locations, and Users. The main area has a heading "Start by creating your first template" and a sub-instruction: "Template is a digital model of a physical object. It is used in Blynk platform as a template to be assigned to devices." A red box highlights the "+ New Template" button at the bottom right of this section. The bottom right corner of the screen displays the text "Region: ny3 Privacy Policy".



Crie um novo **Template**.

Create New Template

NAME
Sequencial de LEDs

HARDWARE
ESP32

CONNECTION TYPE
WiFi

DESCRIPTION
This is my template

19 / 128

Cancel Done

Adicione um nome para o **Template**.

Em **Connection Type** selecione **WiFi**.

Em **Hardware** selecione **ESP32**.

Por fim, clique em **Done**.

4

Clique sobre **Datastreams**.

B

Temperatura e Umidade

Info Metadata **Datastreams** Events Automations Web Dashboard Mobile Dashboard

TEMPLATE NAME
Temperatura e Umidade

HARDWARE
ESP8266 CONNECTION TYPE
WiFi

DESCRIPTION
This is my template

TEMPLATE ID
TMPLugnop2Bp MANUFACTURER
My organization 9402DL

OFFLINE IGNORE PERIOD
00 hrs 00 mins 00 secs

HOTSPOT PREFIX
Hotspot Prefix

TEMPLATE IMAGE (OPTIONAL)

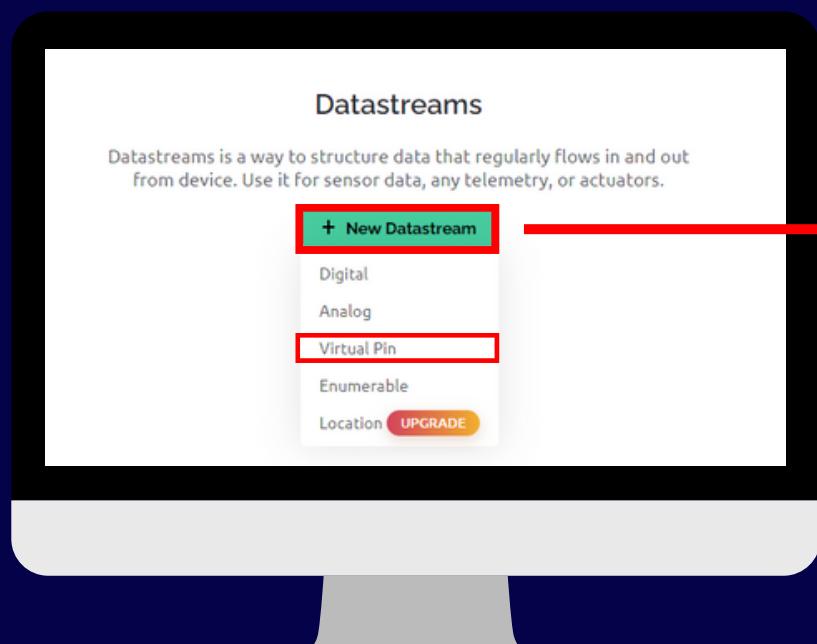
Add image
Upload from computer or drag-n-drop .png or .jpg, minimum width 500px

FIRMWARE CONFIGURATION

```
#define BLYNK_TEMPLATE_ID "TMPLugnop2Bp"  
#define BLYNK_DEVICE_NAME "Temperatura e Umidade"
```

Template ID and Device Name should be included at the top of your main firmware

Save



Clique sobre **New Datasteam** e selecione a opção **Virtual Pin** (pinos virtuais).

Configurando os Pinos Virtuais

Vamos precisar de três **pinos virtuais**, um para cada **LED**. As configurações dos três vão ser iguais, basta alterar o **nome** e o **Pin** (caso não mude automaticamente).

Virtual Pin Datastream

NAME	LED Vermelho	ALIAS	LED Vermelho		
PIN	V0	DATA TYPE	Integer		
UNITS	None				
MIN	0	MAX	1	DEFAULT VALUE	0
[+] ADVANCED SETTINGS					
				Cancel	Create

Após finalizar a configuração, clique em **Create**.

O **LED Vermelho** deve estar no pino **V0**.

O **LED Amarelo** deve estar no pino **V1**.

O **LED Verde** deve estar no pino **V2**.

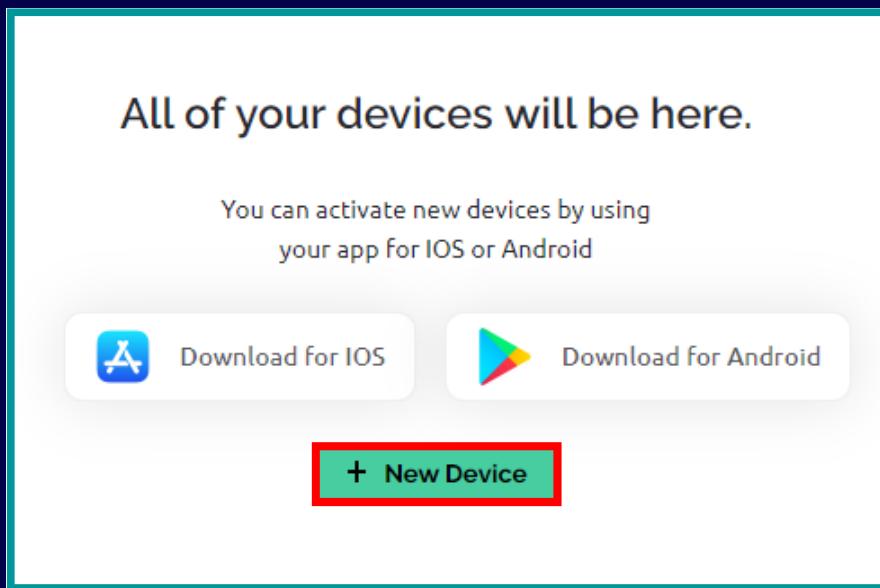
	Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Actions
1	1	LED Vermelho	LED Vermelho	Red	V0	Integer		False	0	
2	2	LED Amarelo	LED Amarelo	Yellow	V1	Integer		False	0	
3	3	LED Verde	LED Verde	Green	V2	Integer		False	0	

Após finalizar as configurações dos pinos, clique em **Save** para salvar e após clique sobre a **Lupa** (Search).

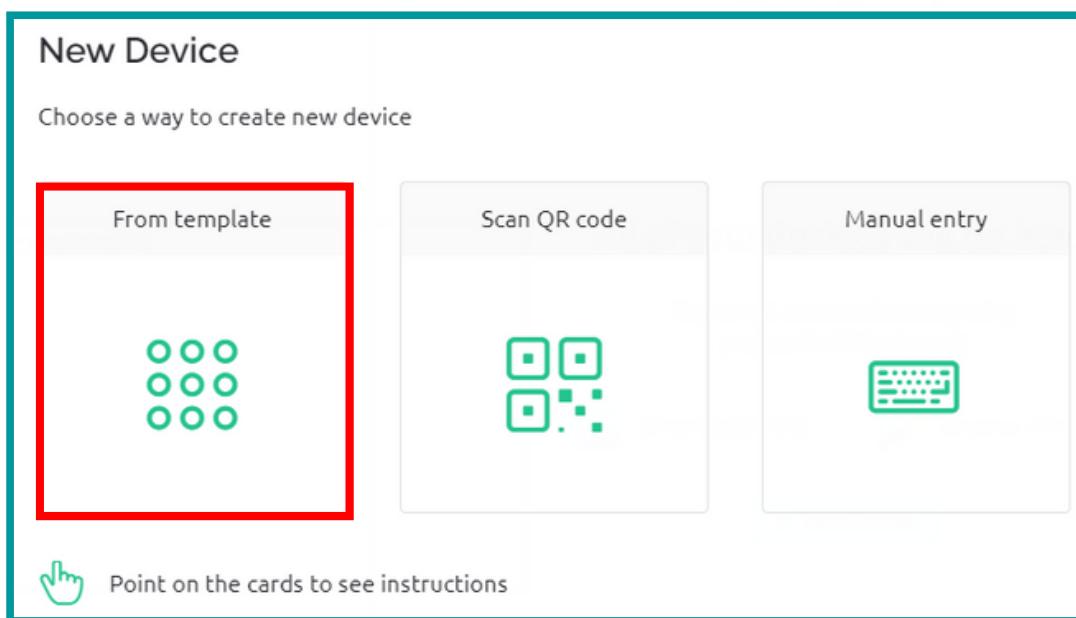
The screenshot shows a configuration interface for a "Sequential de LEDs" (Sequential LEDs) project. On the left, there is a sidebar with various icons: a green circle with a white letter 'B', a magnifying glass (highlighted with a red box), a gear, a circular arrow, a double arrow, a speaker, and a gear. The main area has a title "Sequential de LEDs" and tabs for "Info", "Metadata", "Datastreams" (which is selected and highlighted with a green underline), "Events", "Automations", "Web Dashboard", and "Mobile Dashboard". A "Save" button is located in the top right corner, also highlighted with a red box. Below the tabs is a search bar labeled "Search datastream" and a green button labeled "+ New Datastream". The main content area displays a table titled "3 Datastreams" with the following data:

	Id	Name	Alias	Color	Pin	Data Type	Units	Is Raw	Min	Actions
1	LED Vermelho	LED Vermelho		Red	V0	Integer		false	0	
2	LED Amarelo	LED Amarelo		Yellow	V1	Integer		false	0	
3	LED Verde	LED Verde		Green	V2	Integer		false	0	

Ao clicar sobre a lupa, abrirá uma nova janela. Clique sobre **New Device**.



Selecione **From Template**.



Em **Template**, escolha o criando anteriormente. Após, clique em **Create**.

New Device

Create new device by filling in the form below

TEMPLATE

Séquencial de LEDs

DEVICE NAME

Séquencial de LEDs

Cancel Create

Em **Device Info** será apresentado informações importantes na hora da programação !!

Sequencial de LEDs Offline

Device Info

Dashboard Timeline Metadata Actions Log

STATUS LAST UPDATED

● Offline 1:02 AM Today

DEVICE ACTIVATED ORGANIZATION

1:02 AM Today

AUTH TOKEN TEMPLATE NAME

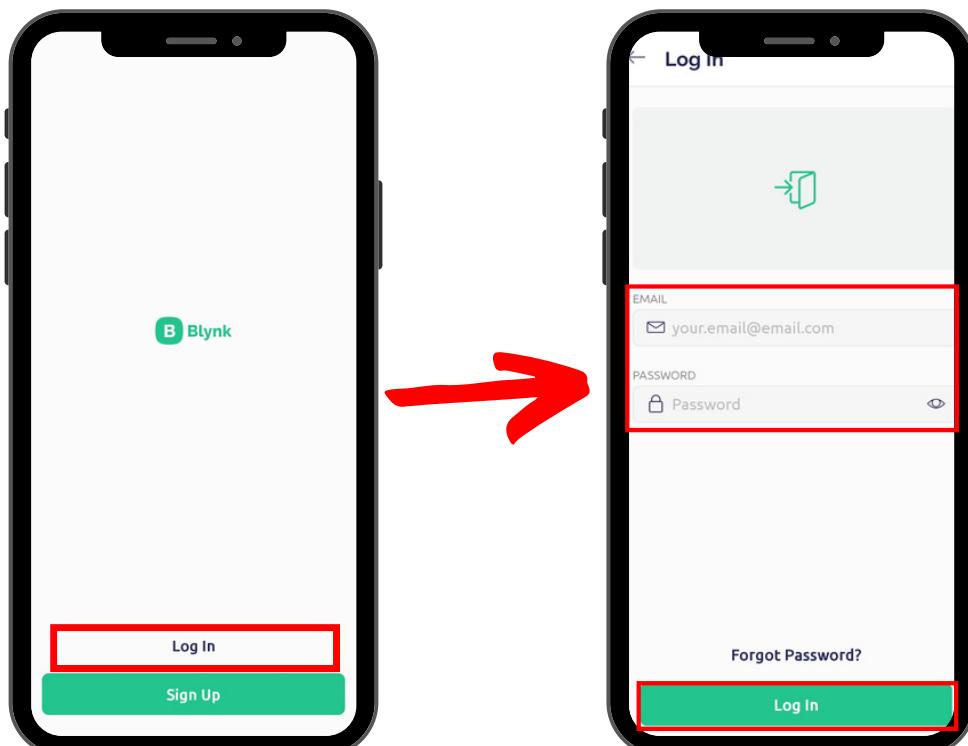
DKGH - **** - **** - **** Sequencial de LEDs

FIRMWARE CONFIGURATION

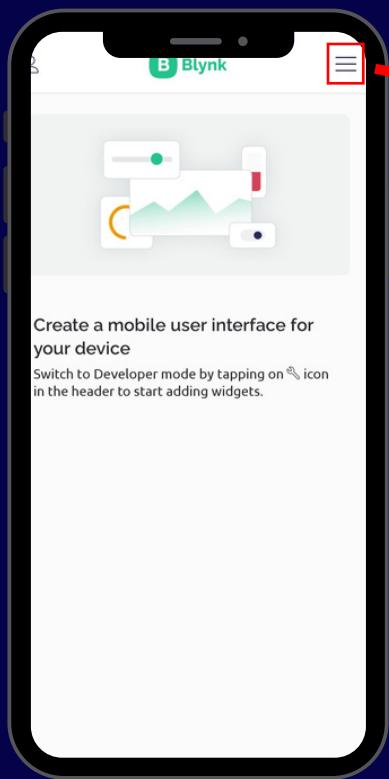
```
#define BLYNK_TEMPLATE_ID "TMI"
#define BLYNK_DEVICE_NAME "Sequencial de LEDs"
#define BLYNK_AUTH_TOKEN "DKGH-****-****-****"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

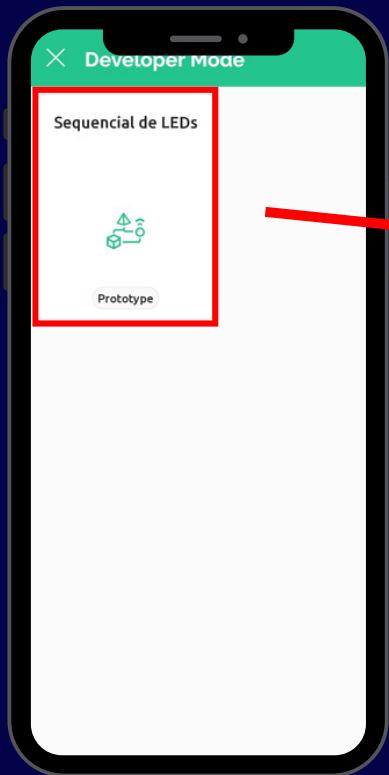
Configurando o Blynk no celular



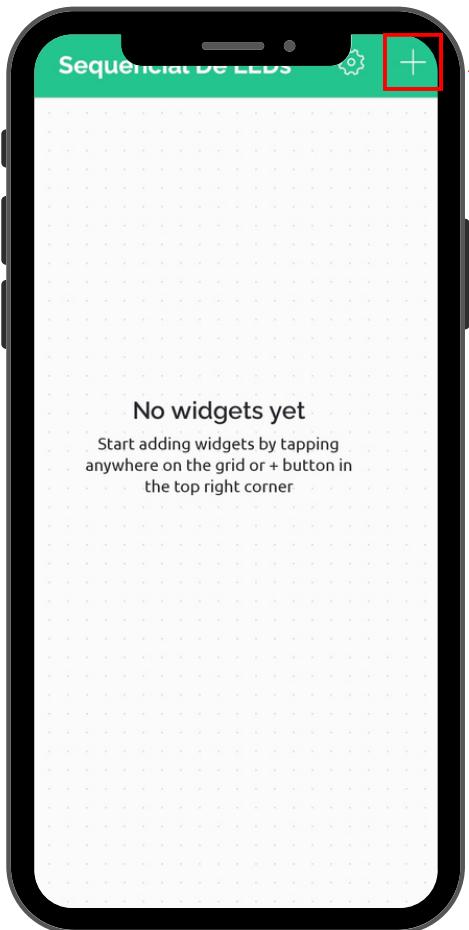
Abra o aplicativo baixado anteriormente, faça login com o mesmo **e-mail** e **senha** usado no site.



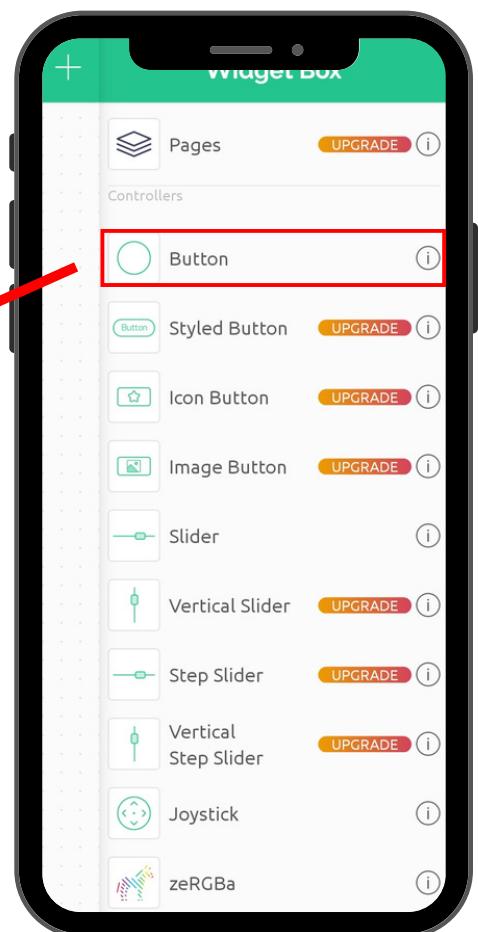
Na página inicial, clique sobre as **três barras** e depois selecione a opção **Developer Mode**.



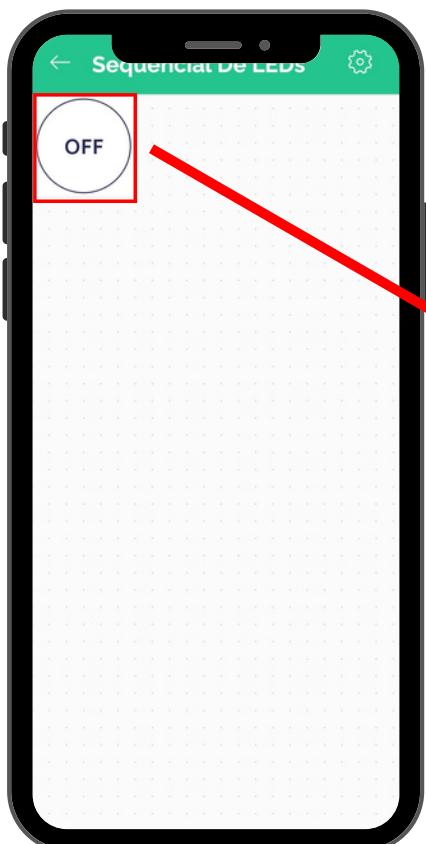
Selecione o **projeto** que criamos pelo **site**.



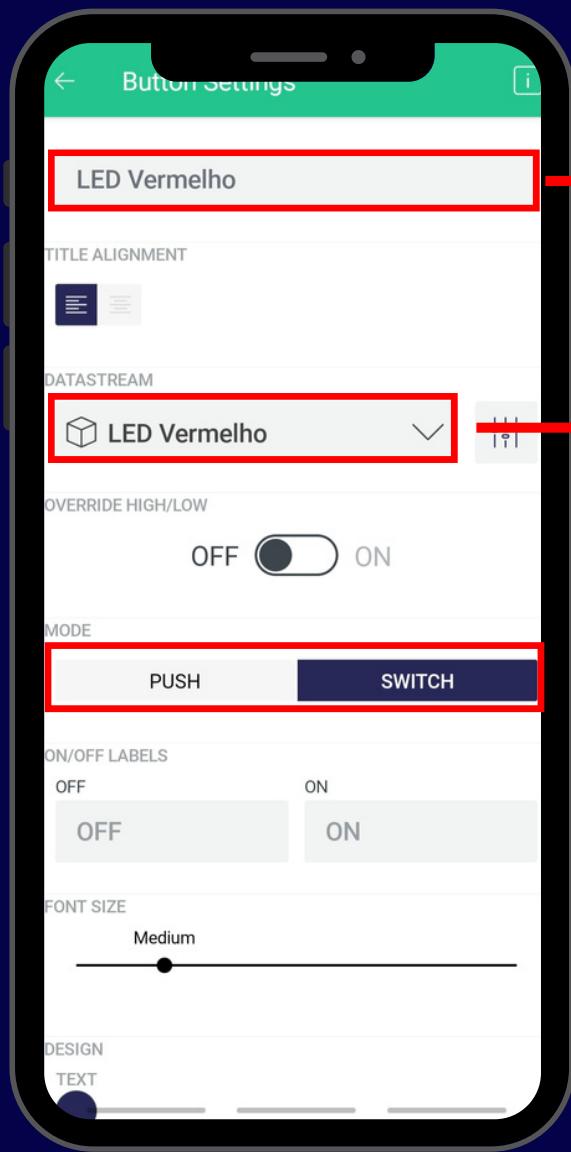
Clique sobre o **+** para adicionar os botões que farão com que os **LEDs** seja acessos.



Selezione **Button**,
vamos precisar de
três.



Clique sobre o **botão** para realizar as
configurações necessárias.



Coloque o nome do botão.

Selecione o Pino Virtual que vai acender o LED (já definimos ele pelo site.).

Selecione a opção Switch, dessa forma um toque liga e outro desliga o LED.

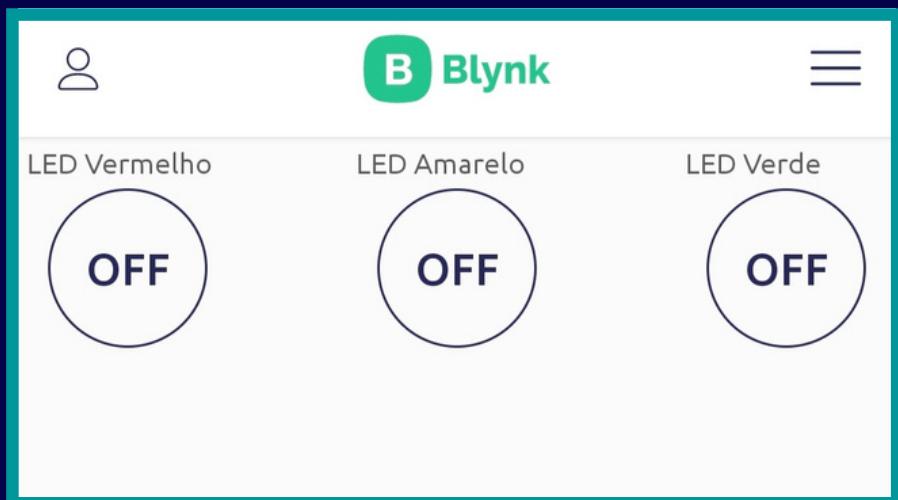
OBS: As configurações dos três botões são iguais, basta alterar o nome e o Pino Virtual. Lembre-se:

LED Vermelho: V0

LED Amarelo: V1

LED Verde: V2

Após finalizar a configuração dos botões, basta voltar para a tela inicial do App.



Programação

```
#define BLYNK_PRINT Serial
#define BLYNK_DEVICE_NAME "Nome do projeto"
#define BLYNK_TEMPLATE_ID "Insira o código do Template"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = "Insira o Token";
char ssid[] = "Insira o nome da rede";
char pass[] = "Insira a senha da rede";

int LEDVermelho = 32;
int LEDAmarelo = 26;
int LEDVerde = 13;

BLYNK_WRITE (V0){

    int valor = param.asInt();
    digitalWrite(LEDVermelho, valor);
}

BLYNK_WRITE (V1){

    int valor = param.asInt();
    digitalWrite(LEDAmarelo, valor);

}

BLYNK_WRITE (V2){
```

```

int valor = param.asInt();
digitalWrite(LEDVerde, valor);
}

void setup()
{
Serial.begin (115200);

Blynk.begin(auth, ssid, pass);

pinMode(LEDVermelho, OUTPUT);
pinMode(LEDAmarelo, OUTPUT);
pinMode(LEDVerde, OUTPUT);

}

void loop()
{
Blynk.run();
}

```

Atenção !!

