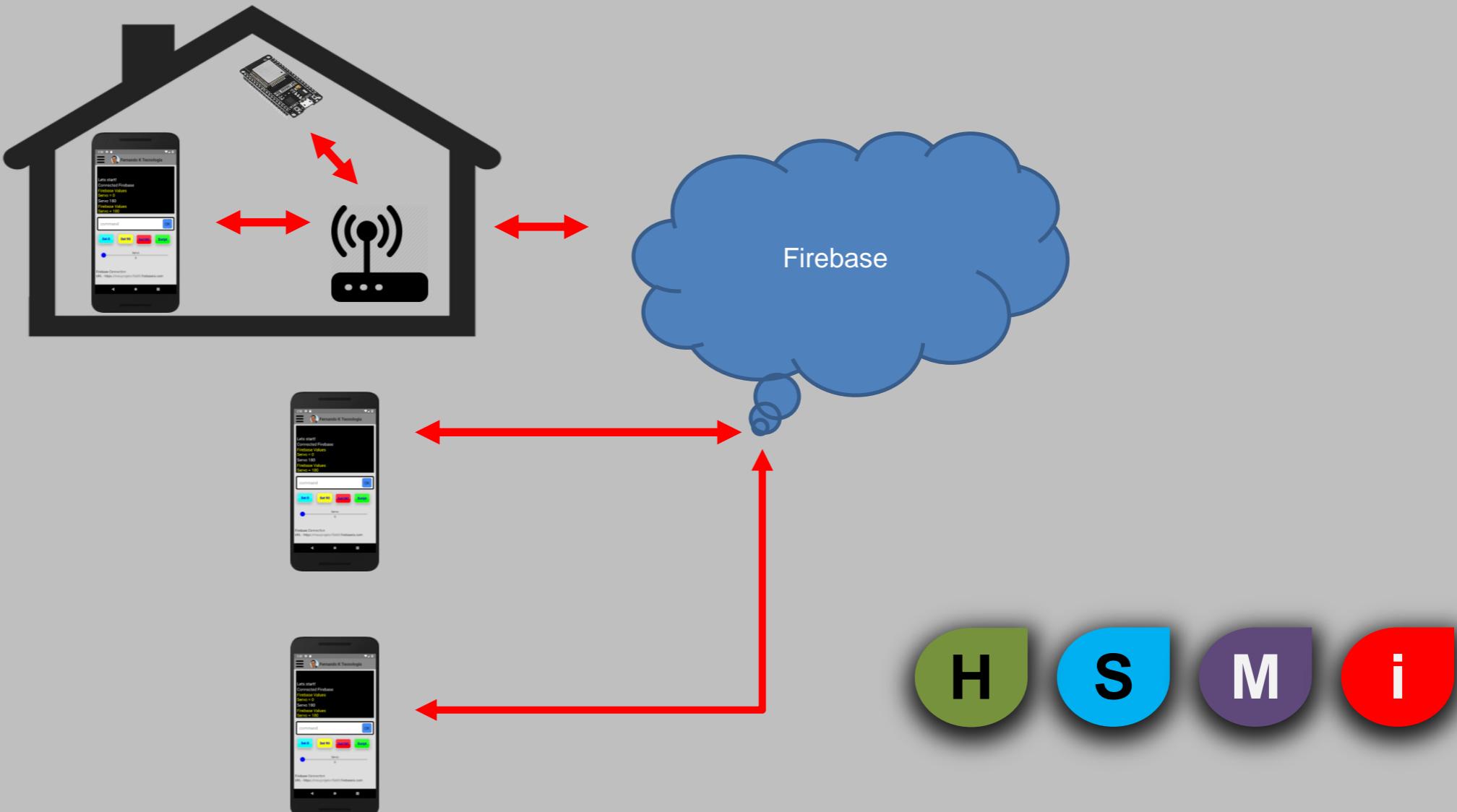
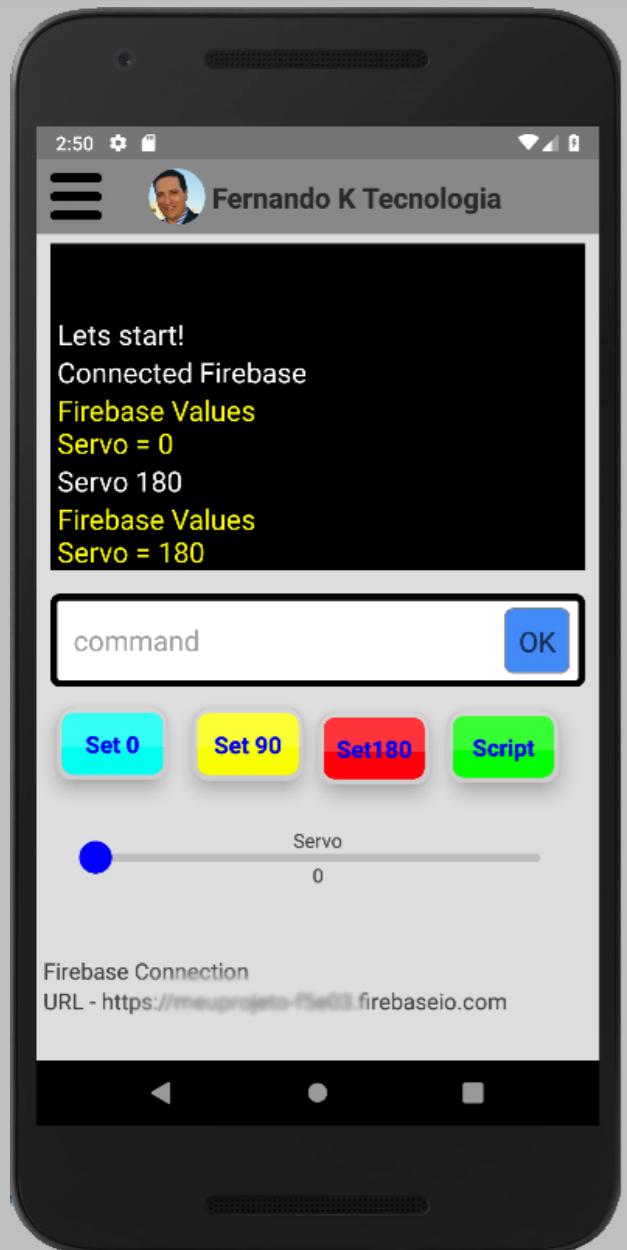