```

Carregando...
Variáveis globais usam 32588 bytes (9%) de memória dinâmica, deixando 295092 bytes para variáveis locais. O máximo são 327680 bytes.
esptool.py v3.0-dev
Serial port COM3
Connecting.....
8-8
ESP32 Dev Module. Disabled. Default 4MB with splits (1.2MB APP/1.5MB SPIFFS), 240MHz (WIFI/BT), QIO, 80MHz, 4MB (2.2Mb), 921600, None em COM

```

Quando aparecer a mensagem "connecting" na IDE, pressione por alguns segundos a tecla **BOOT** do **ESP-32**.

Baixe o código do projeto [Clicando Aqui!](#)

Quer ver o funcionamento ?? [Clique aqui!!](#)



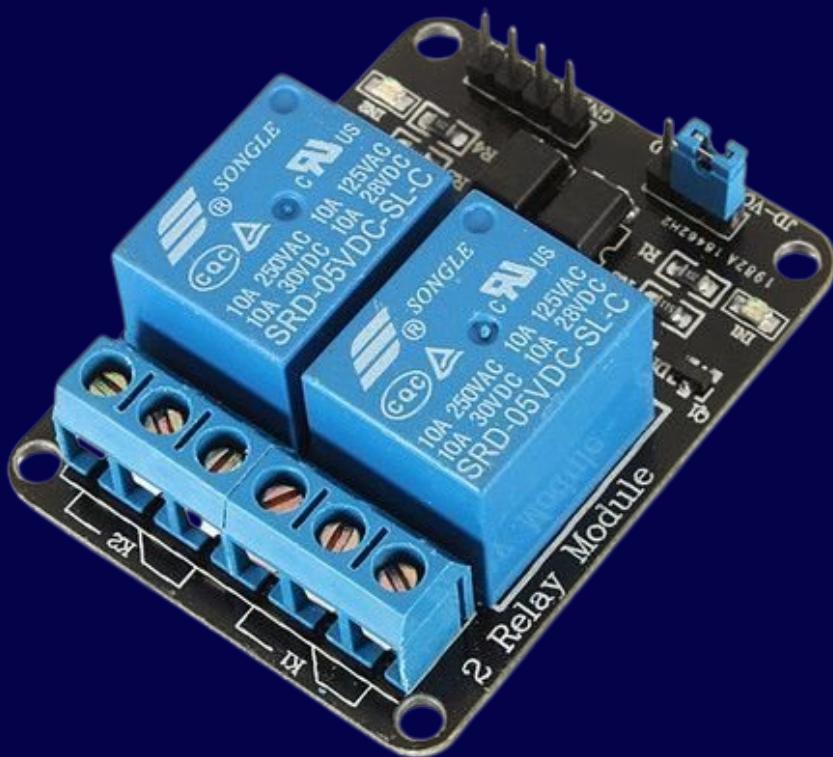
8

Ligando Lâmpada e Ventilador

Vamos utilizar o **Relé** e o **Blynk** para acender e apagar lâmpadas e ligar um ventilador, mas antes, vamos conhecer melhor sobre o **Módulo Relé**.

Módulo Relé

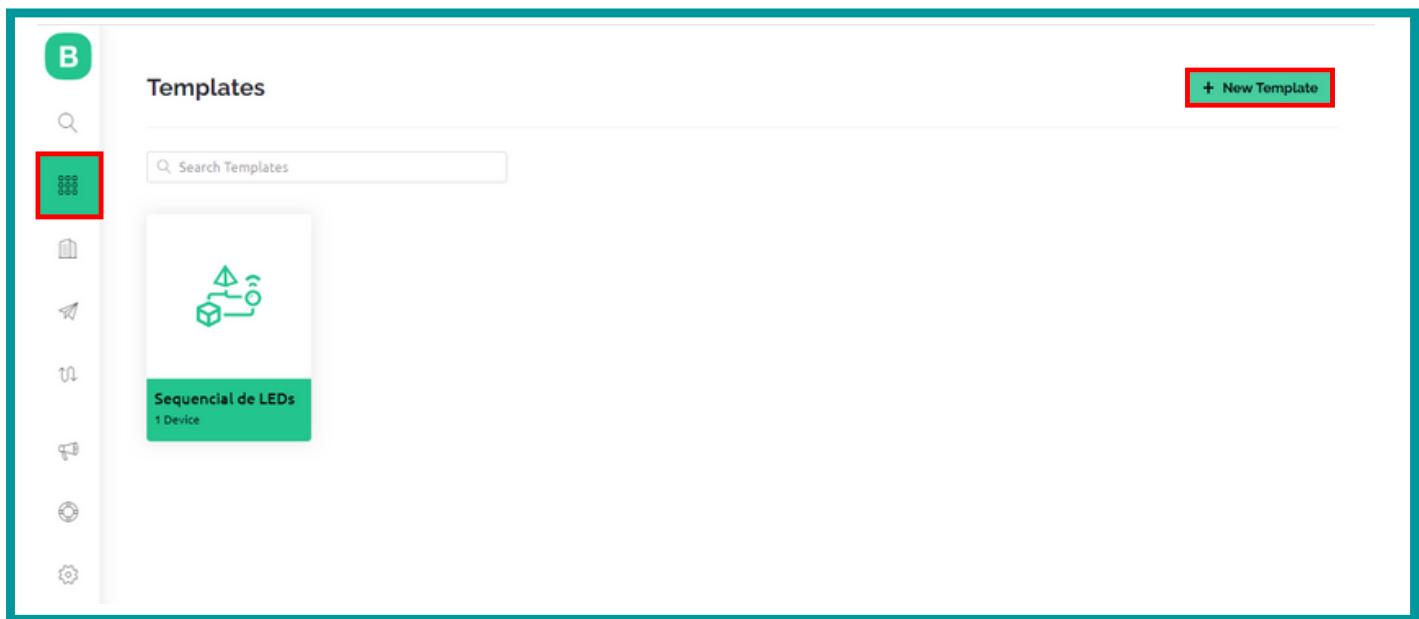
Este **Módulo Relé** permite uma integração com uma ampla gama de microcontroladores como o **Arduino**. A partir das **saídas digitais** pode-se, através do **relé**, **controlar cargas maiores e dispositivos** como motores **AC** ou **DC**, **eletroímãs, solenóides e lâmpadas incandescentes**.



Este módulo tem **dois canais** sendo assim concebido para ser integrado para controlar até **2 relés**. O módulo é equipado com um relé de **alta qualidade**, com carga nominal **10A/250VAC, 10A/125VAC, 10A/30VDC**. Cada canal possui um **LED** para indicar o estado da saída do relé.

Configurando o Blynk

Acesse o site blynk.io, clique sobre **Templates** e depois em **New Template**.



Nas configurações do novo **template**, adicione um **nome**, em **Hardware** selecione **ESP32** e em **Connection Type** escolha **WiFi**. Após finalizar clique em **Done**.

A screenshot of the 'Create New Template' dialog box. It has fields for 'NAME' (containing 'Ventilador e Lâmpada'), 'HARDWARE' (set to 'ESP32'), 'CONNECTION TYPE' (set to 'WiFi'), and a 'DESCRIPTION' text area (containing 'This is my template'). At the bottom are 'Cancel' and 'Done' buttons, with 'Done' highlighted with a red box.

Para iniciar a **configuração** dos botões, clique sobre **Datastreams**, depois em **New Datastreams** e selecione a opção **Virtual Pin**.

The screenshot shows the 'Ventilador e Lâmpada' configuration page. The 'Datastreams' tab is selected, highlighted with a red box. A modal window titled 'Datastreams' is open, explaining its purpose: 'Datastreams is a way to structure data that regularly flows in and out from device. Use it for sensor data, any telemetry, or actuators.' It contains a button '+ New Datastream' and a list of stream types: Digital, Analog, Virtual Pin (which is also highlighted with a red box), Enumerable, and Location. An 'UPGRADE' button is visible at the bottom right of the modal.

Configurando os Botões

- Vamos iniciar realizando as **configurações** do **botão** que vai acionar o **ventilador**.

The screenshot shows the 'Virtual Pin Datastream' configuration dialog. A red arrow points to the 'NAME' field where 'Ventilador' is typed. Another red arrow points to the 'PIN' dropdown menu where 'V0' is selected. A third red arrow points to the 'Create' button at the bottom right. The dialog includes fields for 'ALIAS', 'DATA TYPE' (set to Integer), 'UNITS' (None), 'MIN' (0), 'MAX' (1), and 'DEFAULT VALUE' (0). There is also an 'ADVANCED SETTINGS' button.

Adicione o nome

Selezione o pino V0

Após finalizar clique em **Create**.

- Agora vamos **configurar** o **botão** que acenderá a **lâmpada**. Em **Name** escolha um nome e em **Pin** selecione **V1**. Após terminar a configuração clique em **Create**.

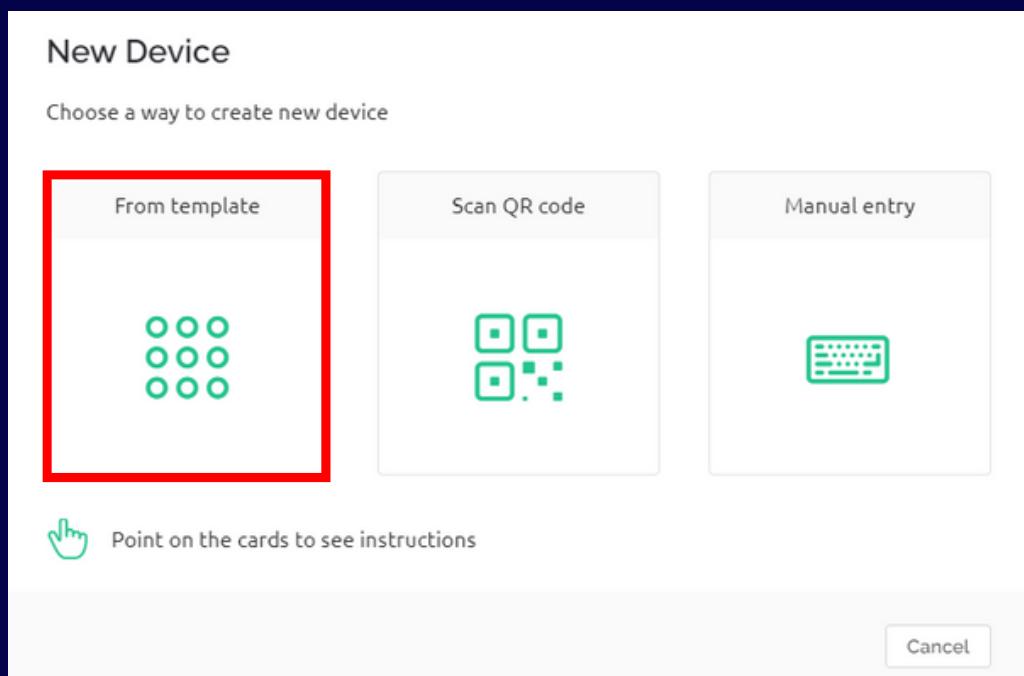
Virtual Pin Datastream

NAME	ALIAS	
<input type="text" value="Lâmpada"/> Lâmpada	<input type="text" value="L"/> L	
PIN	DATA TYPE	
<input type="text" value="V1"/> V1	<input type="text" value="Integer"/> Integer	
UNITS		
<input type="text" value="None"/> None		
MIN	MAX	DEFAULT VALUE
<input type="text" value="0"/> 0	<input type="text" value="1"/> 1	<input type="text" value="0"/> 0
+ ADVANCED SETTINGS		
		Cancel Create

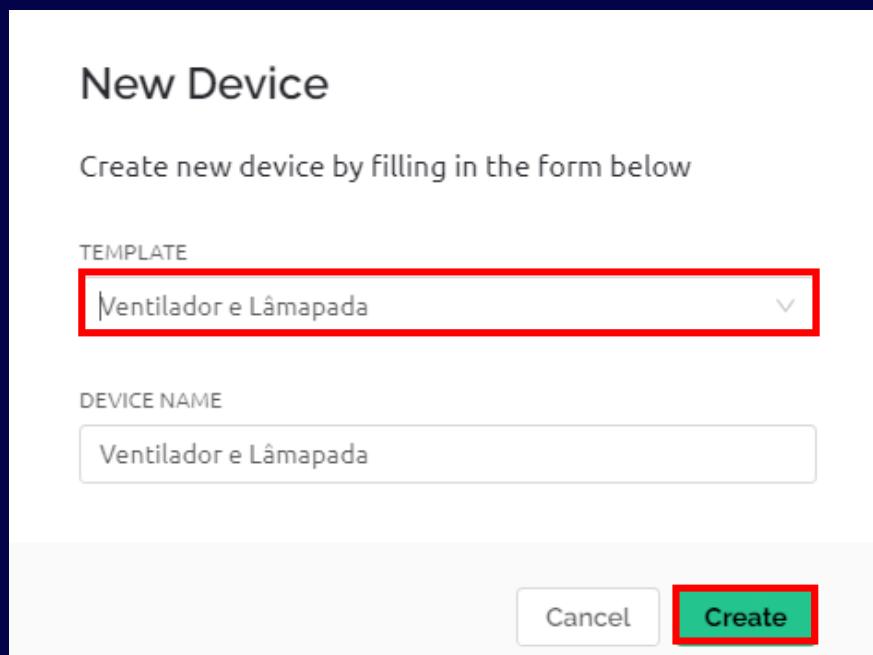
Após finalizar a **configuração** dos botões, clique sobre a **Lupa** e depois em **New Device**.

The screenshot shows the MyDevices platform interface. On the left, there is a sidebar with various icons and sections: **My organization - 9402DL**, **DEVICES** (highlighted with a red box), **LOCATIONS**, **USERS**, and settings. In the center, the **My Devices** page is displayed, showing 1 device named "Sequencial de LEDs" owned by "Rangel". A red box highlights the search icon in the sidebar and the "New Device" button in the top right corner of the main panel.

Selecione a opção **From Template**.



Em **Template** selecione o projeto que criamos anteriormente. Depois clique em **Create**.



Em **Device Info** é possível ver **informações necessárias** na hora da programação.

Ventilador e Lâmapada Offline

Device Info

STATUS: Offline LAST UPDATED: 12:26 AM Today

DEVICE ACTIVATED: 12:26 AM Today by rangelarena@gmail.com

ORGANIZATION: m

AUTHTOKEN: MzGU - ***** - **** - ****

TEMPLATE NAME: Ventilador e Lâmapada

MANUFACTURER: My organization 9402DL

FIRMWARE CONFIGURATION:

```
#define BLYNK_TEMPLATE_ID "T1"
#define BLYNK_DEVICE_NAME "Ventilador e Lâmapada"
#define BLYNK_AUTH_TOKEN
"Ma"
```

Template ID, Device Name, and AuthToken should be declared at the very top of the firmware code.

2 Devices

- Sequencial de LEDs
- Ventilador e Lâmapada

Configurando o Blynk no celular

Clique sobre a ferramenta.

Blynk

Developer Mode

Ventilador e Lâmapada

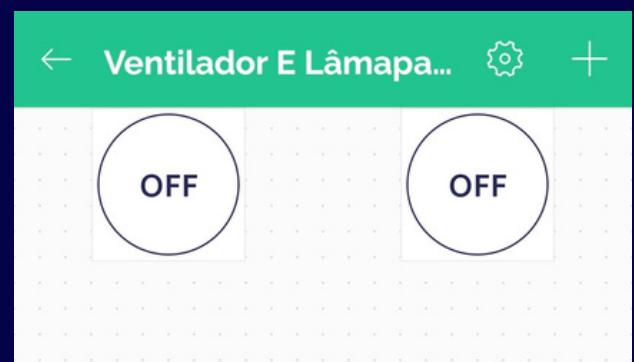
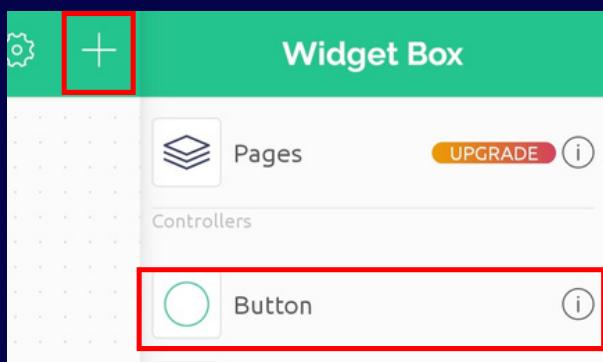
Protótipo

Sequencial de LEDs

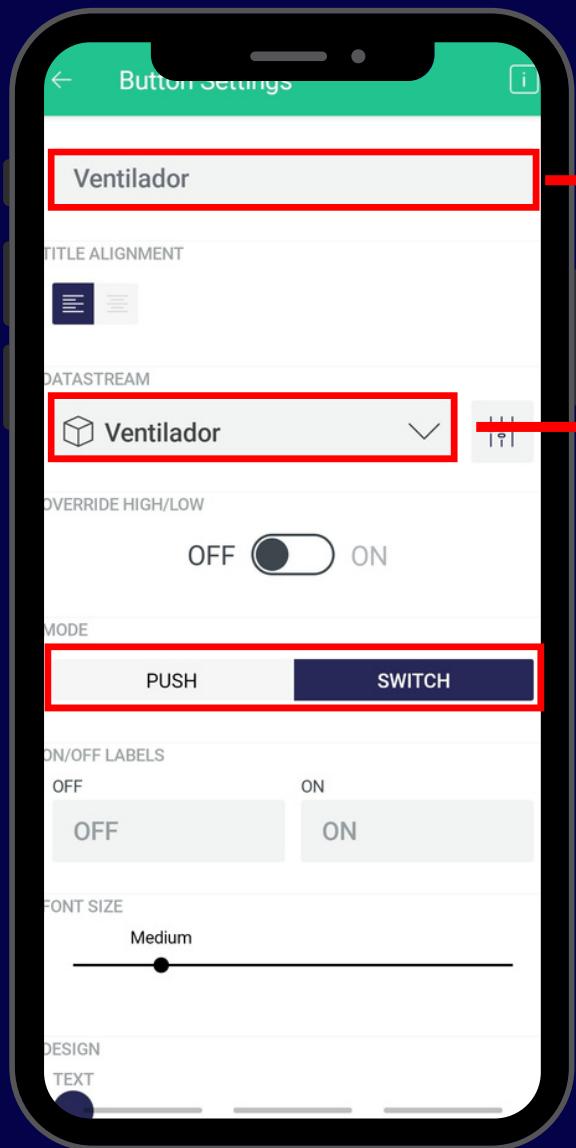
Protótipo

Selezione o projeto criado anteriormente no site.

Clique sobre o **+** e selecione dois **Button**.



Configurando os botões



Coloque o **nome** do botão.

Selecione o **Pino Virtual** que vai acionar o **dispositivo** (já definimos ele pelo site).

Selecione a opção **Switch**, dessa forma um toque liga e outro desliga o **LED**.

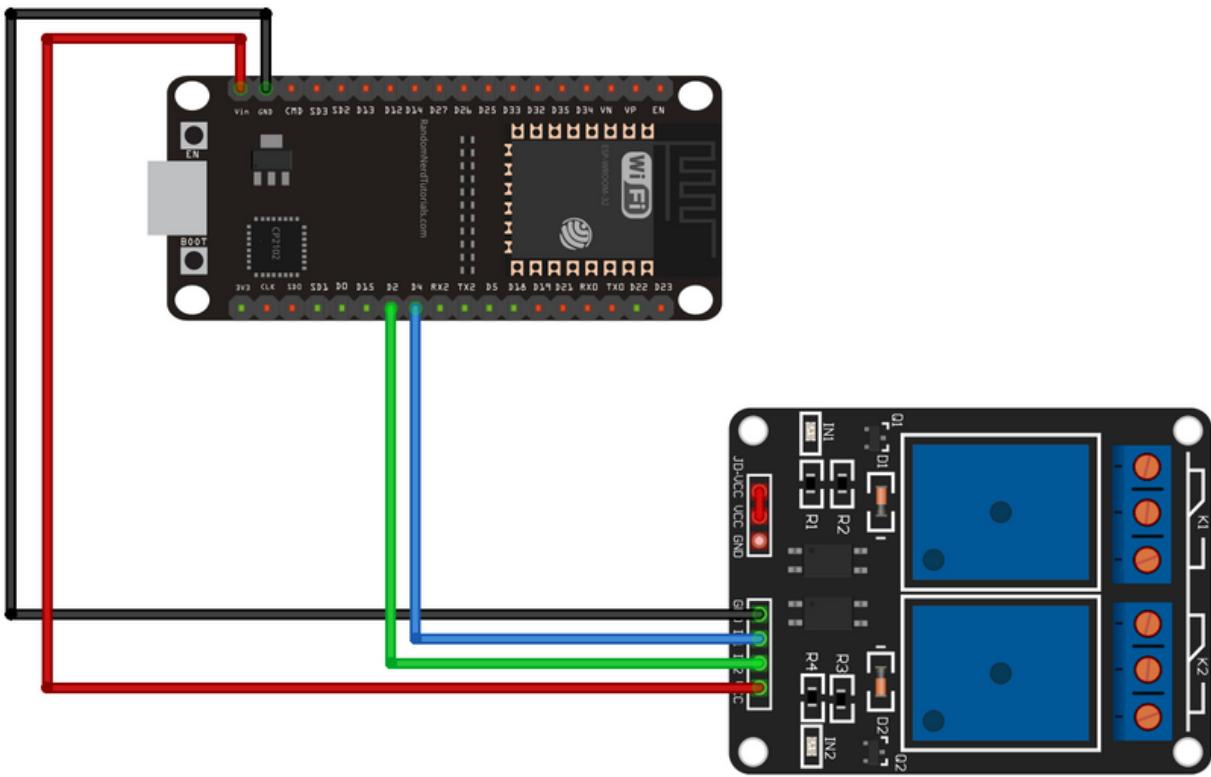
OBS: As configurações dos **botões** são iguais, basta alterar o nome e o **Pino Virtual**. Lembre-se:

Ventilador: **V0**

Lâmpada: **V1**

Após finalizar a configuração dos **botões**, basta voltar para a **tela inicial** do App.

Círcito



É necessário conectar o pino **Vin** da placa no **VCC** do Relé, visto que ele precisa de uma tensão de **5V** para funcionar.

Conecte o pino **GND** do **ESP-32** no pino **GND** do **Módulo Relé**, esse é o **negativo** do circuito.

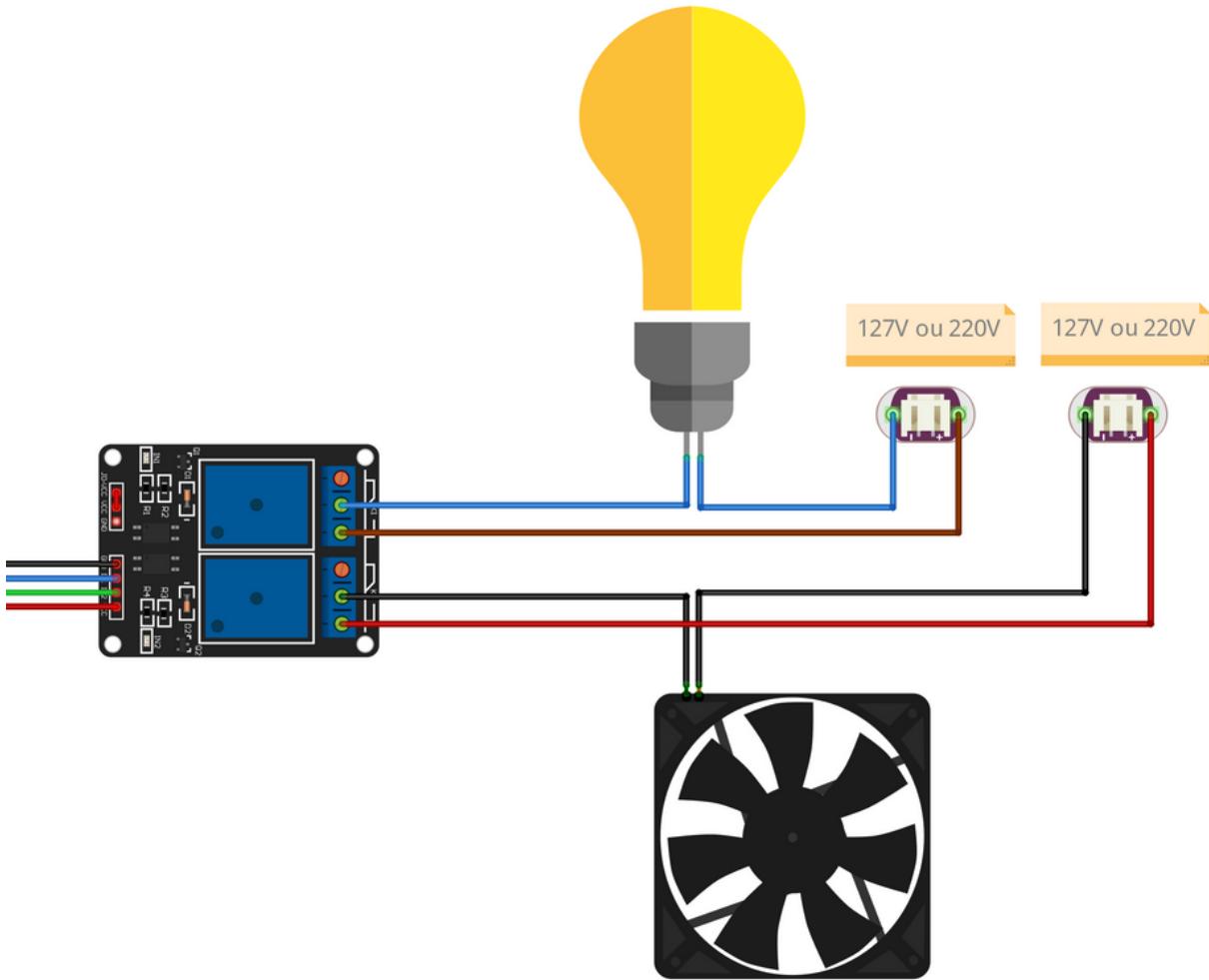
O **Pino Digital D4** deve estar no **IN1** do Relé.

O **Pino Digital D2** deve estar no **IN2** do Relé.

Resumindo:

Vin	---	VCC
G	---	GND

D4	---	IN1
D2	---	IN2



No primeiro canal do relé (**K1**), vamos conectar uma **lâmpada**.

Um terminal da lâmpada deve estar conectado no **contato aberto do relé K1** e outro na **rede elétrica**.

O **contato fechado do relé K1** deve estar conectado diretamente na **rede elétrica**.

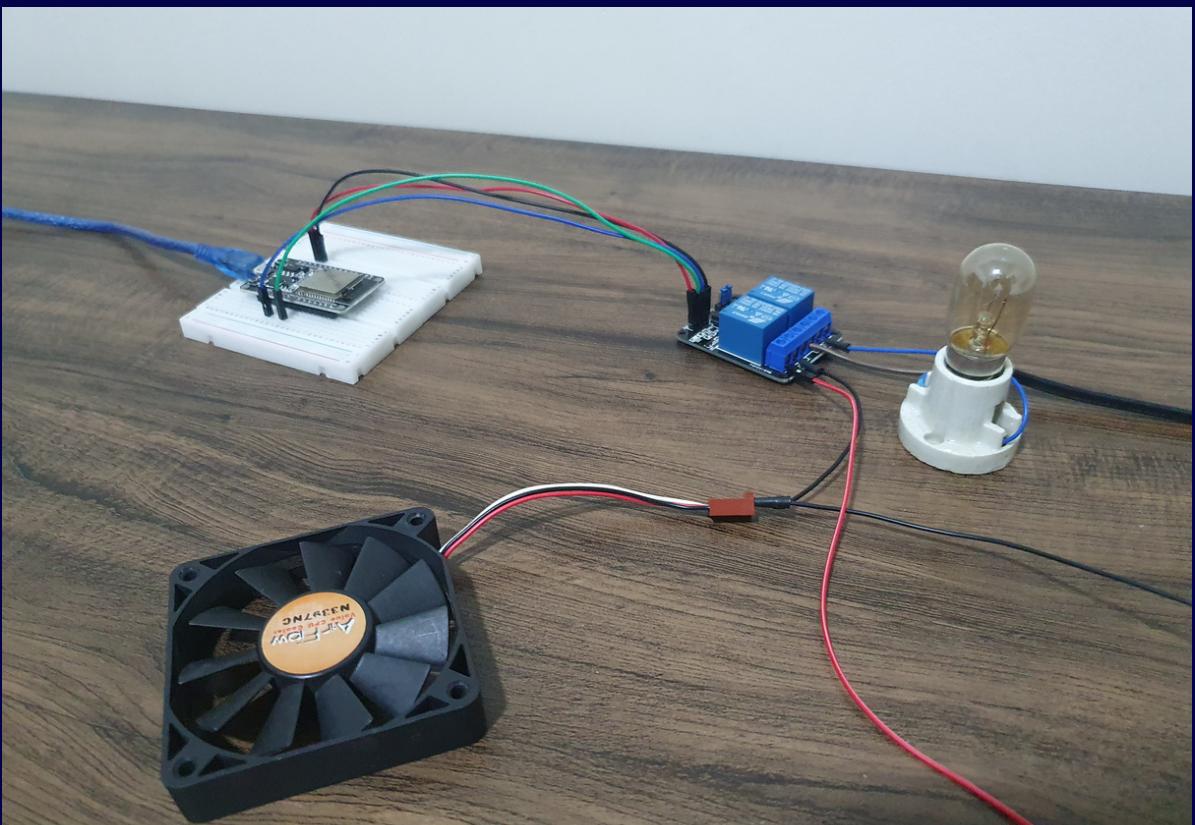
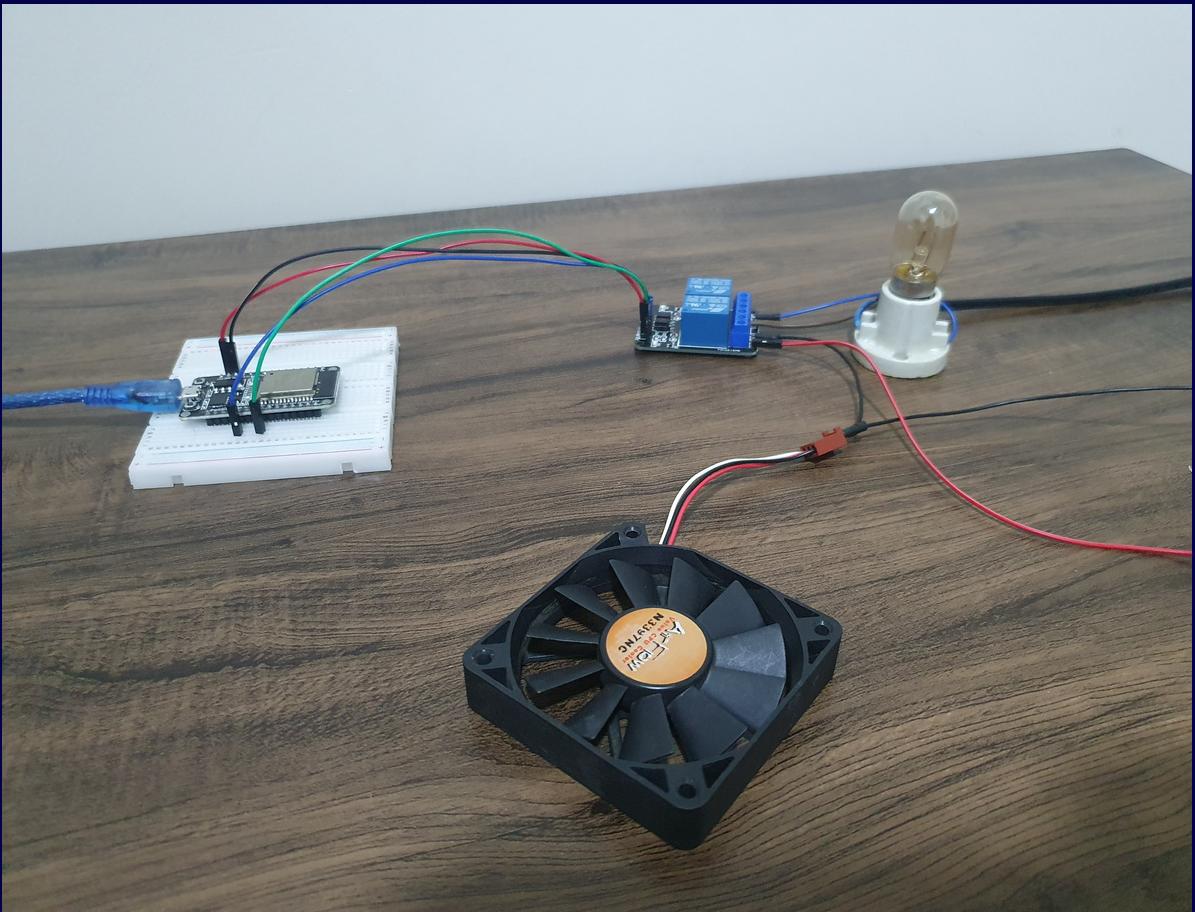
No segundo canal do relé (**K2**), será conectado uma **ventoinha** para simular um ventilador.

Um terminal do ventilador deve estar conectado no **contato aberto do relé K2** e outro na **rede elétrica**.

O **contato fechado do relé K2** deve estar conectado diretamente na **rede elétrica**.

ATENÇÃO: Caso você não tenha experiência com circuitos elétricos, para evitar acidentes peça ajuda a uma pessoa capacitada.

Círcuito na Prática



Programação

```
#define BLYNK_PRINT Serial
#define BLYNK_DEVICE_NAME "Nome do projeto"
#define BLYNK_TEMPLATE_ID "Insira o código do Template"

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>

char auth[] = "Insira o Token";
char ssid[] = "Insira o nome da rede";
char pass[] = "Insira a senha da rede";

int ventilador= D2;
int lampada= D4;

BLYNK_WRITE (V0){

    int valor = param.asInt();
    digitalWrite(ventilador, valor);
}

BLYNK_WRITE (V1){

    int valor = param.asInt();
    digitalWrite(lampada, valor);

}
```

```
void setup()
{
    Serial.begin (115200);

    Blynk.begin(auth, ssid, pass);

    pinMode(ventilador, OUTPUT);
    pinMode(lampada, OUTPUT);

}

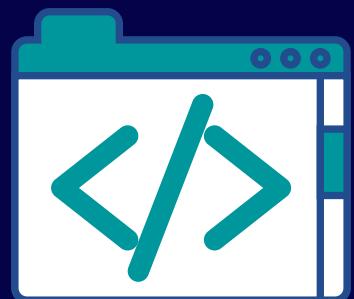
void loop()
{
    Blynk.run();
}
```

LEMBRE-SE: Quando aparecer a mensagem "**connecting**" na IDE, pressione por alguns segundo a tecla **BOOT** do **ESP-32**.

IMPORTANTE: O Token necessário para realizar a programação é apresentado no site do **Blynk**, como demonstrado na página 50.

Baixe o código do projeto [Clicando Aqui !](#)

Quer ver o funcionamento ?? [Clique aqui !!](#)



9

Monitoramento de Temperatura

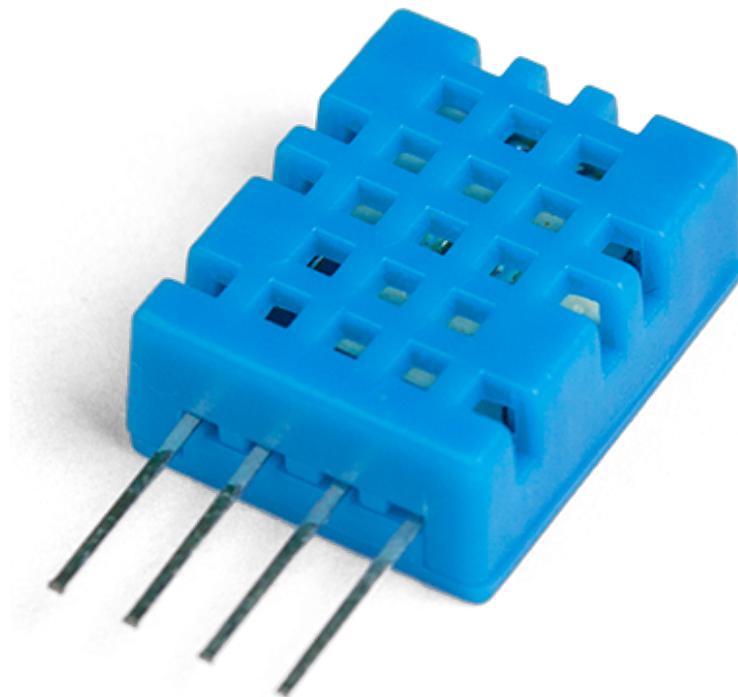
Nesse projeto, nós vamos **monitorar a temperatura e a umidade** do ambiente utilizando o **Blynk**, a placa **ESP-32** e um **sensor !!!**

Mas antes é preciso conhecer um pouco mais sobre os componentes que serão utilizados no projeto.

Sensor DHT11

O **Sensor DHT11** é um sensor de **temperatura** e **umidade** que permite fazer leituras de temperaturas entre **0** a **50** Celsius e umidade entre **20** a **90%**, é muito utilizado em projetos com **microcontroladores**.

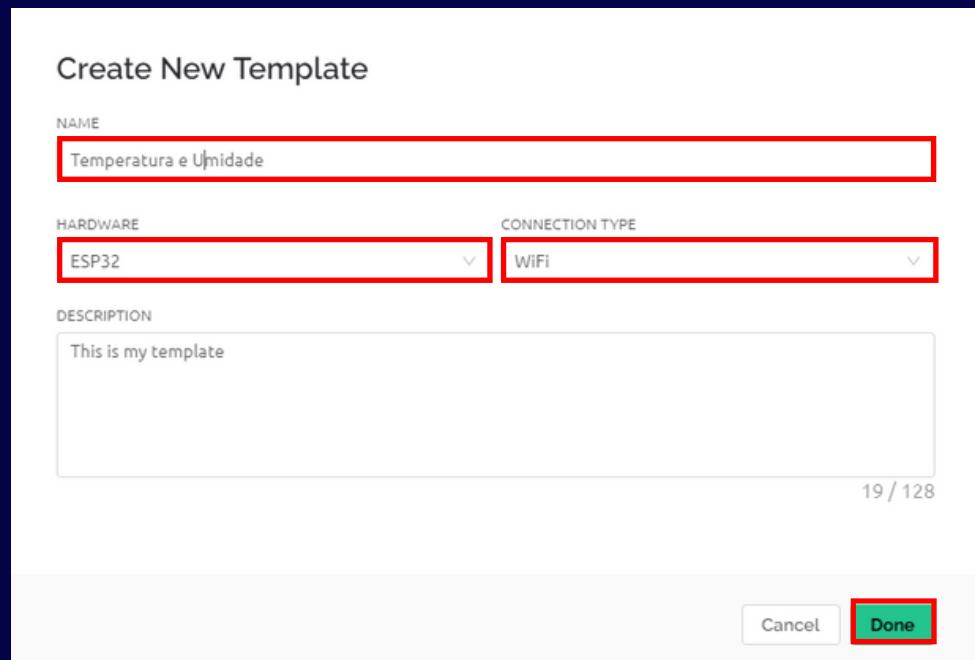
O elemento sensor de temperatura é um **termistor** do tipo **NTC** e o sensor de Umidade é do tipo **HR202**, o circuito interno faz a leitura dos sensores e se comunica a um **microcontrolador** através de um sinal serial de uma via.



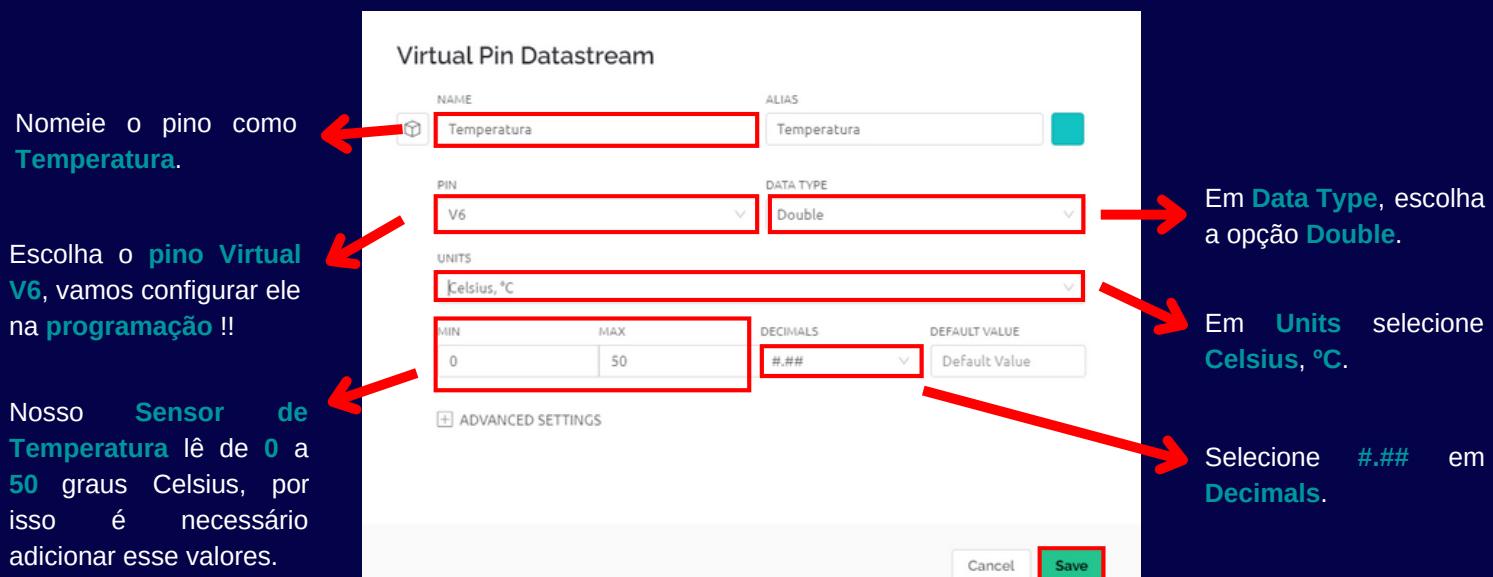
Configurando o Blynk

Primeiro vamos criar um novo **template** pelo site do **Blynk**. Se houver dúvidas de como criar, siga o passo a passo nos projetos acima.

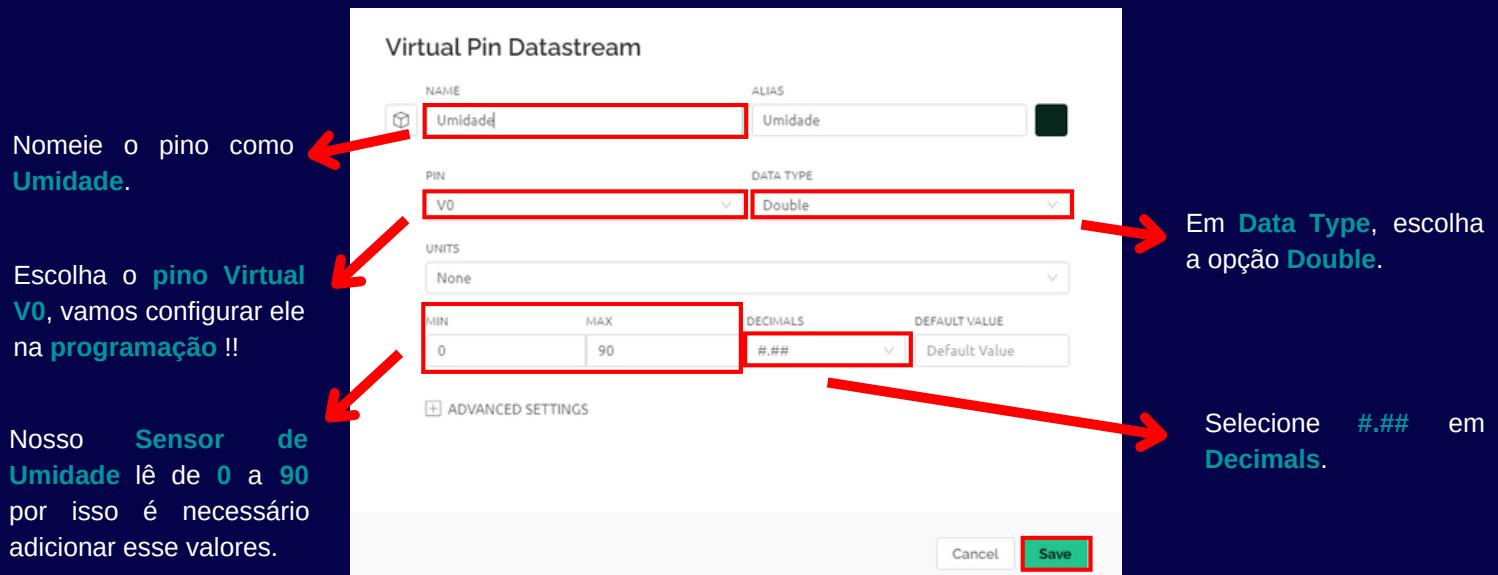
Adicione um nome para o **template**, em **Hardware** selecione **ESP32**, em **Connection Type** selecione **WiFi**. Após concluir clique em **Done**.



Depois, clique em **Datastreams**, em seguida **New Datastream** e escolha a opção **Virtual Pin**. Vamos iniciar configurando o **pino virtual** para a **Temperatura**.



Agora vamos configurar o **pino virtual** responsável pela **umidade**.

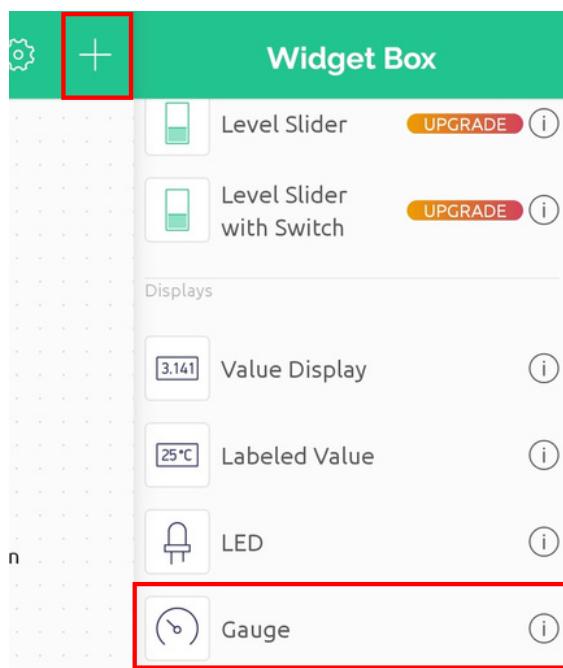


Após finalizadas as configurações dos **pinos virtuais**, clique sobre a lupa e depois sobre **New Device**. Selecione a opção **From Template** e escolha o projeto que acabamos de criar. Em **Device Info** é possível ver informações **necessárias** para a programação. Se causo houver dúvida siga o passo a passo na página 48.

Configurando o Blynk no celular



Clique sobre o **+** e adicione dois Medidores (**Gauge**), um para monitorarmos a **temperatura**, e outro a **umidade** do ambiente.



Configurando os Medidores



Medidor de Temperatura

Para iniciar a configuração, basta clicar sobre o **medidor**.

A screenshot of the "Gauge Settings" configuration screen. It has several sections: "TITLE ALIGNMENT" (with icons for left, center, and right alignment), "DATASTREAM" (with a dropdown menu set to "Temperatura" and a pin selection button), "FONT SIZE" (with a slider set to "Medium"), and "DESIGN" (with a "TEXT" button). A red box highlights the "Temperatura" input field in the DATASTREAM section. An arrow points from this field to the text "Coloque o nome no medidor de temperatura.". Another red box highlights the "Temperatura" dropdown in the DATASTREAM section. An arrow points from this dropdown to the text "Selecione o pino virtual da temperatura (definimos pelo site).".

← Gauge Settings ⓘ

Temperatura → Coloque o **nome** no medidor de **temperatura**.

TITLE ALIGNMENT

DATASTREAM

Temperatura ↘ ↗

FONT SIZE

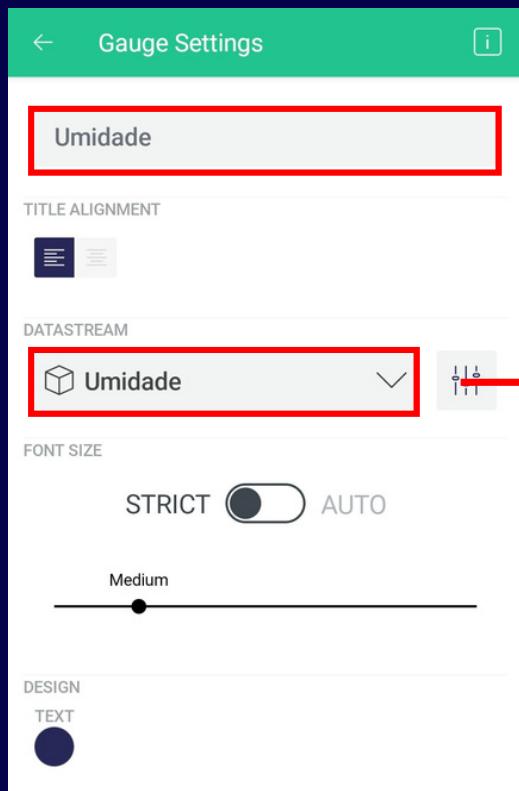
STRICT AUTO

DESIGN

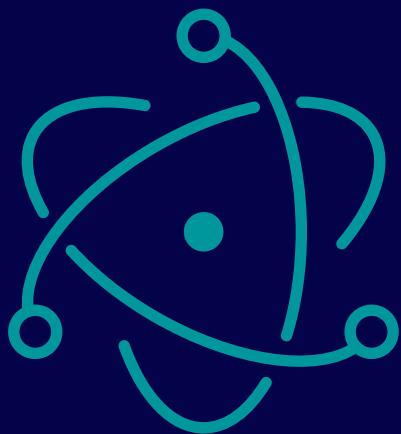
TEXT



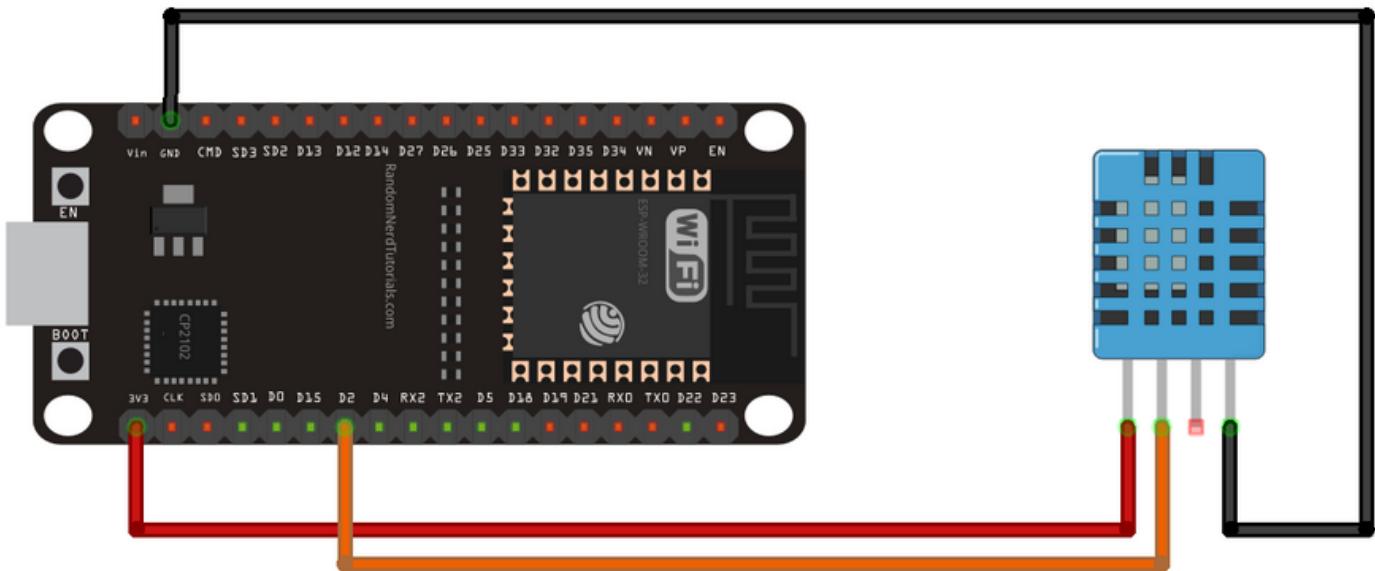
Medidor de Umidade



Após finalizar as **configurações**, basta **salvar** e voltar para a **página inicial** do **Blynk** e selecionar o **projeto**.



Circuito



O primeiro pino do sensor é utilizado para alimentação, conecte ele nos **3.3V** do **ESP-32**.

Conecte o segundo pino do **DTH11** no pino **D2** da nossa placa, ele é responsável por enviar a leitura da temperatura e da umidade.

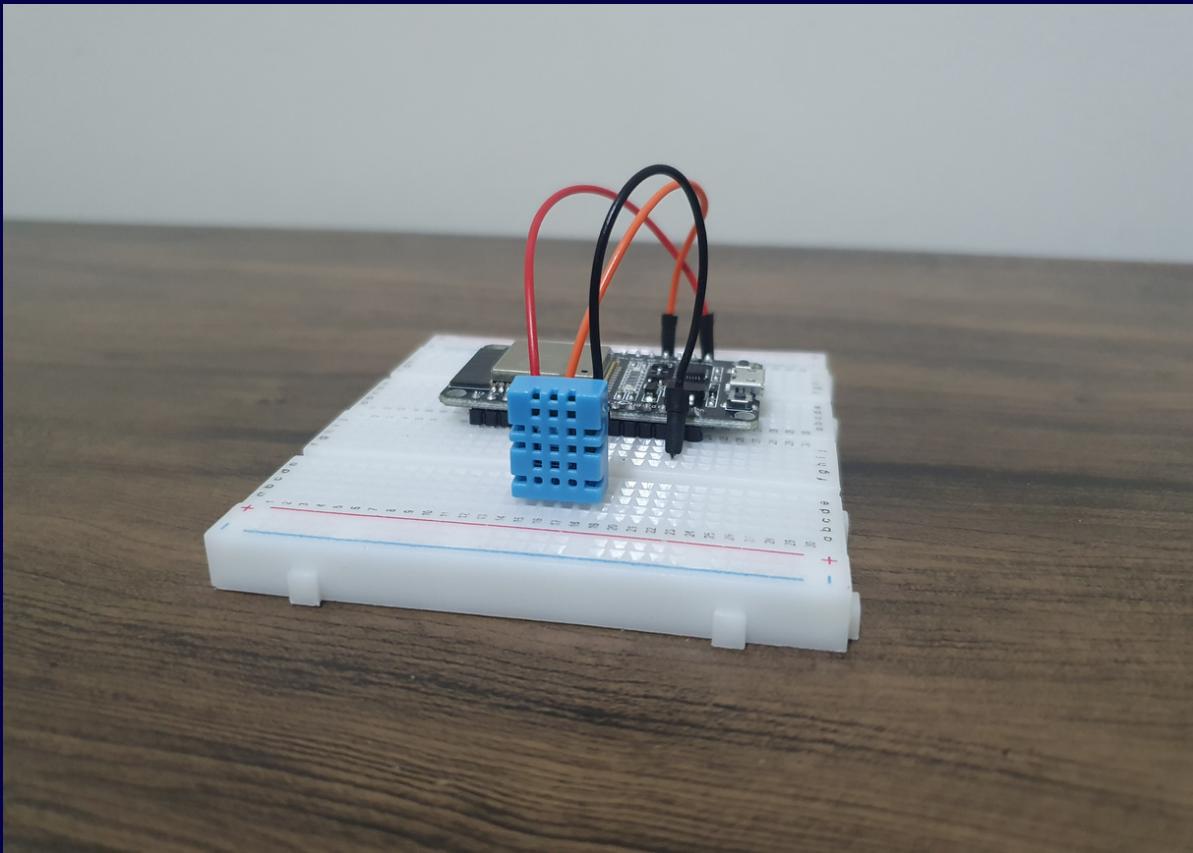
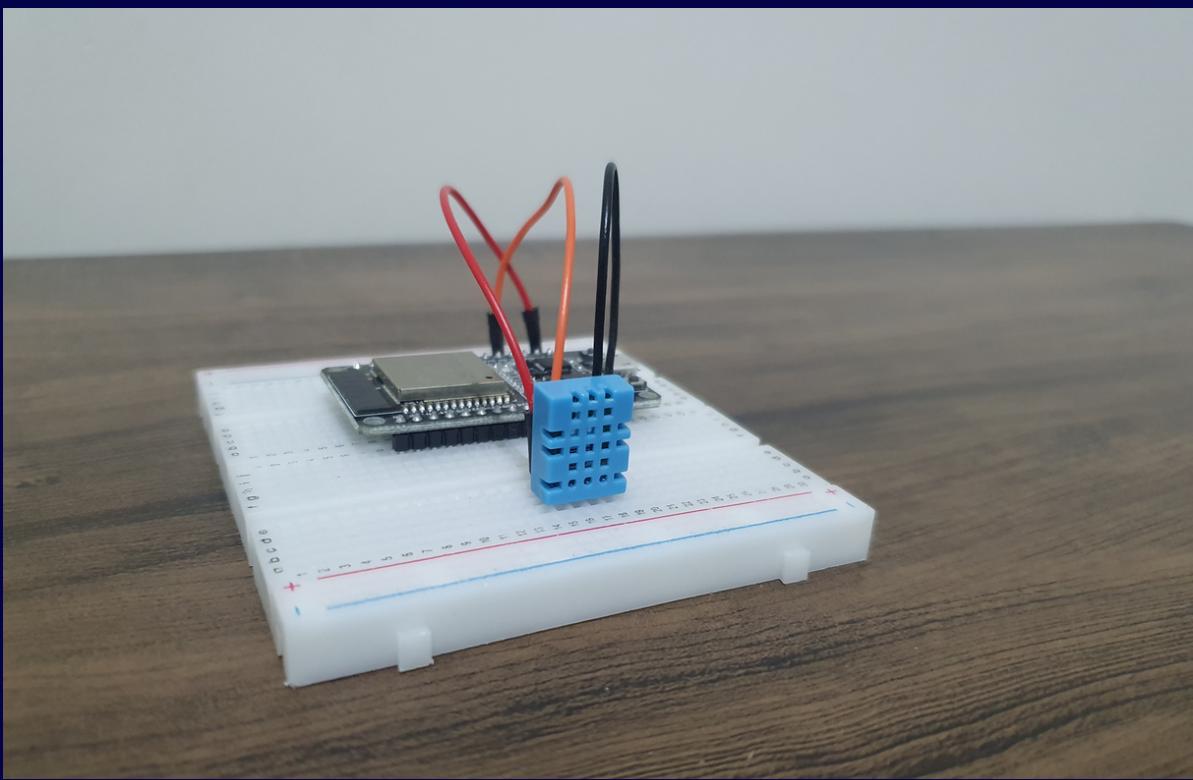
O terceiro pino não será utilizado. Por fim, conecte o ultimo pino no pino **GND** da placa.

Resumindo:

Pino 1 ----	3V
Pino 2 ----	D2
Pino 3 ----	-
Pino 4 ----	GND

OBS: Para facilitar a montagem do circuito, utilize a **Protopboard**.

Círcuito na Prática



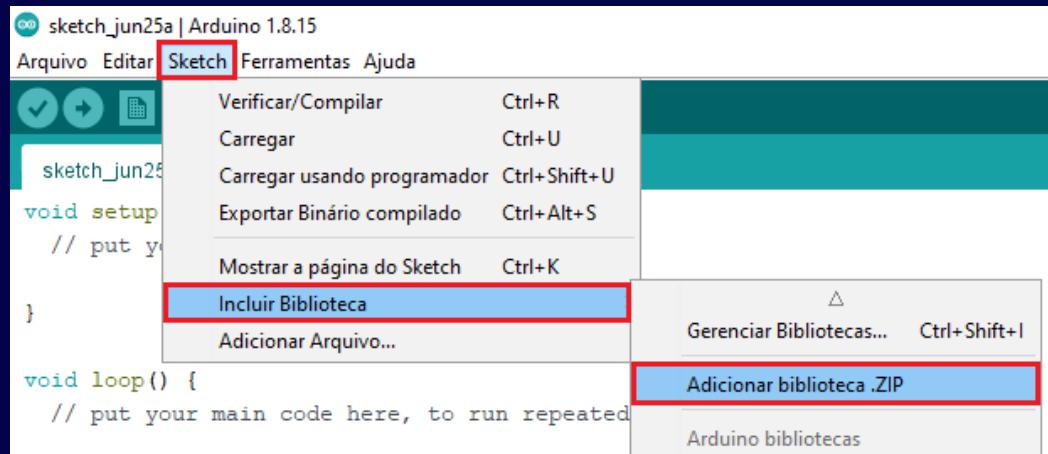
Instalando Bibliotecas

Para conseguirmos programar nosso sensor, é necessário adicionarmos a biblioteca “**DHT.h**”, você pode baixá-la [Clicando Aqui](#).

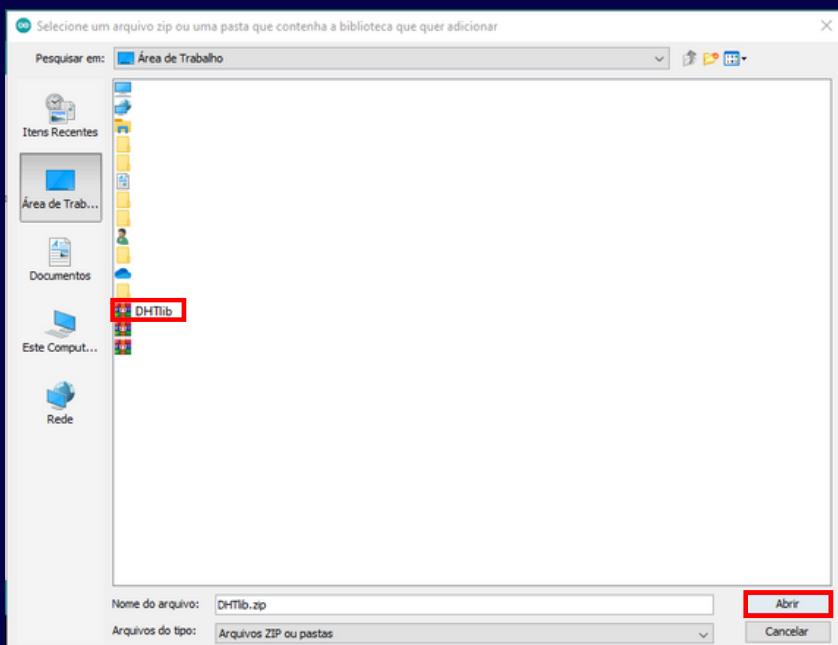
Para Adicionar a **biblioteca** no Arduino, **siga o passo a passo abaixo:**



Abra a **IDE do Arduino**, clique em **Sketch**, selecione **Incluir Biblioteca** e depois clique em **Adicionar Biblioteca.zip**.



Selecione o arquivo **.zip** e clique em **Abrir** e pronto, a biblioteca necessária já está instalada.



Programação

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "Insira o código do Template" // mostrado no site
#define BLYNK_DEVICE_NAME "Nome do projeto"

// Biblioteca necessárias

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>

char auth[] = "Insira o Token"; // Aqui é necessário inserir o token exibido no site
char ssid[] = "Insira o nome da Rede"; // Insira o nome da rede Wi-fi utilizada
char pass[] = "Insira a senha da Rede"; // Insira a senha da rede Wi-fi utilizada

#define DHTPIN 2 // Aqui é o pino digital que estamos utilizando, no nosso caso
D2 (GPIO 2)

#define DHTTYPE DHT11 // Declarando o sensor

DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

void sendSensor() {
```

```

float h = dht.readHumidity();
float t = dht.readTemperature();

if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
}

Blynk.virtualWrite(V0, h); // Pino Virtual 0 para umidade
Blynk.virtualWrite(V6, t); // Pino Virtual 6 para temperatura
}

void setup()
{
    Serial.begin(9600);

    Blynk.begin(auth, ssid, pass);

    dht.begin();

    timer.setInterval(1000L, sendSensor);
}
void loop()
{
    Blynk.run();
    timer.run();
}