Automação com aplicativo Fernando K



Por Fernando Koyanagi

Objetivo desta aula

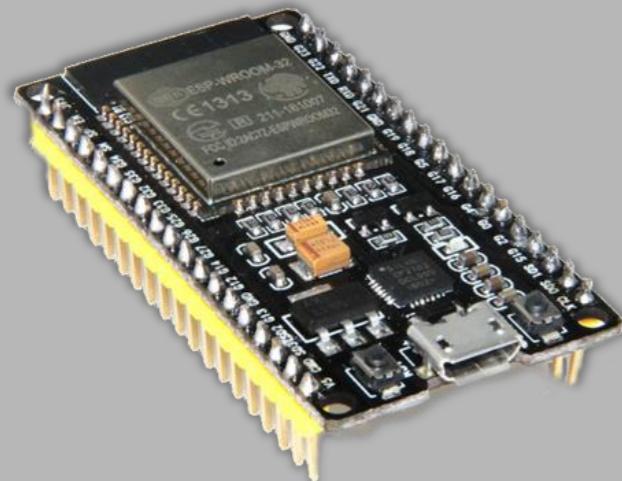
Apresentar o aplicativo Fernando K e suas ferramentas:

- Conexões
 - Socket (Local)
 - Firebase
- Componentes
 - Botões
 - Sliders



Modelo **ESP32** utilizado

Nas figuras abaixo temos os modelos mais comuns de ESP32 atualmente.
O mesmo projeto foi testado para estes 3 modelos.



ESP-WROOM-32 DevKit
DOIT
38 pinos



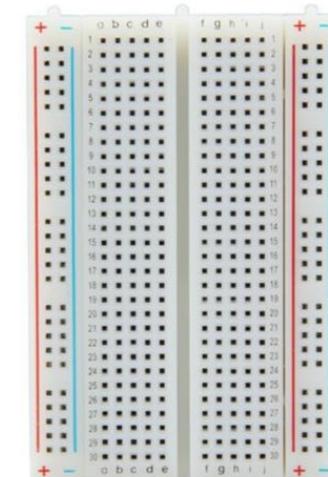
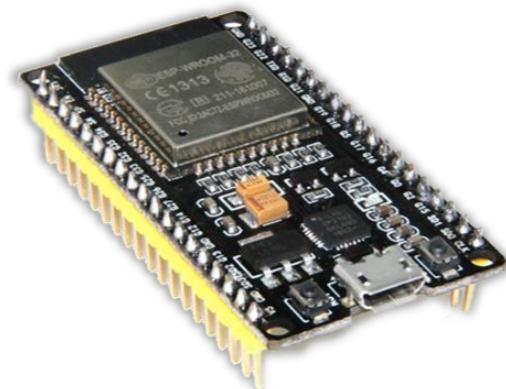
ESP-WROOM-32 DevKit
ESPRESSIF
38 pinos