```

OBS: O **token** necessário para a programação é apresentado no site do **Blynk**, em Device Info, como mostra a página 50.

Baixe o código do projeto [Clicando Aqui!](#)

Quer ver o funcionamento na prática ?? [Clique aqui!!](#)

10 Alerta de Chuva

Nesse próximo projeto, vamos fazer um **sistema de alerta de chuva**. Para isso, vamos utilizar um **módulo sensor de chuva**, nossa placa **NodeMCU** e o aplicativo **Blynk**. Nosso projeto funcionará da seguinte maneira: sempre que começar a chover, uma **notificação** aparecerá no celular alertando.

Módulo Sensor de Chuva

Este **Sensor de Chuva** pode ser usado para monitorar uma variedade de condições climáticas como gotas de chuva ou neve. Quando o clima está seco a saída do sensor fica em estado **alto** e quando há uma gota de chuva, a saída fica em estado **baixo**. O limite entre tempo seco e chuva pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0.

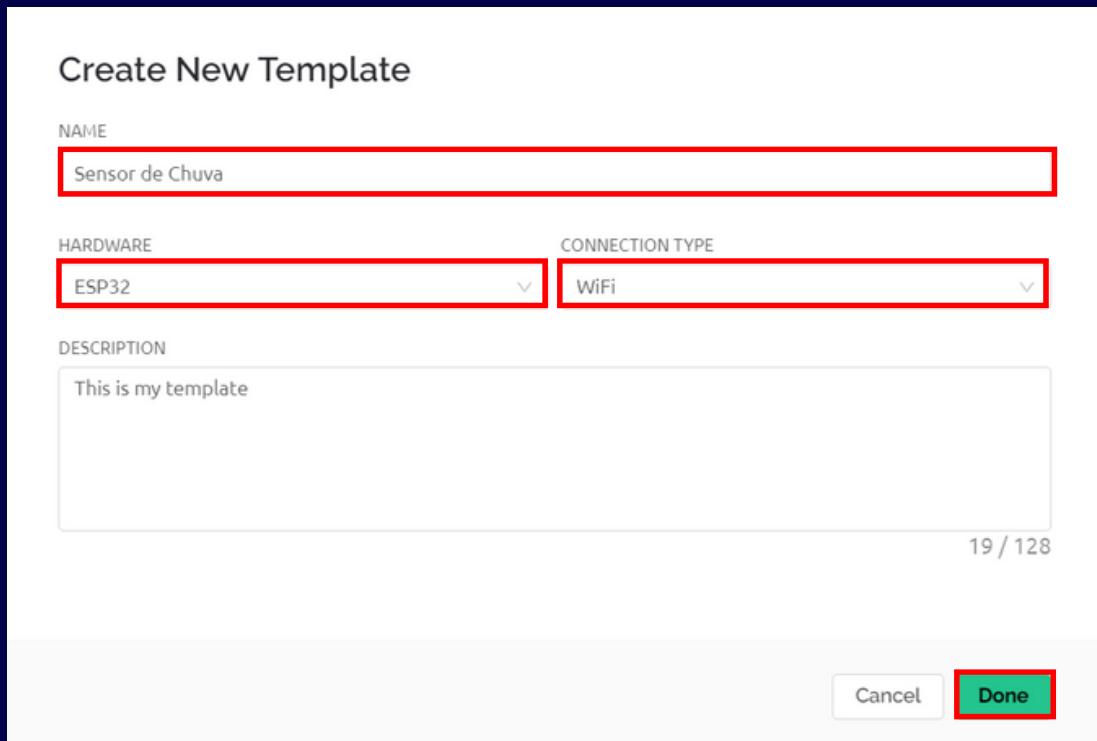


A placa do **Sensor de Chuva** é revestida em ambos os lados com um tratamento de **níquel contra oxidação**, melhorando assim a condutividade, desempenho e duração.

Configurando o Blynk

Primeiro vamos criar um novo **template** pelo site do [Blynk](#). Se houver dúvidas de como criar, siga o passo a passo nos projetos acima.

Adicione um nome para o **template**, em **Hardware** selecione **ESP32**, em **Connection Type** selecione **WiFi**. Após concluir clique em **Done**.



Depois, clique em **Datastreams**, em seguida **New Datastream** e escolha a opção **Virtual Pin**. Vamos iniciar configurando o **pino virtual** para o **Sensor de Chuva**.



Nomeie o pino como **Chuva**.

Escolha o **pino Virtual V0**, vamos configurar ele na **programação !!**

Essa mensagem aparecerá enquanto o celular se conecta ao nosso **hardware**.

Após finalizar as configurações do pino virtual, clique em **Events** e depois em **Add New Event**, para adicionarmos um novo evento.

The screenshot shows a configuration interface for a 'Sensor de Chuva'. At the top, there are tabs: Info, Metadata, Datastreams, **Events** (which is highlighted with a red box), Automations, Web Dashboard, and Mobile Dashboard. Below the tabs is a search bar labeled 'Search event' and a green button labeled '+ Add New Event' (also highlighted with a red box). On the left, there's a sidebar with icons for Home, Sensors, Automations, and Help. The main area displays a table with columns: Name, Code, Color, Type, Description, and Actions. Two rows are shown: 'Online' (Code: ONLINE, Color: Green, Type: Online) and 'Offline' (Code: OFFLINE, Color: Red, Type: Offline). A 'Save And Apply' button is at the top right.

Em **General**, de um nome ao evento, em **Type** selecione a opção **Warning**. Deixe ativado **Send event to Notifications tab** e **Send event to Timeline**.

The screenshot shows the 'Edit Event' dialog. The 'General' tab is selected. In the 'EVENT NAME' field, 'Chuva' is entered. In the 'EVENT CODE' field, 'chuva_' is entered. Under 'TYPE', the 'Warning' option is selected (highlighted with a red box). In the 'DESCRIPTION (OPTIONAL)' field, 'Começou a chover' is written. At the bottom, two checkboxes are shown in a red box: 'Send event to Notifications tab' (checked) and 'Send event to Timeline' (checked). Both descriptions below the checkboxes mention making the event visible in the respective mobile app tabs.

Após concluir as configurações em **General**, clique sobre **Notifications**.

Ative a opção **Enable notifications**. Em **Default recipients** na opção **E-mail to** e **Push notifications to** selecione a opção **Device Owner**. Em **Notifications limit**, selecione **1 minuto** em **Limit Period** e **1** em **Event Counter**. Depois, basta salvar.

Enable notifications

Default recipients

E-MAIL TO

PUSH NOTIFICATIONS TO

SMS TO

Notifications limit

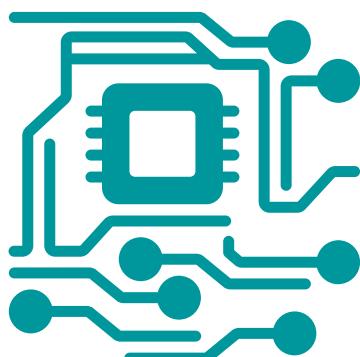
LIMIT PERIOD

EVENT COUNTER

Notifications Management
When turned ON, end-users will access advanced notification management for this event

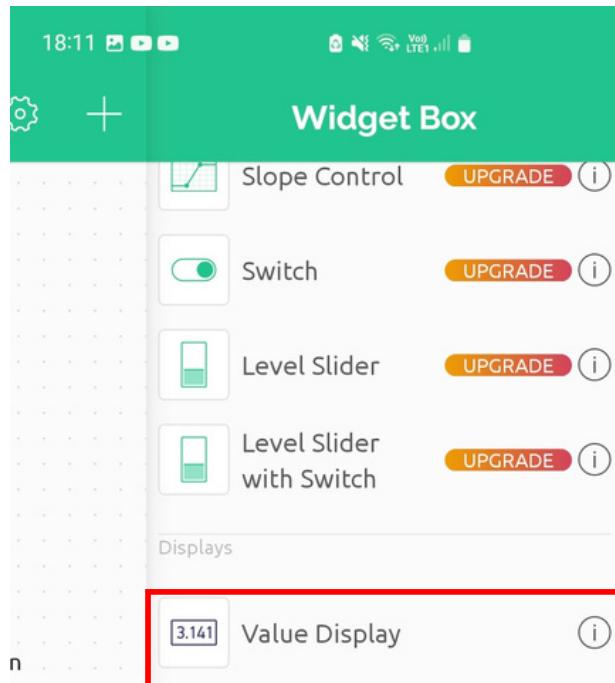
Enable notifications management

Após finalizadas as configurações, clique sobre a lupa e depois sobre **New Device**. Selecione a opção **From Template** e escolha o projeto que acabamos de criar. Em **Device Info** é possível ver informações **necessárias** para a programação. Se causo houver dúvida siga o passo a passo na página 37.



Configurando o Blynk no celular

Primeiro é necessário abrir o aplicativo do **Blynk**. Após entre no **modo de desenvolvedor** e selecione o projeto do **sensor de chuva**. Vamos precisar de um **display**.

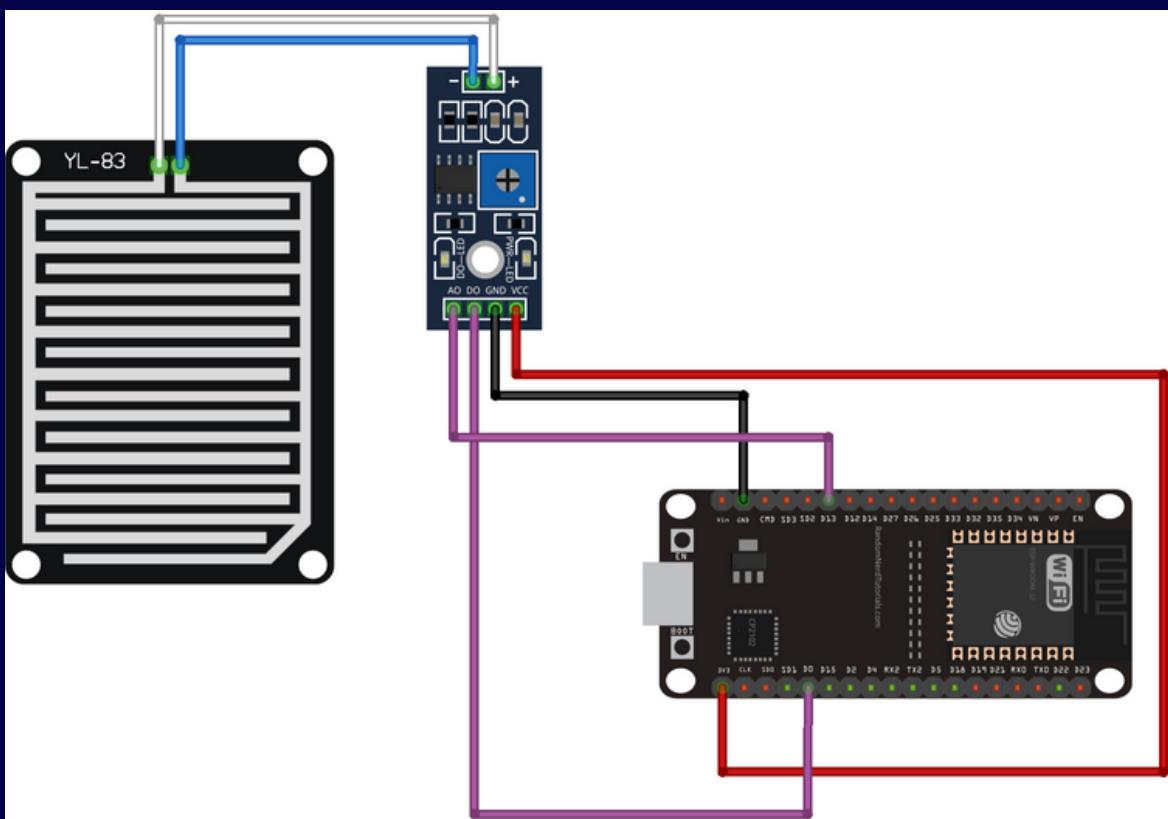


The screenshot shows the 'Value Display Settings' screen. It includes fields for 'Title (optional)', 'TITLE ALIGNMENT' (with two icons), 'DATASTREAM' (set to 'Chuva'), 'FONT SIZE' (with 'STRICT' and 'AUTO' options and a slider set to 'Medium'), and 'DESIGN' (with a 'TEXT' button). A red box highlights the 'DATASTREAM' dropdown, and a red arrow points from it to the explanatory text on the right.

Para configura-lo basta somente em **Datastream** selecionar o pino virtual que criamos pelo site.

Após realizar a **configuração**, basta salvar e voltar para a página inicial do **App**.

Circuito



Primeiro é necessário conectar dois jumper no sensor e conectar na **placa de controle**, no negativo e positivo.

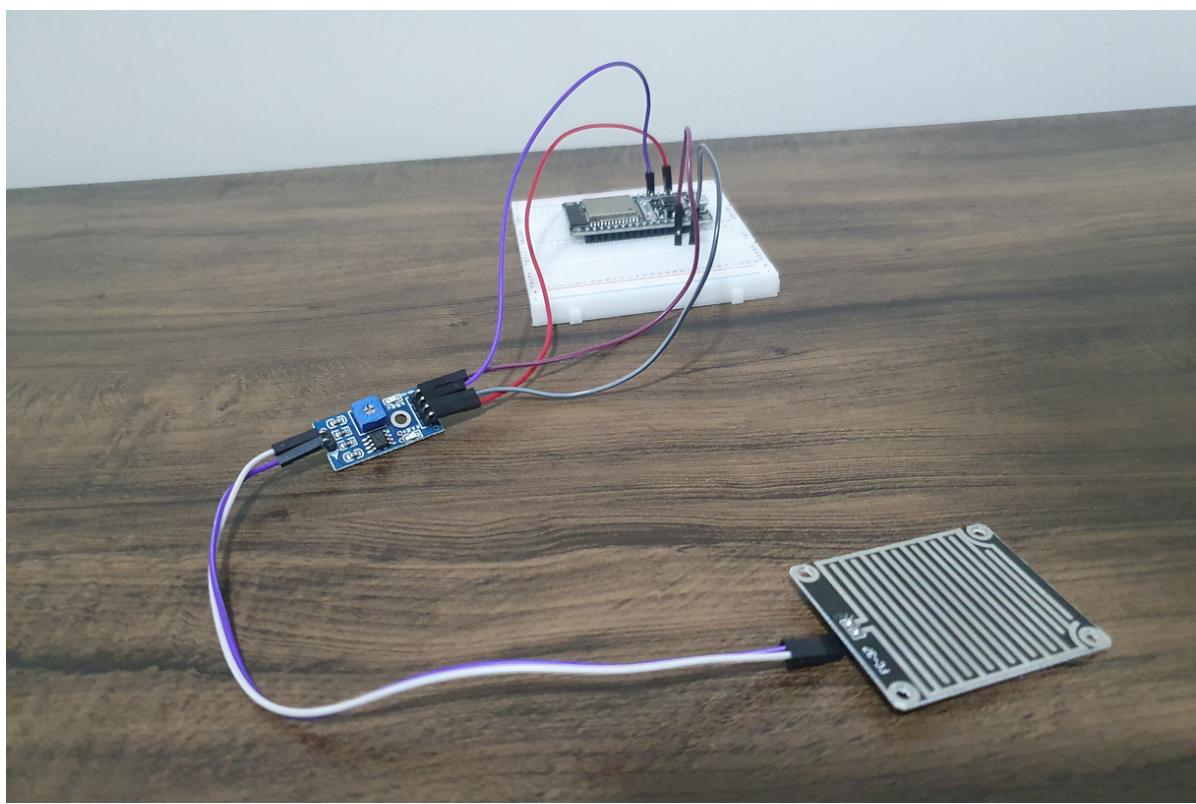
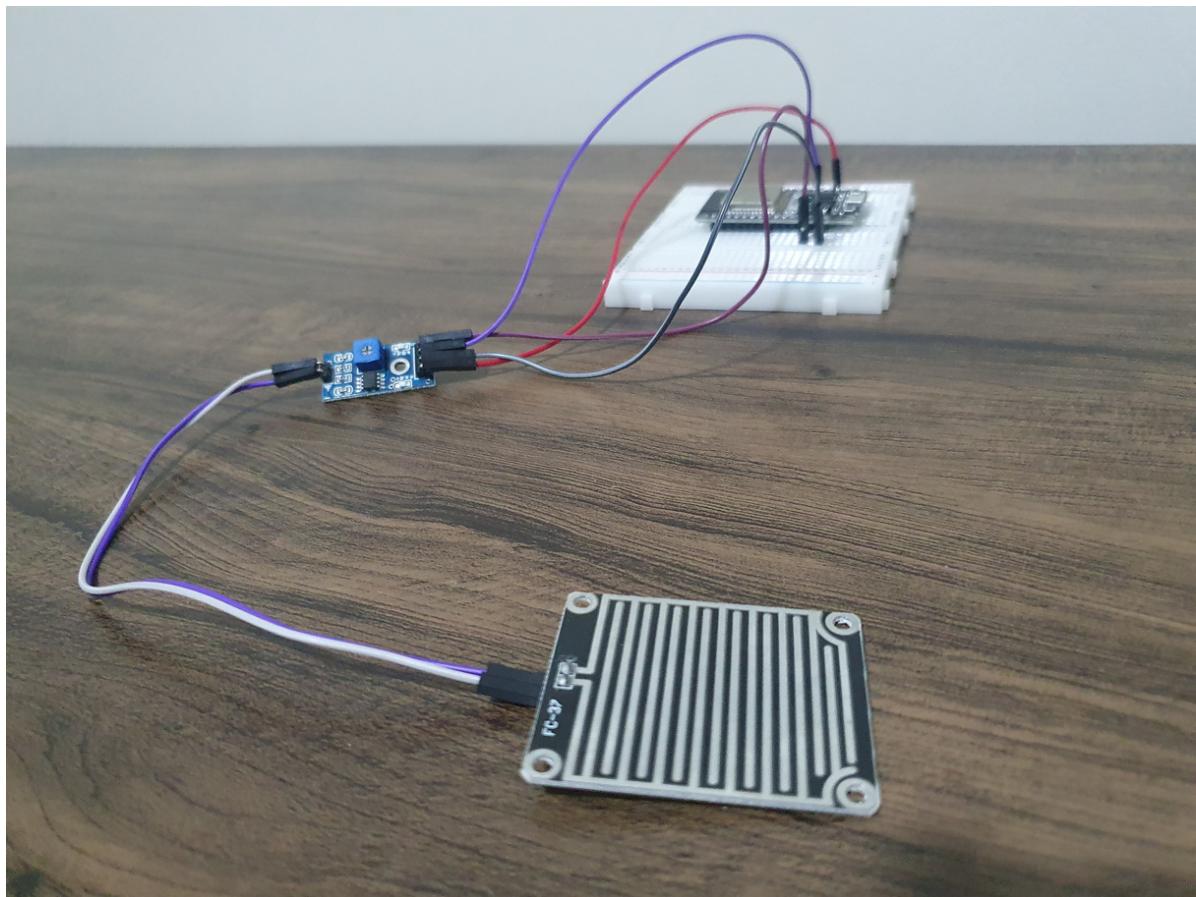
Conecte o pino analógico (**A0**) da **placa de controle** no pino analógico do **ESP-32 (D13)**.

Conecte o pino digital (**D0**) da **placa de controle** no pino analógico do **ESP-32 (D0)**.

O pino **GND** da **placa de controle** deve ser conectado no pino **G** do microcontrolador.

Por fim, é necessário conectar o pino **VCC** da **placa de controle** no pino **3V do ESP-32**.

Círcuito na Prática



Programação

```
#define BLYNK_TEMPLATE_ID "Insira o template ID"
#define BLYNK_DEVICE_NAME "Insira o Nome do Projeto"
#define BLYNK_AUTH_TOKEN "Insira o Token"

char ssid[] = "Nome da Rede";
char pass[] = "senha";

#define RAIN_SENSOR 13

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
BlynkTimer timer;

int RAIN_SENSOR_Value = 0;
bool isconnected = false;
char auth[] = BLYNK_AUTH_TOKEN;

#define VPIN_BUTTON_2 V0

void checkBlynkStatus() {
    isconnected = Blynk.connected();
    if (isconnected == true) {

        sendData();

    }
    else{
        Serial.println("Blynk Not Connected");
    }
}
```

```

void getSensorData()
{
    RAIN_SENSOR_Value = digitalRead(RAIN_SENSOR);
    if (RAIN_SENSOR_Value == 0 ){
    }
    else{
    }
}
void sendData()
{
    Blynk.logEvent("rain", "Water Detected!");
    Blynk.virtualWrite(VPIN_BUTTON_2, "Começou a Chover !!");

    if (RAIN_SENSOR_Value == 1 )
    {
        Blynk.virtualWrite(VPIN_BUTTON_2, "Não está chovendo.");
    }
}
void setup()
{
    Serial.begin(9600);
    pinMode(RAIN_SENSOR, INPUT);
    WiFi.begin(ssid, pass);
    timer.setInterval(2000L, checkBlynkStatus);
    Blynk.config(auth);
    delay(1000);
}
void loop()
{
    getSensorData();
    Blynk.run();
    timer.run();
}

```

Baixe o código do projeto [Clicando Aqui !](#)

Quer ver o funcionamento ?? [Clique aqui !!](#)



11

Controlando um Display LCD

Nesse próximo projeto, vamos aprender a controlar um **Display LCD** utilizando o **ESP-32!!**

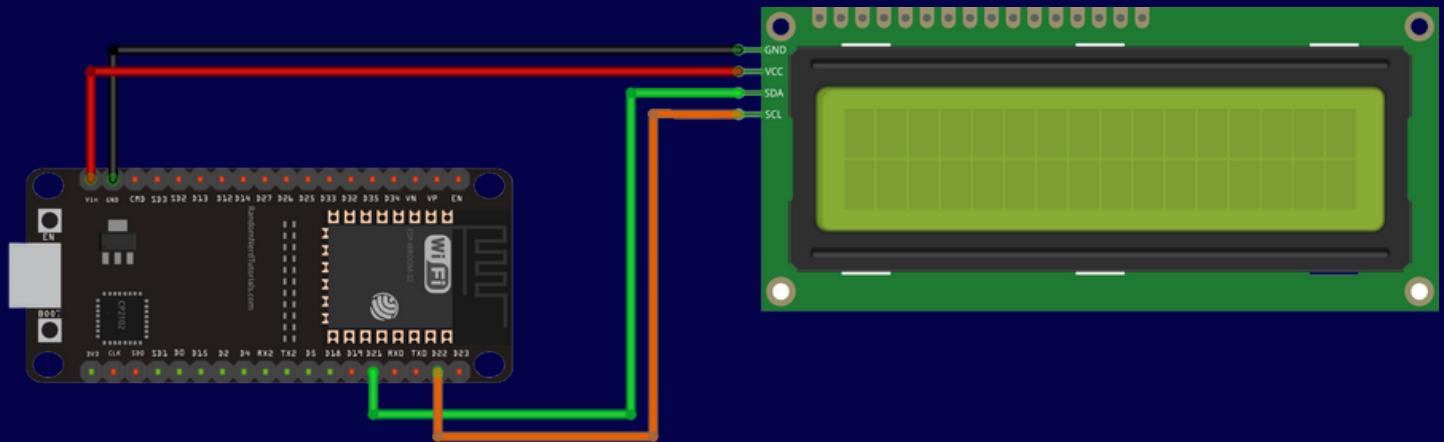
Display LCD 16x2 e I2C

O Display LCD serve para escrevermos textos e sinalizações, tem outras diversas aplicações com **microcontroladores**, como **NodeMCU** e **Arduino**. São 16 colunas por 2 linhas, backlight azul e escrita branca. Possui o controlador HD44780 usado em toda indústria de LCD's como base de interface.



Uma vantagem que esse display traz é o **Módulo I2C** integrado, dessa forma você faz a conexão entre o **microcontrolador** e o display utilizando apenas os pinos **SDA** e **SCL**, deixando as outras portas livres para o desenvolvimento do seu projeto.

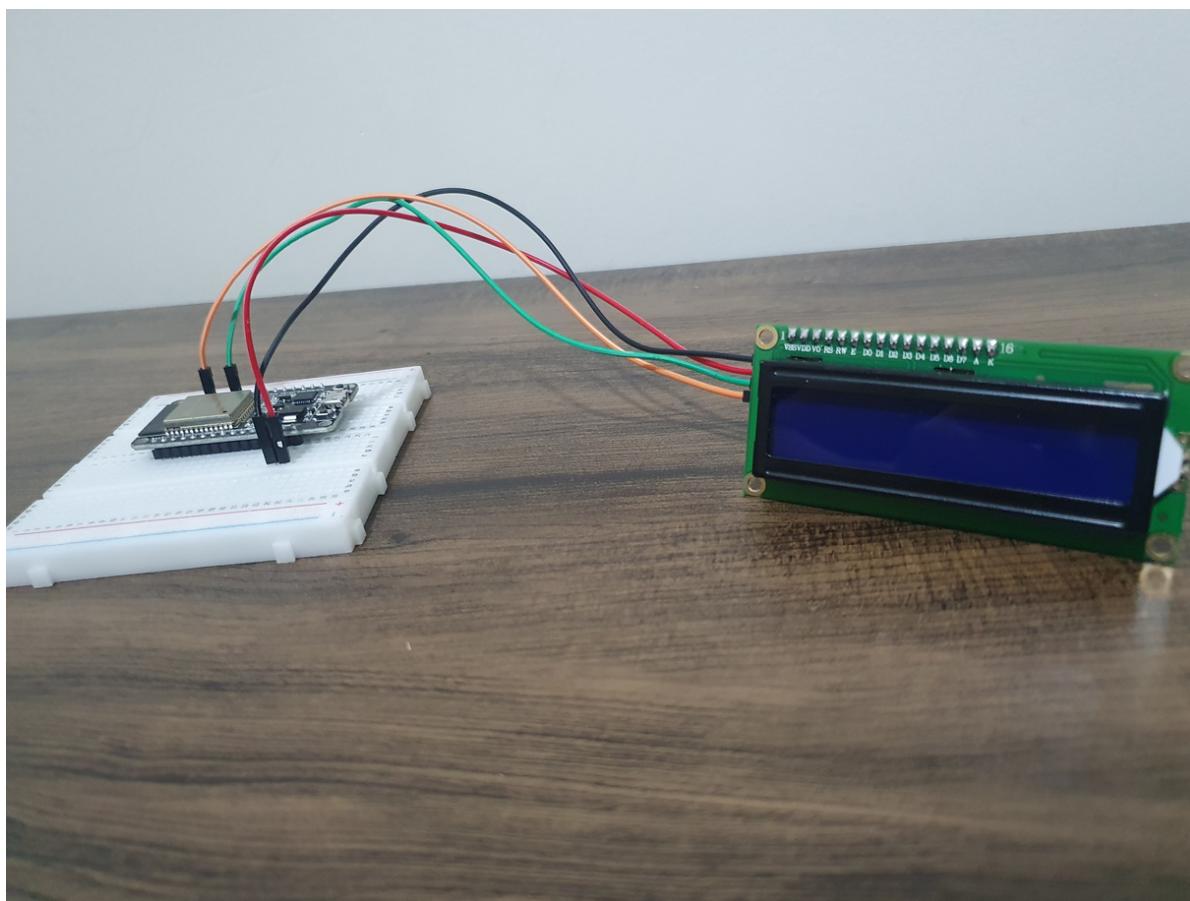
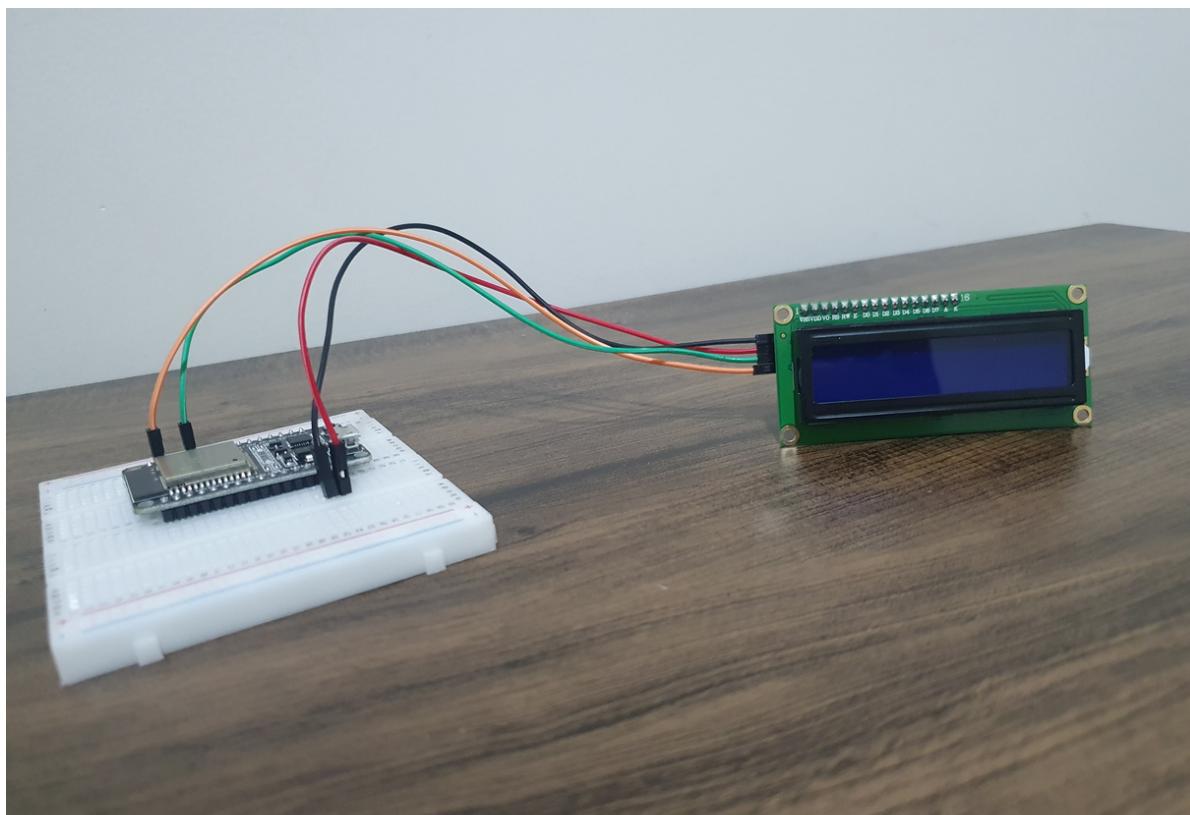
Círcuito



Resumo

GND ---- Pino G
VCC ---- Vin
SDA ---- D21
SCL ---- D22

Círcuito na Prática



Programação

Para nossa programação funcionar corretamente, é necessário instalarmos a biblioteca **LiquidCrystal I2C** na **IDE do Arduino**, para baixar a biblioteca, [Clique Aqui](#).

```
// Incluindo bibliotecas necessárias
```

```
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

```
void setup () {
```

```
lcd.begin(16,2); // linhas e colunas do display
```

```
lcd.init();
```

```
lcd.backlight();
```

```
}
```

```
void loop () {
```

```
lcd.setCursor(0,0); // coluna 0, linha 0
```

```
lcd.print(" E-book IoT"); // Insira seu texto Aqui
```

```
delay(5000); // Espera de 5 segundos
```

```
lcd.clear(); // apaga o texto escrito no Display
```

```
lcd.setCursor(0,0); // coluna 0, linha 0
```

```
lcd.print(" Arduino Omega"); // Insira seu texto Aqui
```

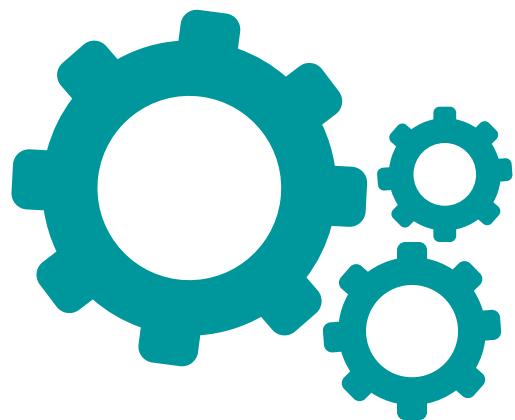
```
delay(5000); // Espera de 5 segundos
```

```
lcd.clear(); // apaga o texto escrito no Display
```

```
}
```

Baixe o código do projeto [Clicando Aqui !](#)

Quer ver o funcionamento ?? [Clique aqui !!](#)



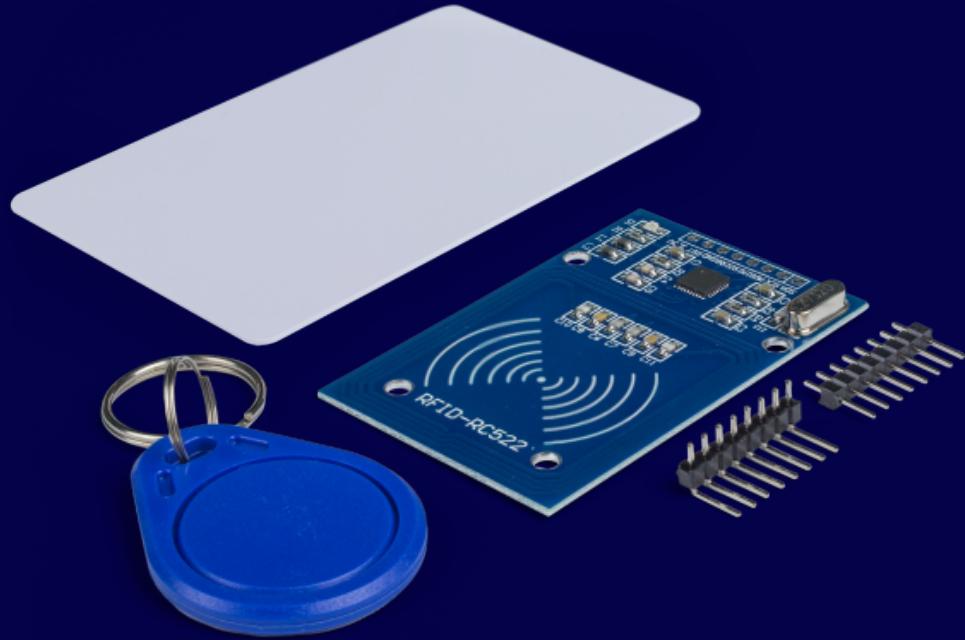
12

Controle de Acesso com RFID

Nesse projeto vamos desenvolver um **controle de acesso** utilizando o **Módulo RFID** e outros componentes que já conhecemos, como o **Buzzer**, o **LED**, o **Display** e outros.

Módulo RFID

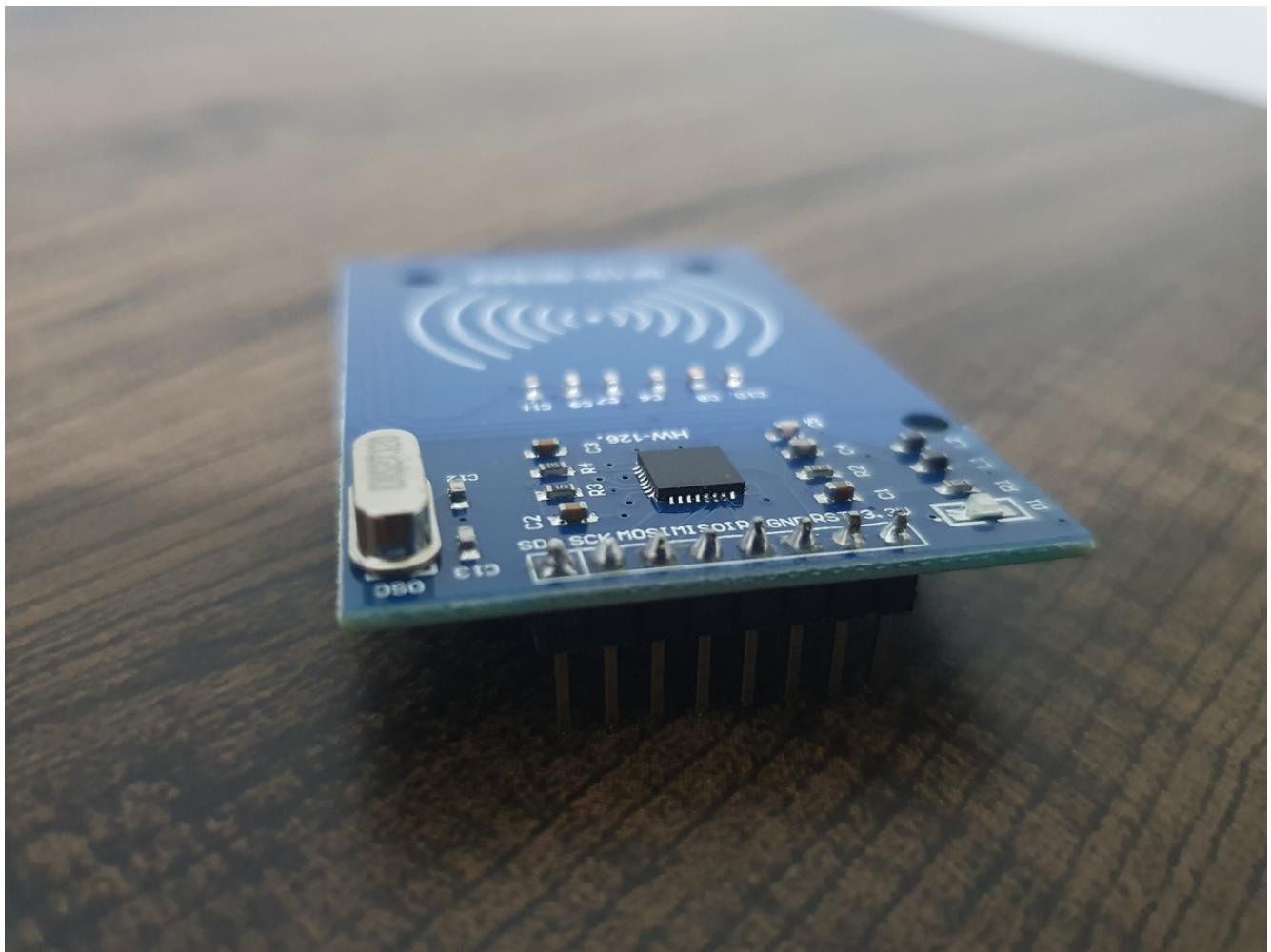
Este **Kit Módulo Leitor RFID** baseado no chip **MFRC522** da empresa NXP é altamente utilizado em comunicação sem contato a uma frequência de 13,56MHz. Este chip, de baixo consumo e pequeno tamanho, permite sem contato ler e escrever em cartões que seguem o padrão **Mifare**, muito usado no mercado.



Possui ferramentas que é preciso para projetos de controle de acesso ou sistemas de segurança. As tags (ou etiquetas) **RFID**, podem conter **vários dados** sobre o proprietário do cartão, como nome e endereço e, no caso de produtos, informações sobre procedência e data de validade, entre outros exemplos.

Soldando os Pinos

É necessário soldar a barra de pinos no [Módulo Leitor RFID-RC522](#). Como vamos utilizar a [Protoboard](#), é recomendado soldar a barras de pinos de 180 graus. Para realizar a solda é preciso utilizar um [ferro de solda](#) e [estanho](#).

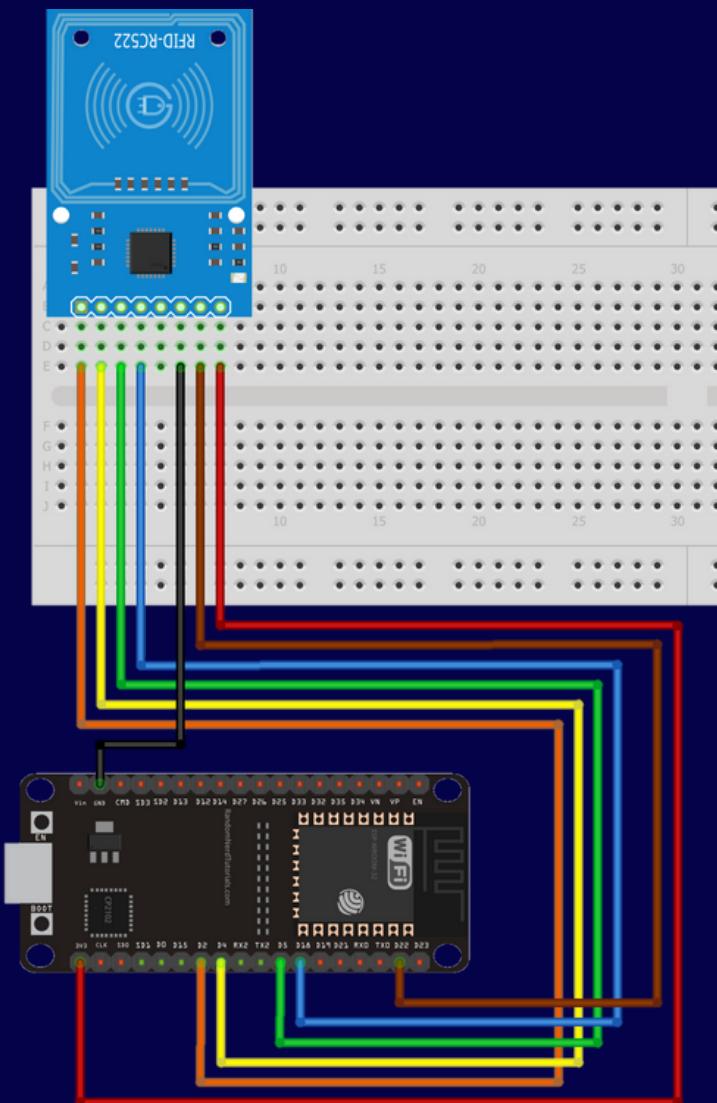


Caso você seja [**menor de idade**](#), peça ajuda a um [**adulto**](#) para a realização da solda dos fios nos motores !!



Círcuito

Primeiro vamos realizar as **conexões necessárias** no **Módulo RFID**.



O **Pino SDA** deve estar conectado ao **Pino Digital D2**.

O **Pino SCK** deve estar conectado ao **Pino Digital D4**.

O **Pino MOSI** deve estar conectado ao **Pino Digital D5**.

O **Pino MISO** deve estar conectado ao **Pino Digital D18**.

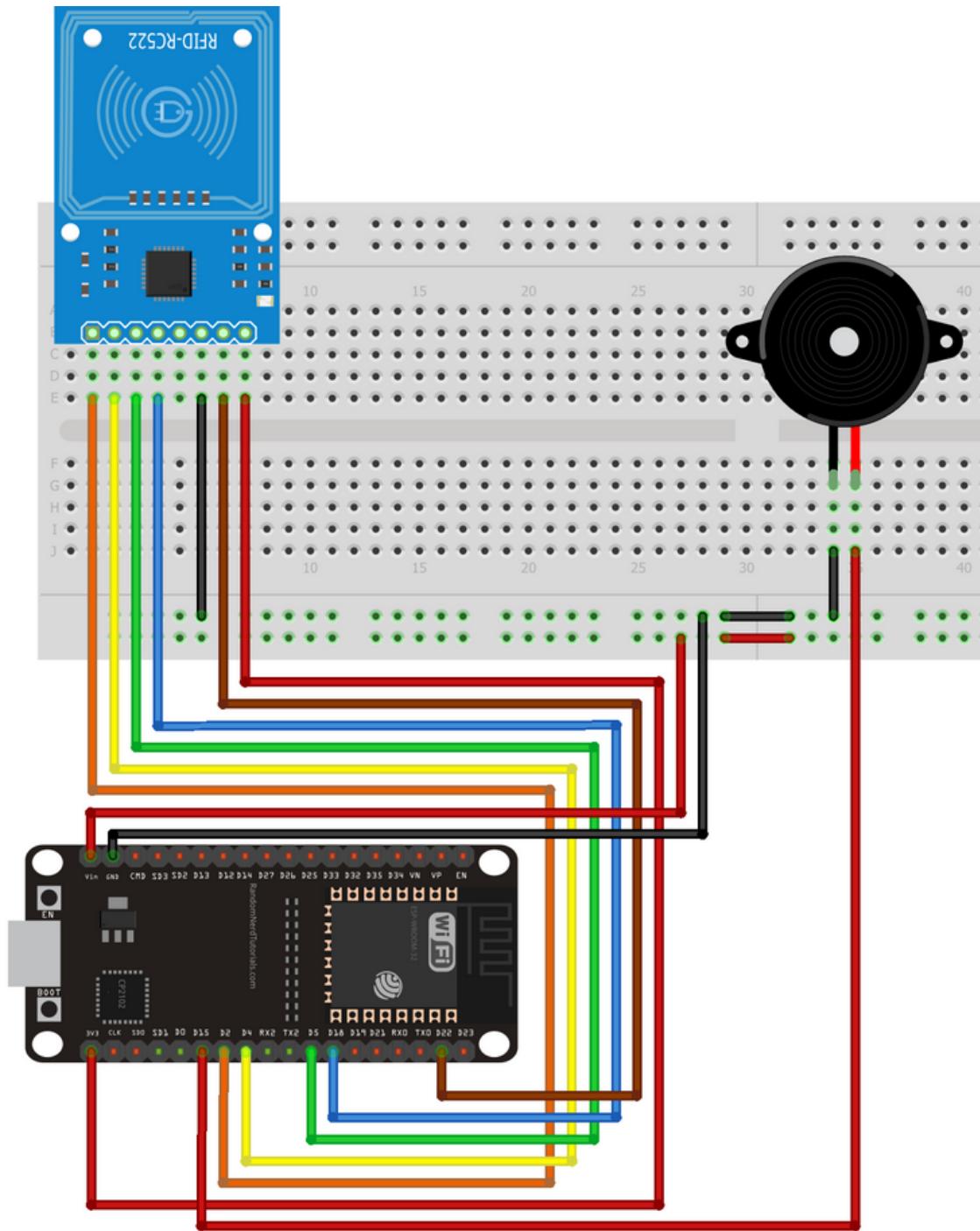
O Pino IRQ não deve ser conectado a nada.

O **Pino GND** deve estar conectado ao **Pino GND**.

O **Pino RST** deve estar conectado ao **Pino Digital D22**.

O **Pino 3.3V** deve estar conectado ao **Pino 3V**.

Agora vamos adicionar o **Buzzer** em nosso circuito.

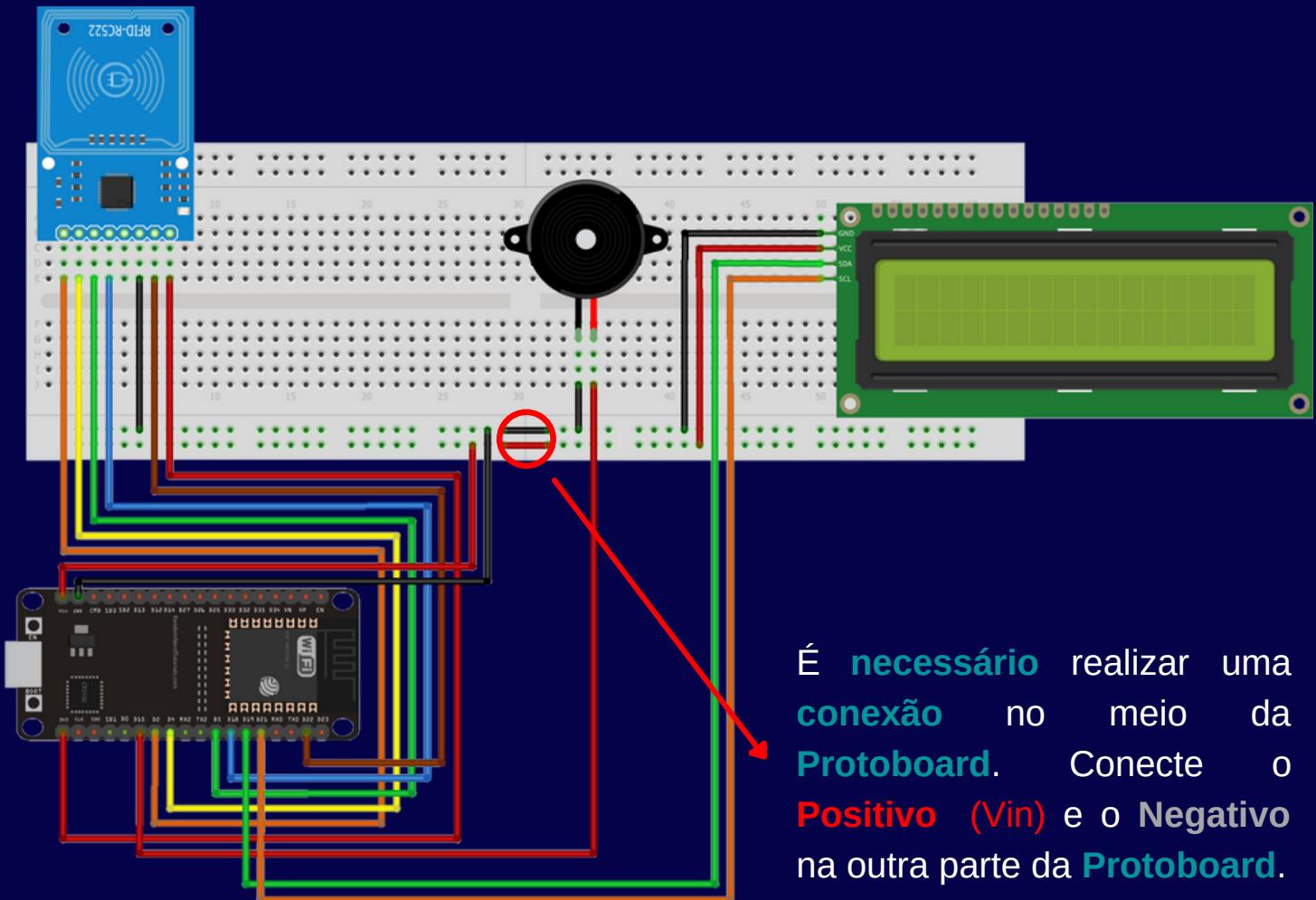


É necessário conectar o **Pino G** (GND) e o **Pino Vin** do **NodeMCU** na **Protoboard**.

O **Maior terminal** do **Buzzer** deve ser conectado no **Pino Digital D8** e o **menor** no **GND** da **Protoboard**.

Círcuito Final

Por fim, vamos incluir no nosso projeto o **Display LCD 16x2** com o **Módulo I2C**.



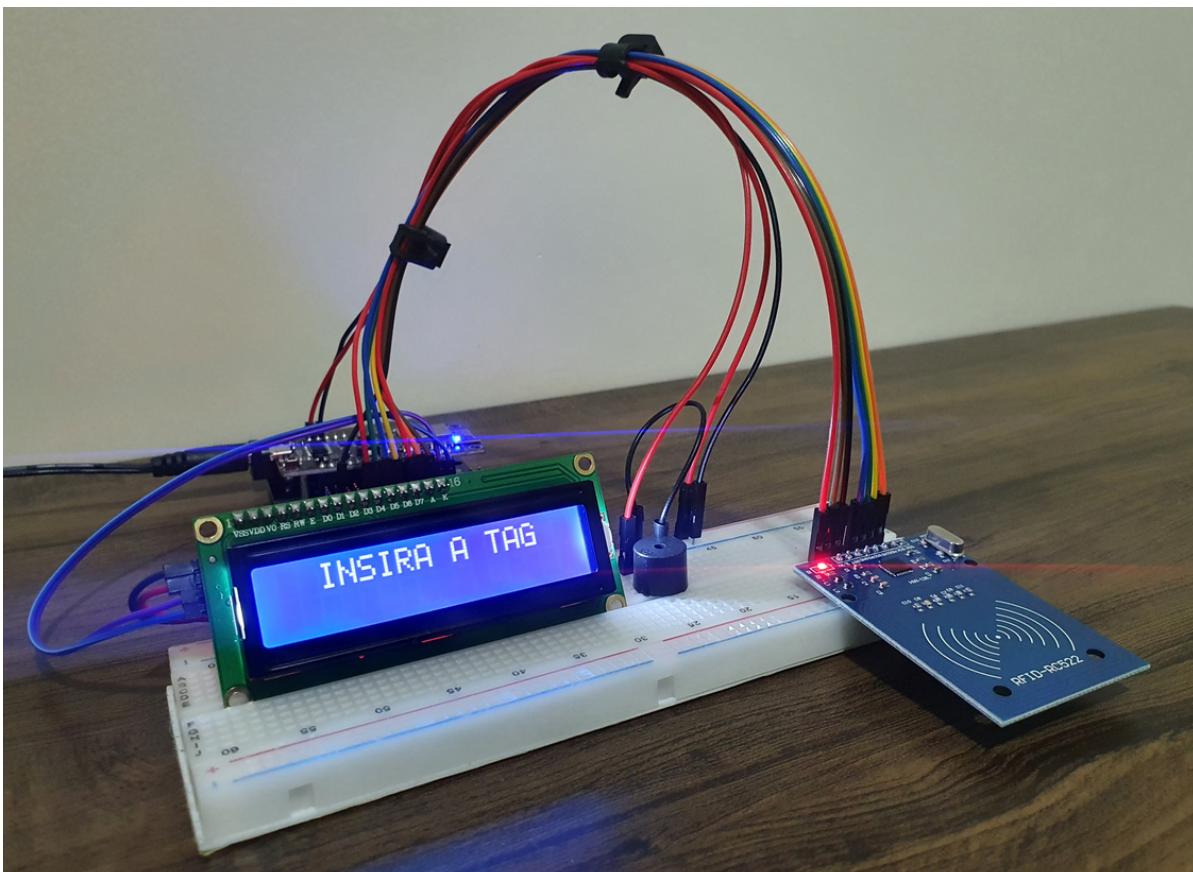
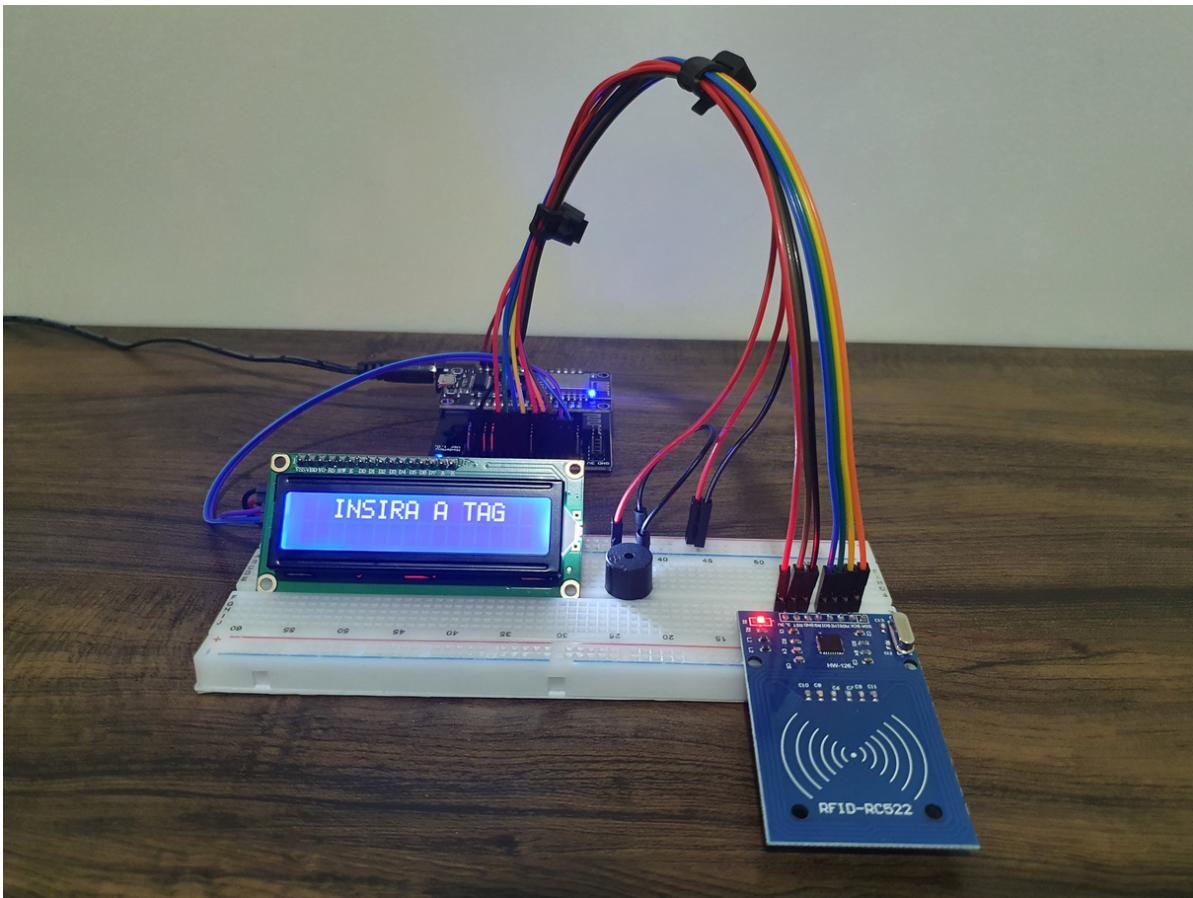
O Pino GND do **Display** deve ser conectado ao **GND do circuito**.

O Pino **VCC** no **positivo** do circuito (**Vin**).

O Pino **SDA** deve estar conectado no **Pino Digital D19**.

O Pino **SCL** deve estar conectado no Pino **Digital D21**.

Círcito Na Prática



Capturando as UIDs

Precisamos saber a **UID** da **Tag** para cadastrarmos na programação de **validação de Tags**.

Para isso, vamos utilizar a programação abaixo:

OBS: É necessário instalar a **biblioteca MFRC522**, [Clique Aqui](#) para baixar.

```
// Bibliotecas Necessárias

#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 2 //Pino SDA
#define RST_PIN 22 //Pino de Reset

MFRC522 rfid(SS_PIN, RST_PIN);

void setup () {

    Serial.begin(9600); //Iniciando a Serial

    SPI.begin();
    rfid.PCD_Init();
}

void loop () {
    if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
        return;

    // Código Responsável por gerar a Tag

    String strID = "";
    for (byte i = 0; i < 4; i++) {
```

```

strID +=  

    (rfid.uid.uidByte[i] < 0x10 ? "0" : "") +  

    String(rfid.uid.uidByte[i], HEX) +  

    (i!=3 ? ":" : "");  

}  

strID.toUpperCase();  
  

Serial.print("Identificador da tag: "); //Imprime texto na Serial  

Serial.println(strID); //Imprime o UID da Tag  
  

rfid.PICC_HaltA();  

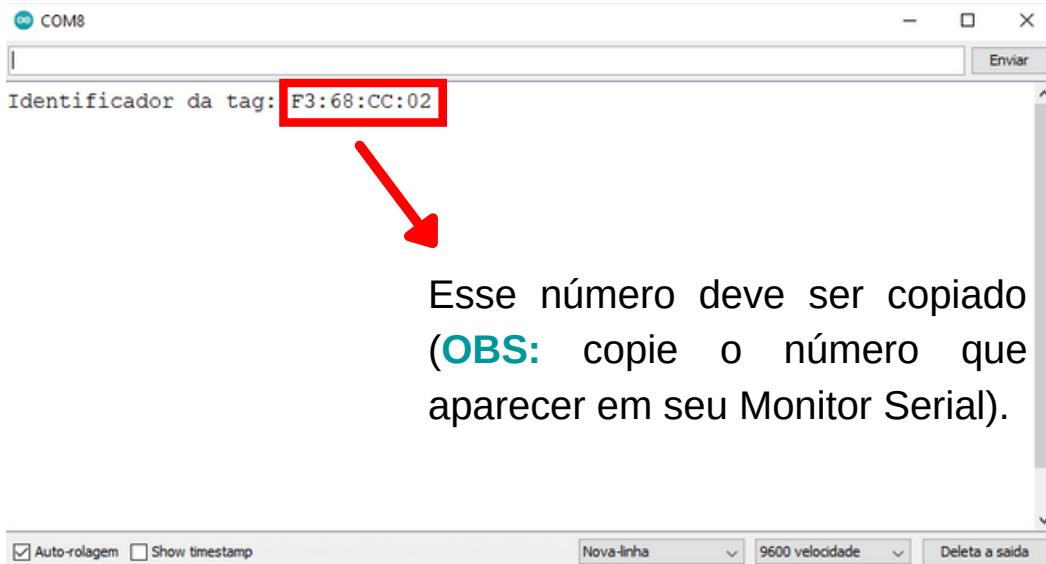
rfid.PCD_StopCrypto1();
}

```

Baixe o código do projeto [Clicando Aqui !](#)

Envie a programação para o **NodeMCU**, deixe ele conectado via **cabo USB** ao computador e pressione as teclas **Ctrl + Shift + m** para abrir o **Monitor Serial** na **IDE do Arduino**.

Após abrir o **Monitor Serial** aproxime as **tags** no **Módulo RFID** e aparecerá no Monitor Serial a **UID**, escolha uma e **copie**, vamos usar na **programação final**.



Esse número deve ser copiado
(OBS: copie o número que
aparecer em seu Monitor Serial).

Programação

// Incluindo Bibliotecas Necessárias

```
#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C Display = LiquidCrystal_I2C(0x27, 16, 2);
```

```
#define SS_PIN D4 //Pino SDA no Pino Digital D4
#define RST_PIN D3 //Pino RST Pino Digital D4
```

```
MFRC522 rfid(SS_PIN, RST_PIN);
```

```
int Buzzer = 8;
```

```
void setup(){
```

```
lcd.begin(16,2); // linhas e colunas do display
lcd.init();
lcd.backlight();
```

```
// Iniciando as bibliotecas
```

```
Wire.begin();
SPI.begin();
rfid.PCD_Init();
```

```
// Definindo o Buzzer como Saída Digital
```

```
pinMode(Buzzer, OUTPUT);
digitalWrite(Buzzer, LOW);
```

```

// Inicia o Display

Lcd.setCursor(0,0); // coluna 0, linha 0
Lcd.print(" INSIRA A TAG"); // Escreve no Display

}

void loop() {
    leituraRfid();
}

//Função de validação da TAG
void leituraRfid(){
    if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial())
        return;

    String strID = "";
    for (byte i = 0; i < 4; i++) {
        strID += (rfid.uid.uidByte[i] < 0x10 ? "0" : "") +
            String(rfid.uid.uidByte[i], HEX) +
            (i!=3 ? ":" : "");
    }
    strID.toUpperCase();

    // Insira a UID da tag capturada anteriormente

    if (strID.indexOf("Insira a UID da tag capturada anteriormente") >= 0) { // Se a
TAG for correta

        int qtd_bips = 2; //definindo a quantidade de bips
        for(int j=0; j<qtd_bips; j++){

        // Ligando o Buzzer com uma frequência de 1500 hz

```

```
lcd.clear();
lcd.setCursor(0,0); // coluna 0, linha 0
lcd.print("ACESSO LIBERADO"); // Escreve no Display

tone(Buzzer,1500);
delay(100);
noTone(Buzzer);
delay(100);
}

delay(2000);

lcd.clear(); // Limpa Display
lcd.setCursor(0,0); // coluna 0, linha 0

lcd.print(" INSIRA A TAG"); // Escreve no Display

}else{ //SENÃO, FAZ (CASO A TAG LIDA NÃO SEJÁ VÁLIDA)
  int qtd_bips = 1; //definindo a quantidade de bips
  for(int j=0; j<qtd_bips; j++){

//Ligando o Buzzer com uma frequênciade 500 hz

lcd.clear(); // Limpa Display
lcd.setCursor(0,0); // coluna 0, linha 0
lcd.print(" ACESSO NEGADO "); // Escreve no Display

tone(Buzzer,500);
delay(500);

// Desligando o Buzzer

noTone(Buzzer);
delay(1000);
```

```

lcd.clear(); // Limpa Display

lcd.setCursor(0,0); // coluna 0, linha 0
lcd.print(" INSIRA A TAG");

}

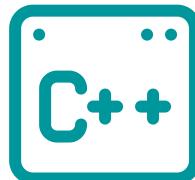
}

rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();
}

```

Baixe o código do projeto [Clicando Aqui !](#)

Quer ver o funcionamento ?? [Clique aqui !!](#)



Sugestões

Nesse projeto você também pode adicionar uma **mini trava elétrica** e incluir um código para que uma **notificação** seja enviada para o **celular** caso alguém tente acessar com uma **Tag errada**.

Seu funcionamento será da seguinte maneira: quando a **Tag** for validada a **trava é acionada**, caso contrário a trava não é acionada e um **SMS** ou **e-mail** é enviado para o celular alertando sobre o ocorrido !!!

O que você achou dessa sugestão, Gostou? Acha que devemos criar esse projeto ?

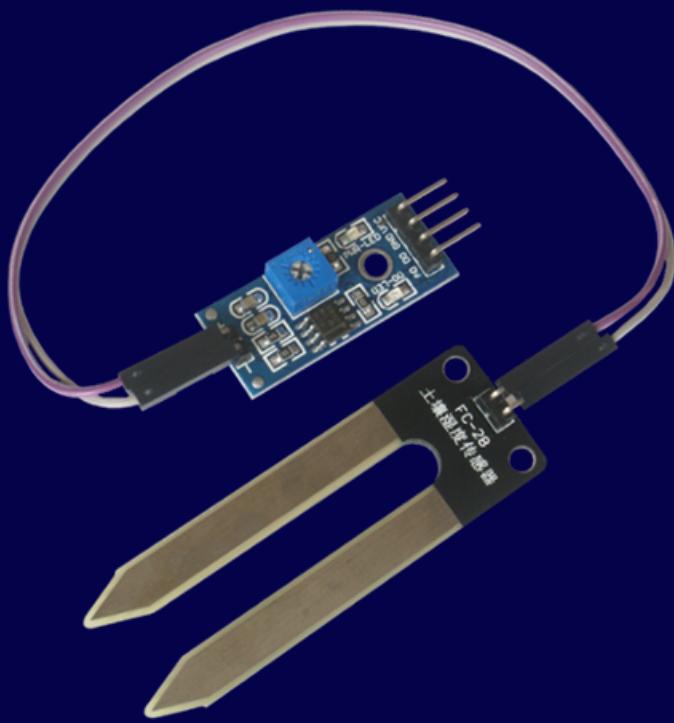
Caso você queira ver um tutorial dessa sugestão envie seu e-mail para [contato@arduinoomega.com](mailto: contato@arduinoomega.com), Se recebermos **muitos e-mails** vamos fazer esse projeto !!



Chegamos no último projeto do nosso E-book, vamos desenvolver um sistema de irrigação automático para plantas, para isso será necessário utilizarmos um **Módulo Sensor de Umidade de Solo** e uma **Mini Bomba de água**.

Sensor de Umidade de Solo

Este **Sensor de Umidade do Solo Higrômetro** foi feito para detectar as variações de umidade no solo, sendo que quando o solo está seco a saída do sensor fica em estado alto (**HIGH**), e quando úmido em estado baixo (**LOW**). Sua tensão de operação é de **3,3** a **5V**.



O limite entre **seco** e **úmido** pode ser ajustado através do potenciômetro presente no sensor que regulará a **saída digital D0**. Contudo para ter uma resolução melhor é possível utilizar a saída analógica A0 e conectar a um conversor AD, como presente no **Arduino** por exemplo.

Mini Bomba de água

A **Mini Bomba de Água RS385** foi criada especialmente para o desenvolvimento de projetos de prototipagem, incluindo automação residencial e protótipos robóticos baseados em plataformas **microcontroladoras** como por exemplo o **ESP-32** e o **Arduino**.

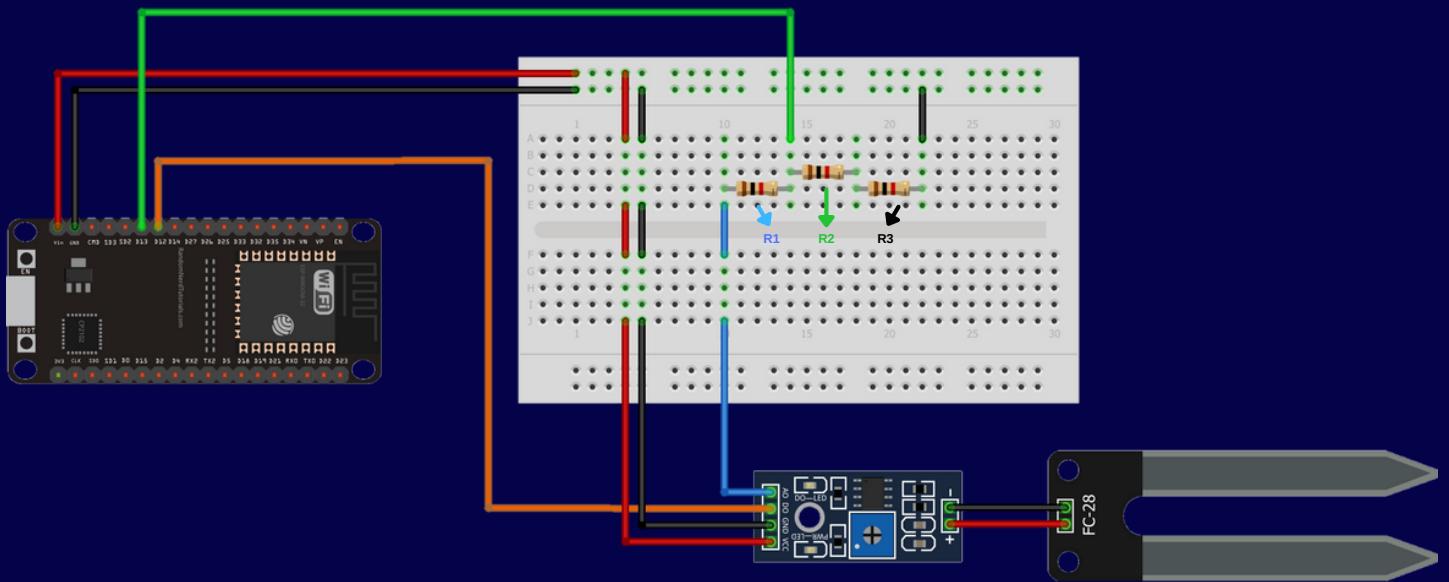


Com um motor de tamanho adequado a **mini bomba** é capaz de impulsionar entre **1500 ml** a **2000 ml** por minuto, sendo destacada pela sua eficiência e precisão durante sua execução em conjunto com o **ESP-32**, por exemplo.

É frequentemente utilizada em desenvolvimento de **carrinhos** ou **robôs bombeiros**, **robôs hidráulicos**, **irrigadores automáticos** no caso de automação residencial, enfim sua criatividade que dará a aplicação final para este incrível acessório.

A **mini bomba** opera com tensão entre **9V** a **15V** e permite elevação máxima de até **3 metros** e altura de aspiração de até **2 metros**.

Círcito



Primeiro é necessário conectar dois jumper no sensor e conectar na **placa de controle**, no **negativo** e **positivo**.

O pino **GND** da **placa de controle** deve ser conectado no pino **GND** do microcontrolador.

O pino **VCC** da **placa de controle** no pino **Vin** do **ESP-32**.

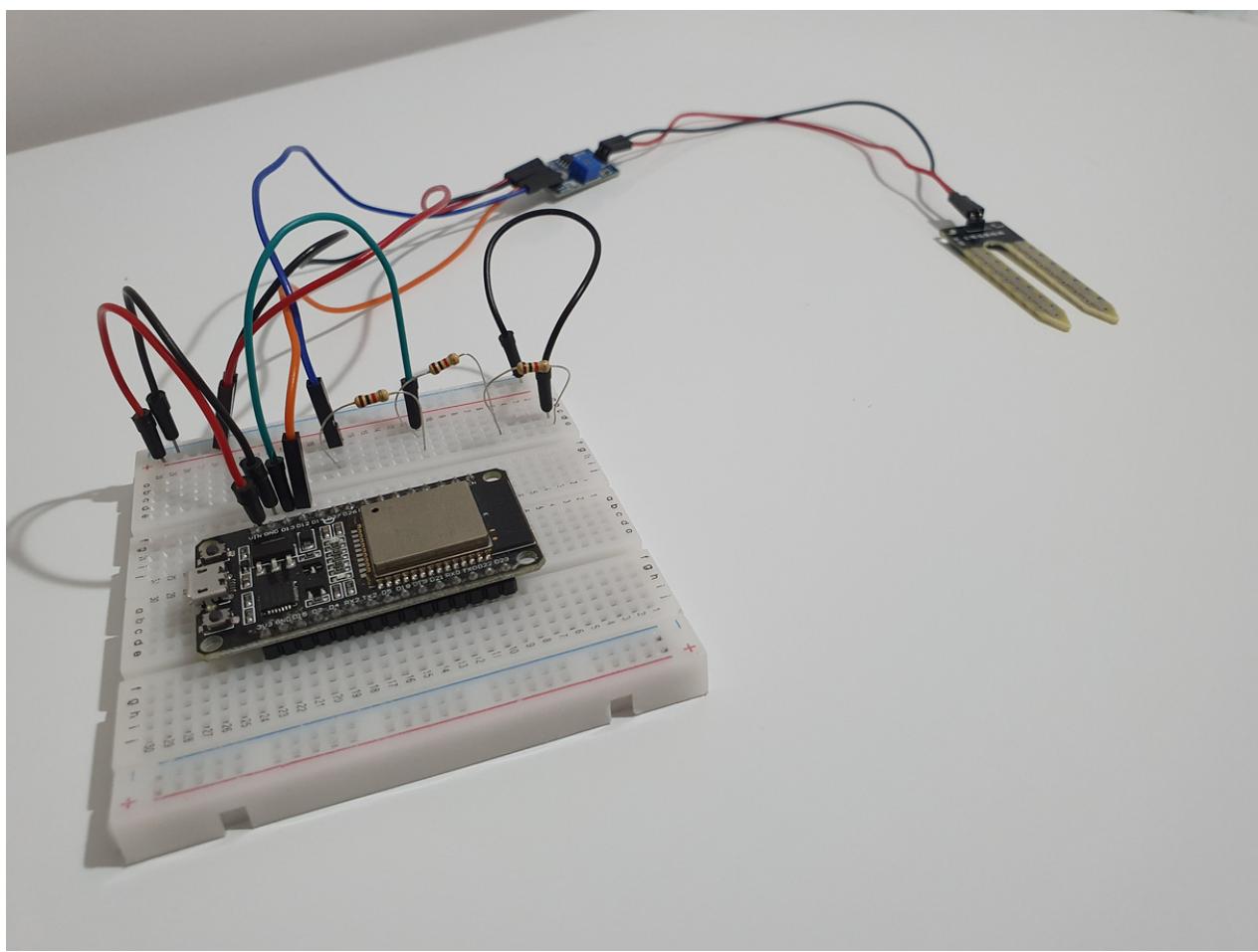
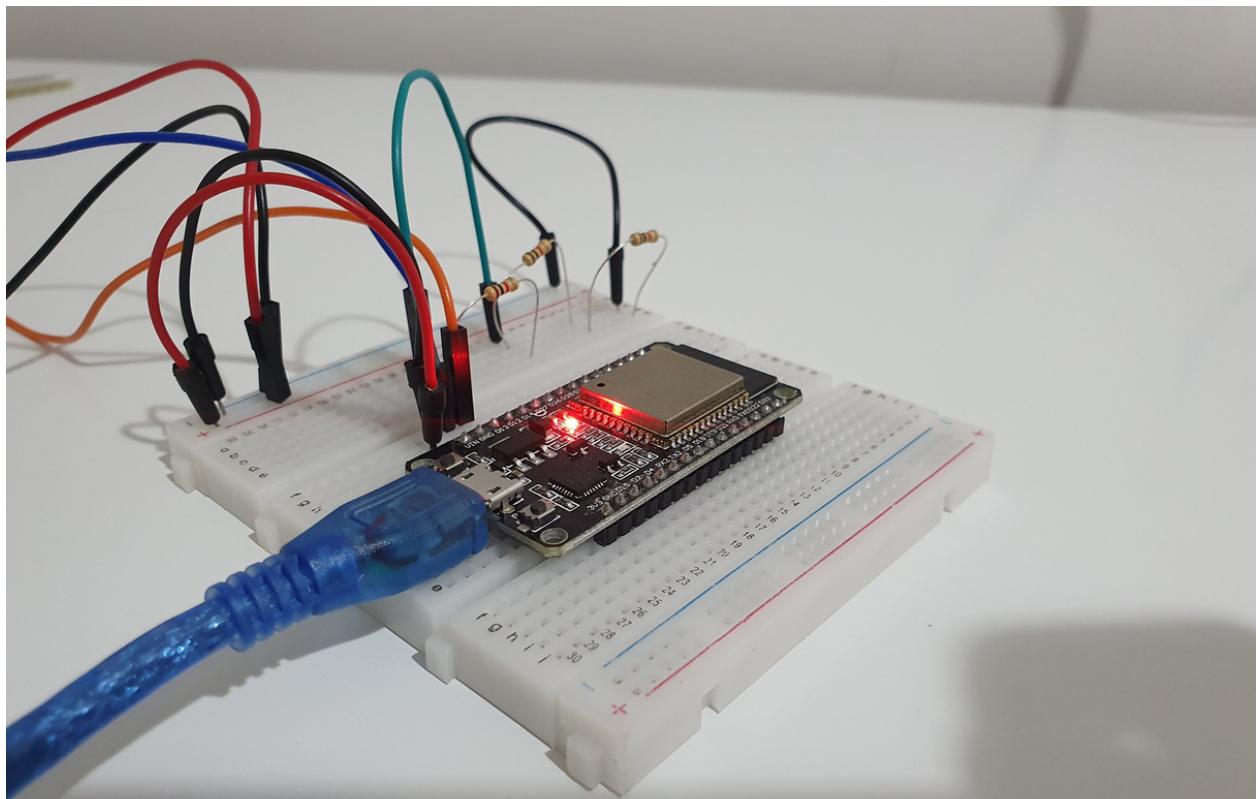
O **Pino D0** do sensor deve estar conectado ao **Pino D12** do **ESP-32**.

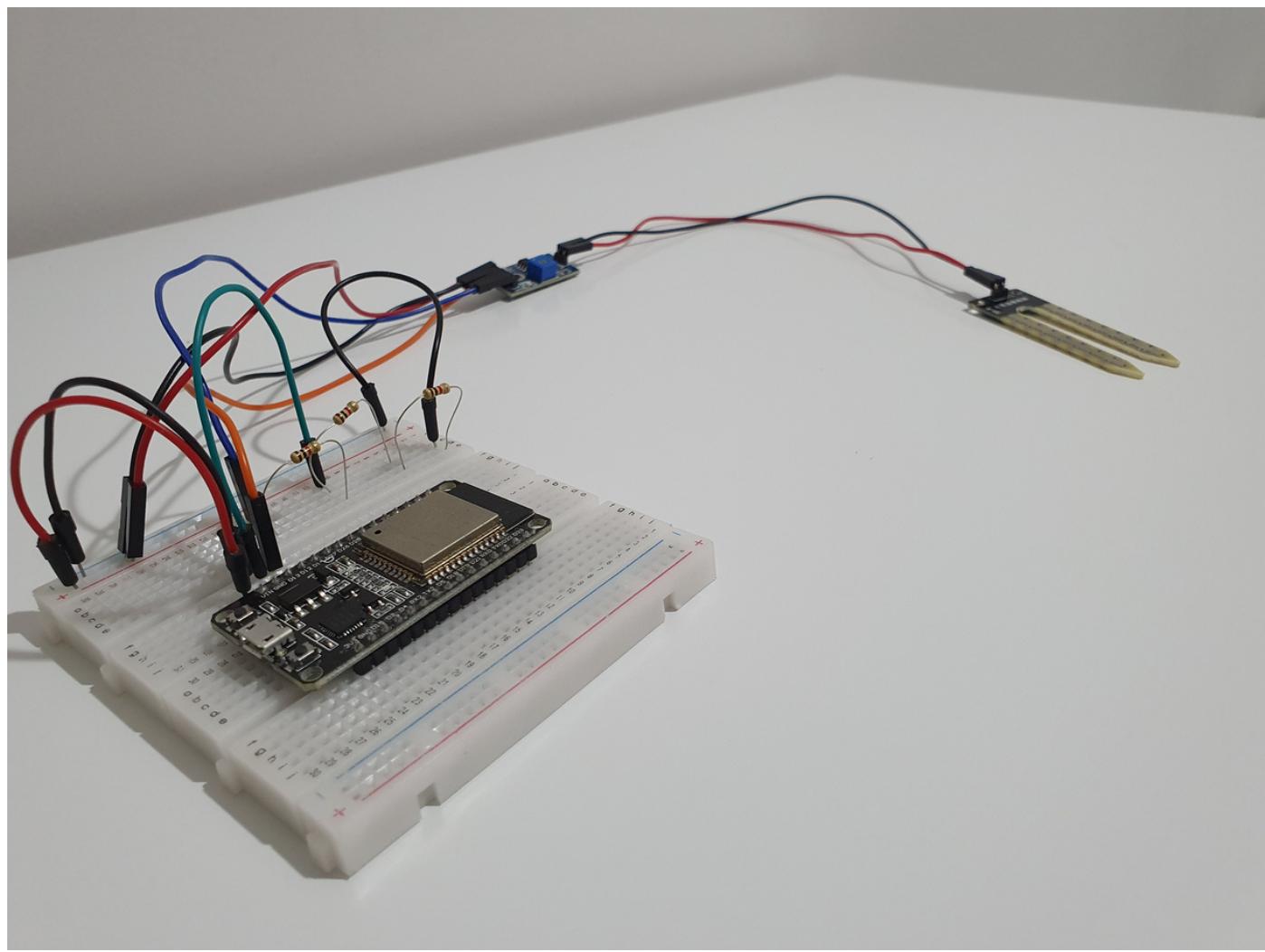
O **Pino A0** do sensor fornece uma tensão de **5V**, como o **ESP-32** suporta somente **3.3V** é necessário fazer um **regulador de tensão**, para isso vamos utilizar **3 resistores** de **1KΩ** em **série**, dessa forma a tensão de saída será de **3.3V**.

O **Pino A0** do **sensor** deve ser conectado ao primeiro **resistor (R1)**.

O segundo **resistor (R2)** precisa estar ligado ao pino **D13 do ESP-32**.

O terceiro **resistor (R3)** deve ser ligado ao **GND** do **círcito**.





Alimentação

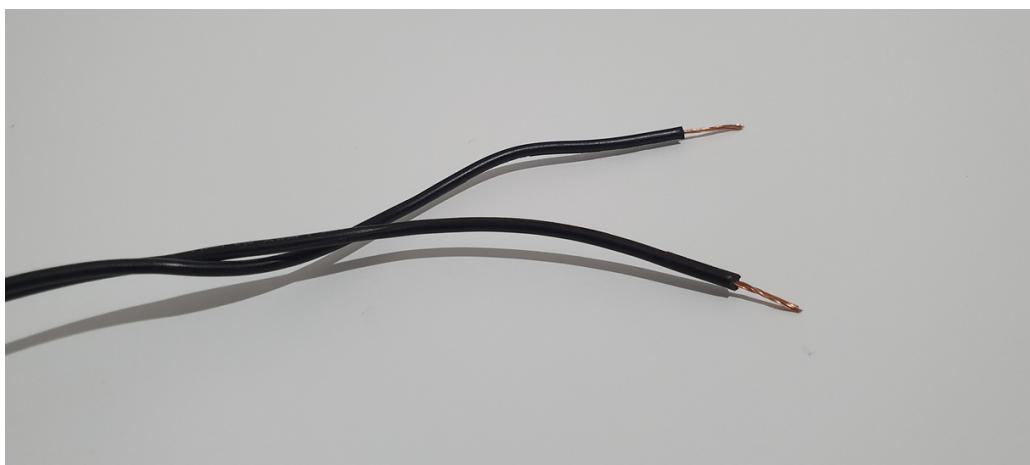
Para alimentarmos a **bomba**, será necessário o uso de uma **fonte de alimentação** de **9V**. É preciso realizar uma adaptação no conector da fonte.



Com um **alicate de corte**, remova o conector da fonte

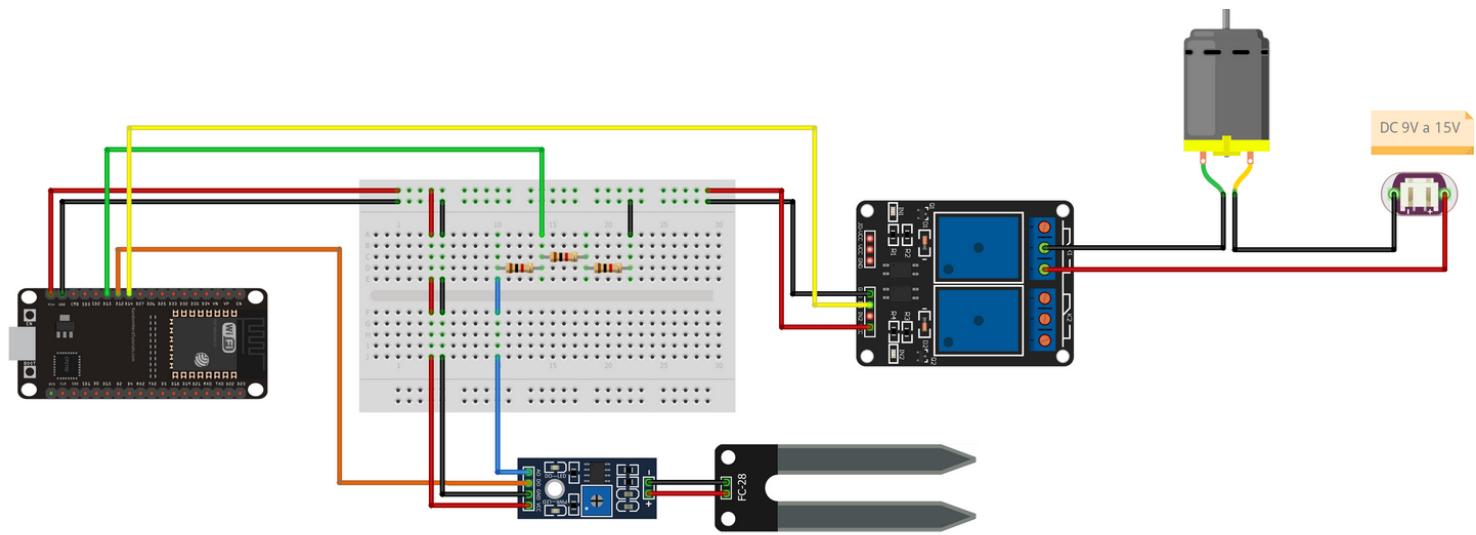


Decape a ponta do fio, expondo o cobre



Círculo Completo

Vamos adicionar o **Relé** e a **Mini Bomba** no circuito.



É necessário conectar **Vin** do circuito no pino **VCC** do Relé.

Conecte o GND do circuito no pino **GND** do **Módulo Relé**.

O **Pino Digital D14** do **ESP32** deve estar no **IN1** do Relé.

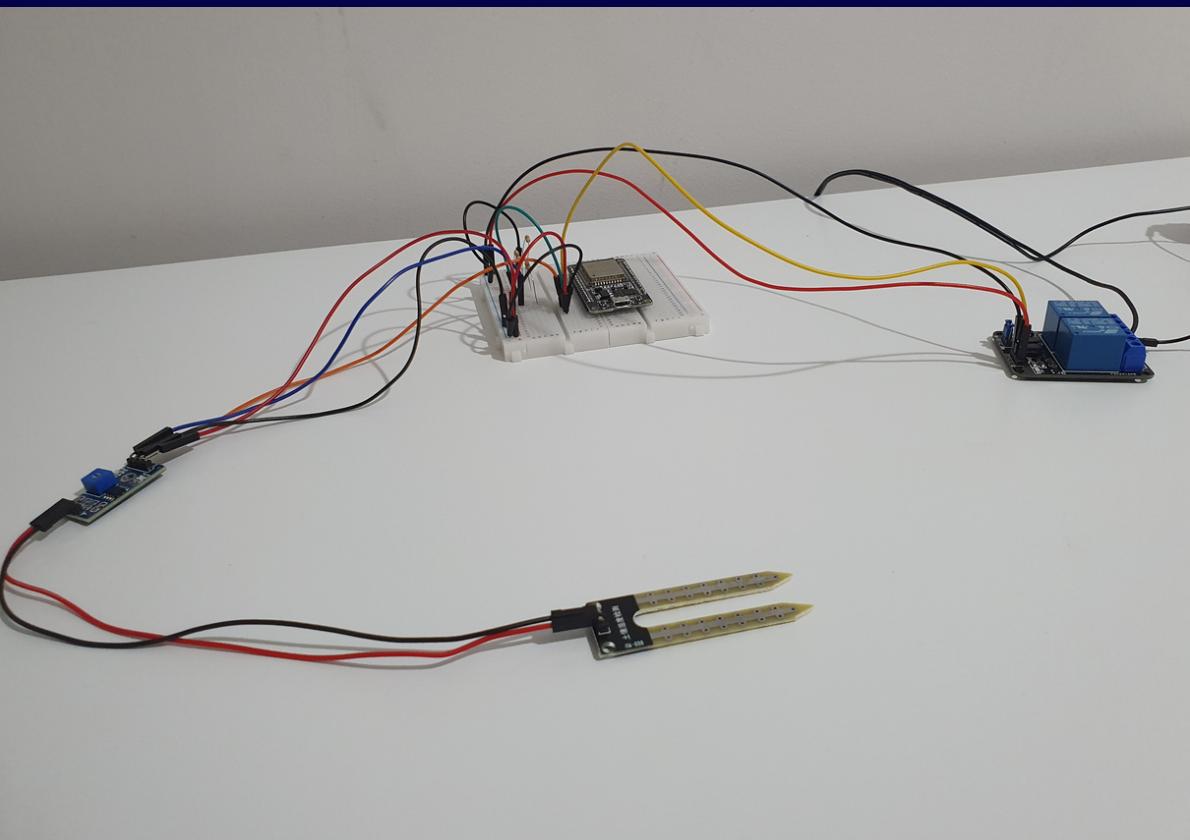
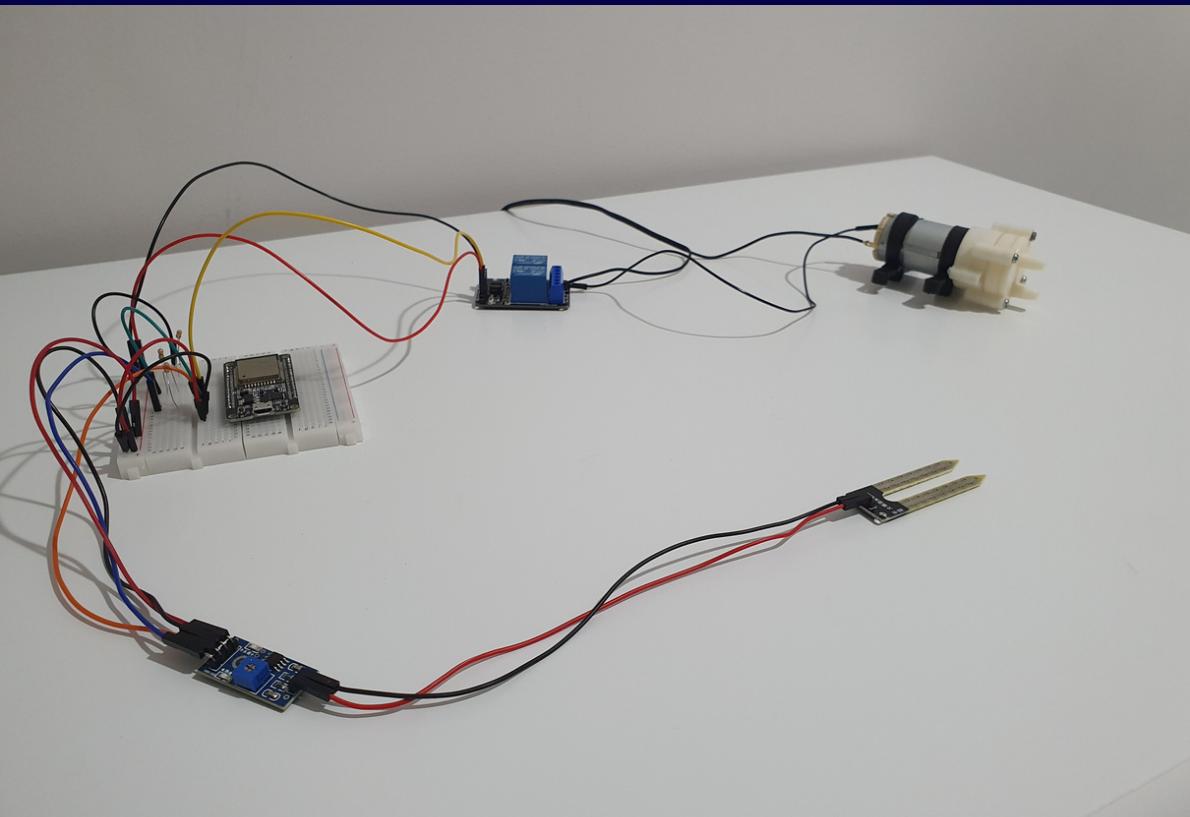
Um terminal da **Mini Bomba** de estar conectado no **contato aberto do relé K1** e outro em uma fonte de **9 a 15V**.

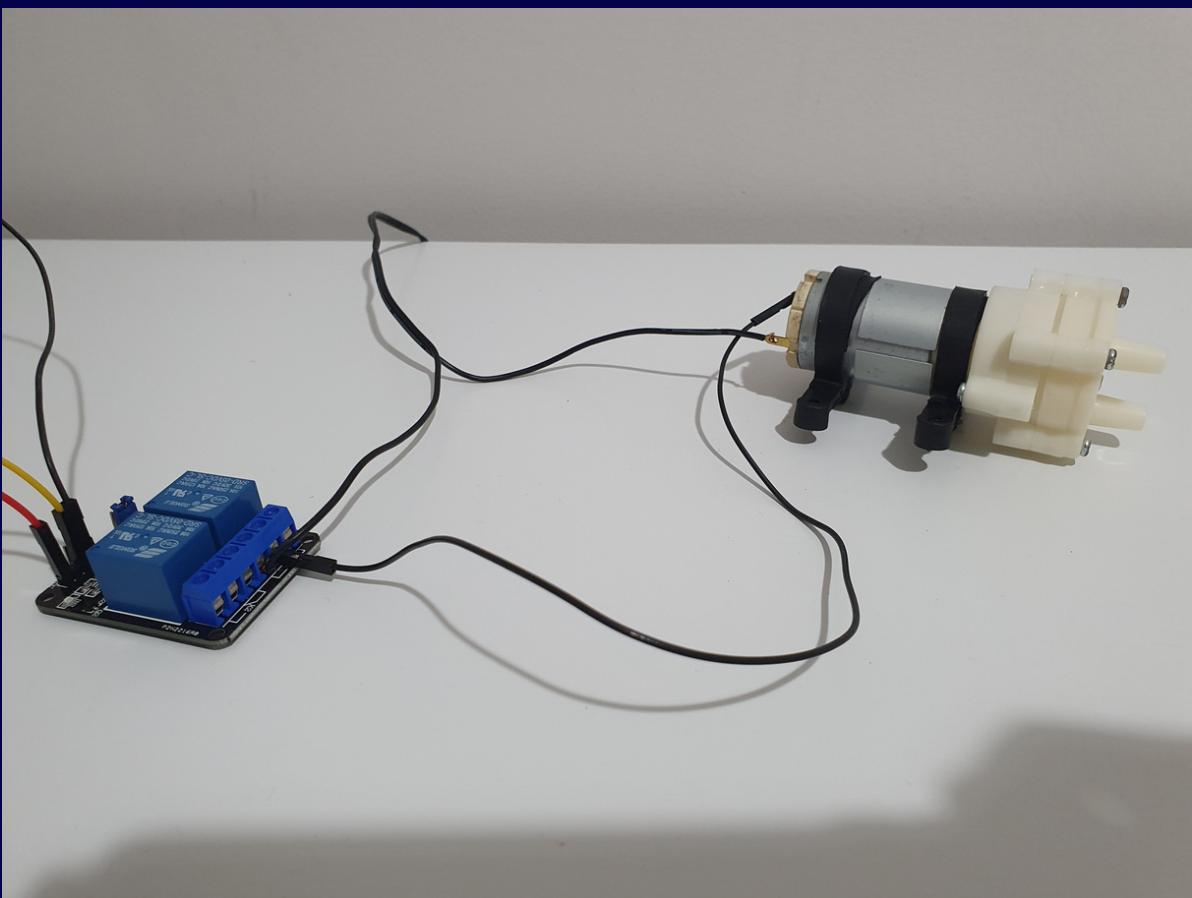
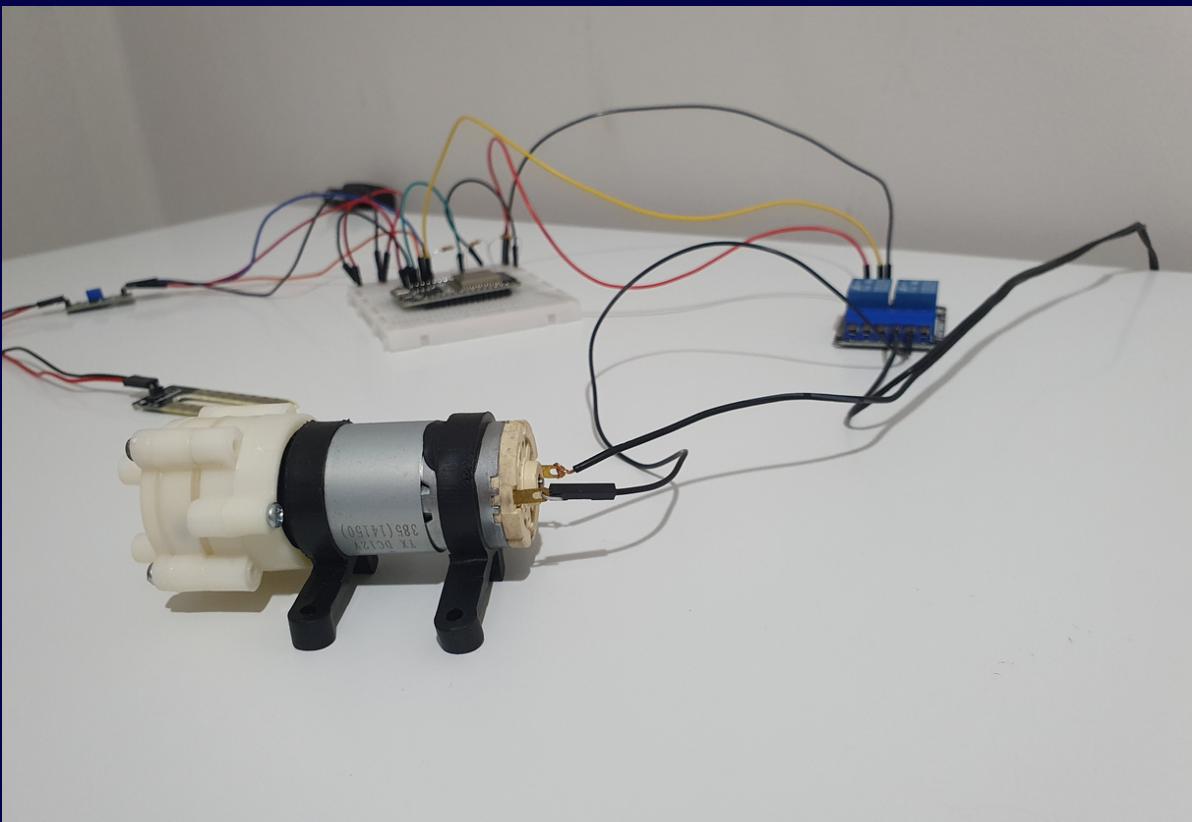
O **contato fechado do relé K1** deve estar conectado diretamente na fonte de **9 a 15V**.

Nesse projeto, também vamos utilizar o aplicativo **Blynk**, para nos avisar quando a umidade da terra está baixa e a bomba será ligada. **Você pode utilizar o mesmo projeto feito no sensor de chuva !!**

OBS: Alimente o **ESP-32** via **USB**.

Círcuito na Prática





Programação

```
int PinoAnalogico = 13; // Define o pino 13 como Pino Analógico do sensor
int PinoDigital = 12; // Define pino D14 como Pino Digital do Sensor

int Rele = 14; // Pino Digital 14 como Relé

int EstadoSensor = 0;
int UltimoEstSensor = 0;

int ValAnalogIn; // Valor analógico no código

void setup() {

Serial.begin(9600);
pinMode(Rele, OUTPUT); // Declara o Rele como Saída Digital
pinMode(PinoDigital, INPUT);
}

void loop() {

ValAnalogIn = analogRead(PinoAnalogico);
int Porcento = map(ValAnalogIn, 1023, 0, 0, 100); // Traforma o valor analógico
em porcentagem

Serial.print("Umidade: "); // Imprime o símbolo no valor
Serial.print(Porcento); // Imprime o valor em Porcentagem no monitor Serial
Serial.println("%");

if (Porcento <= 76) { // Se a porcentagem for menor ou igual à 76%. OBS: Você
pode alterar essa porcentagem

Serial.println("Irrigando Planta"); // Imprime no monitor serial
```

```
digitalWrite(Rele, LOW); // Aciona Relé  
  
}else { // Caso contrario  
  
Serial.println("Planta Irrigada"); // Imprime a no monitor serial  
digitalWrite(Rele, HIGH); // Desliga Relé  
  
delay (1000);  
}  
}
```

Baixe o código do projeto [Clicando Aqui !](#)

Quer ver o funcionamento ?? [Clique aqui !!](#)

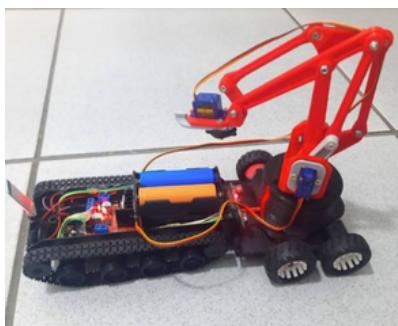


14 Finalização

Nessa apostila abordamos a **introdução a Internet das coisas** para iniciantes utilizando o **ESP-32**, com diversos projetos, nosso intuito foi desvendar um pouco sobre o mundo da Internet das Coisas e te instigar a conhecer cada vez mais sobre essa área que a cada dia vem crescendo.

Caso você tenha interesse em conhecer mais projetos de eletrônica usando o **ESP 32**, a **Eletrônica Ômega** disponibiliza vários outros tutoriais de forma gratuita em nosso blog, temos vários outros tutoriais para você mergulhar no mundo da IoT!

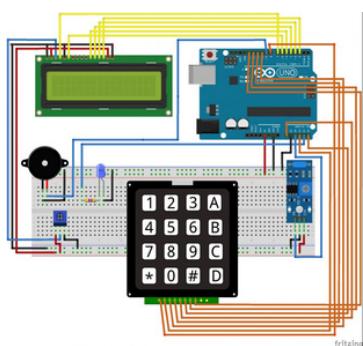
Veja abaixo exemplos de alguns tutoriais que você encontra no blog:



[Tanque Reboque Bluetooth](#)



[Braço Robótico Controlado por voz](#)



[Sistema de Alarme Codificado](#)

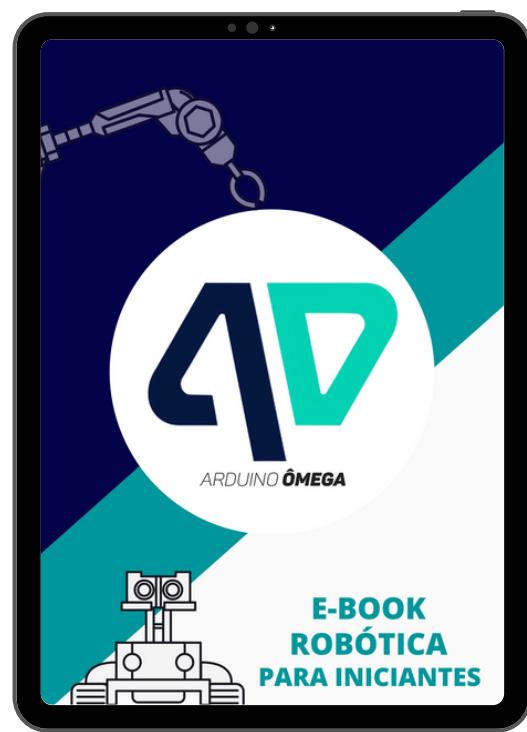


[Sistema de Acesso com RF ID](#)

E-book Robótica

Venha aprender robótica com a gente! Com o [Kit Arduino Robôs](#) e o nosso [E-book Robótica para Iniciantes](#) você irá aprender sobre [Arduino](#), eletrônica, física, programação e robótica! Bora lá?

Elaboramos o [E-book](#) e o [Kit Arduino Robôs](#), para serem totalmente amigáveis aos iniciantes, todo o material descrito no [E-book](#) está contido neste Kit, todos os projetos descritos no [E-book](#) são possíveis de montar com os componentes deste kit!



Em nosso [E-book](#) há o detalhamento de **4 projetos de robôs**: o [Veículo Autônomo](#), o [Veículo Controlado por App](#), [Braço Robótico Controlado por Joystick](#) e [Braço Robótico Controlado por App](#)! Você irá encontrar o código fonte completo, com explicação de todos os pontos do código, diagrama elétrico mostrando como fazer a ligação de cada componentes, detalhamentos dos principais problemas que podem ocorrer na montagem, entre várias dicas.

Quem escreveu o E-book ?



Rangel Gabriel

Formado em **Eletroeletrônica** pelo **SENAI Shunji Nishimura**, tem experiência com eletrônica e microcontroladores.

E-mail: rangelarena@gmail.com



Eletrônica Ômega



A **Eletrônica Ômega** é uma loja virtual sediada em BH/MG, especializada em **Arduino** e **componentes eletrônicos**.

Temos a missão de **promover transformação cultural e social através da educação**, inserindo o maior número possível de pessoas no movimento maker, levando acesso a tecnologia para todos os interessados, e **mostrando que é possível desenvolver suas ideias através da tecnologia**.

Caso queria falar com a gente não deixe de mandar seu email para [contato@arduinoomega.com](mailto: contato@arduinoomega.com).

Redes Sociais:



ARDUINO ÔMEGA
BLOG

Referências

Tensão:

<https://www.todamateria.com.br/tensao-eletrica/> - acesso em 16/06/2021

<https://www.mundodaeletrica.com.br/tensao-eletrica-x-voltagem/> - acesso em 16/06/2021.

Corrente:

<https://mundoeducacao.uol.com.br/fisica/corrente-eletrica.htm> - acesso em 16/06/2021.

<https://brasilescola.uol.com.br/fisica/corrente-eletrica.htm> - acesso em 16/06/2021.

<https://www.mundodaeletrica.com.br/o-que-e-corrente-eletrica/> - acesso em 16/06/2021.

LED:

<https://athoselectronics.com/o-que-e-led-diodo-emissor-luz/#:~:text=O%20LED%C3%A9%20um%20Diodo,energia%20el%C3%A9trica%20em%20energia%20luminosa.&text=Como%20se%20trata%20de%20um,corrente%20el%C3%A9trica%20em%20um%20sentido.> - acesso em 25/06/2021.

Resistores:

<https://brasilescola.uol.com.br/fisica/resistores.htm#:~:text=Resistores%20s%C3%A3o%20dispositivos%20usados%20para,esses%20s%C3%A3o%20conhecidos%20como%20diel%C3%A9tricos.> - acesso em 25/06/2021.

IoT

<https://inforchannel.com.br/2021/04/27/internet-das-coisas-e-protecao-de-dados/>
- acesso em 10/08/2021.

DHT11

<https://www.baudaelectronica.com.br/sensor-de-umidade-e-temperatura-dht11.html> - acesso em 15/08/2021.