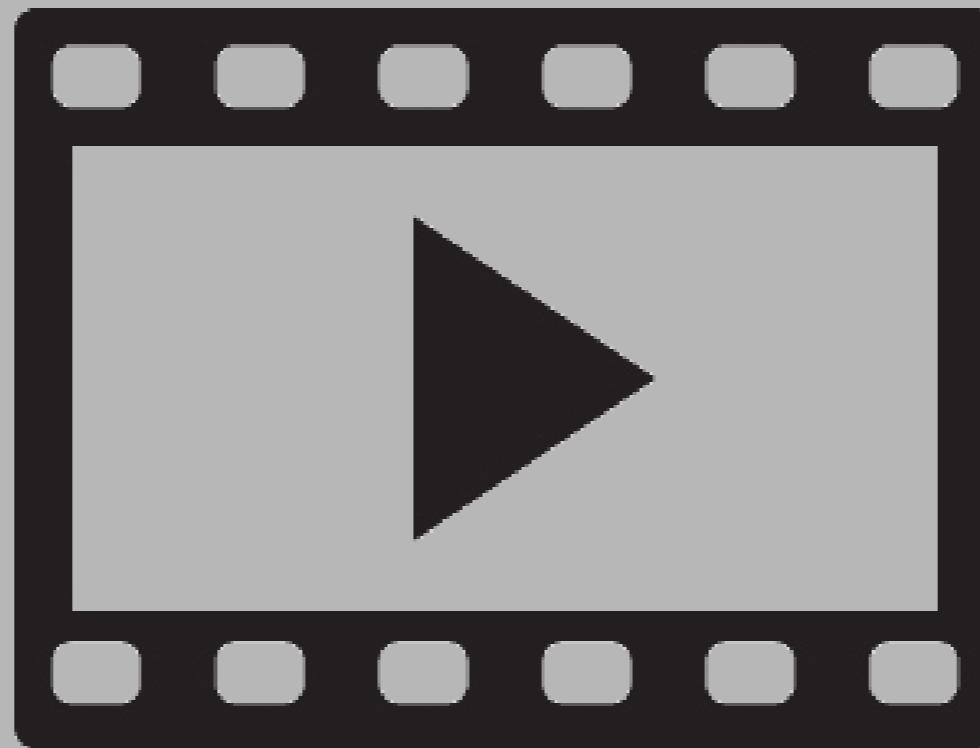
ESP-WROOM-32 DevKit
DOIT
30 pinos

Recursos usados

- ESPWROOM-32 DevKit de 30 pinos
- Servo Motor
- Fios
- Cabo USB para o ESP32
- Celular Com aplicativo Fernando K



Demonstração



Domingo, Janeiro 14 2018

Em www.fernandok.com

PRINCIPAL SOBRE FERNANDO K ARDUINO ESP8266 ESP32 LORAWAN MOTOR DISPLAY MATERIAIS DOWNLOAD

**Receba o meu conteúdo
GRATUITAMENTE**

Insira aqui seu melhor email...

QUERO RECEBER GRÁTIS**Seu e-mail**

Motor de Passo Nema 23 com Driver TB6600 e Arduino Due

by Fernando K Tecnologia - 2:44 PM

Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...

[Leia mais](#)

ESP32 Longa Distância - LoRaWan

by Fernando K Tecnologia - 9:46 AM

Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...

[Leia mais](#)

Motor de HD com Arduino

by Fernando K Tecnologia - 2:00 PM

QUAL ASSUNTO VOCÊ TEM PRA FAZER?

- Arduino
- ESP8266
- ESP32
- Motor
- Display
- Sensor

You may select multiple answers.

[Votar](#) [Exibir resultados](#)Votos até o momento: 32
Dias restantes para votar: 49

FACEBOOK



forum.fernandok.com

Fórum Fernando K Tecnologia
Fórum sobre dúvidas com relação ao conteúdo disponibilizado pelo Fernando Koyanagi

Nosquisar...

www.fernandok.com /fernandokoyanagi /fernandokoyanagi /fernandok_oficial /fernandok_oficial

[Links rápidos](#) [L...](#) [fernandokoyanagi](#)

Bem-vindo: 05/Oct/2018, 11:16 A sua última visita foi em 10/Set/2018, 15:47

Assinalar todos os fóruns como lidos

SUporte Fórum Fernandok

	TÓPICOS	MENSAGENS	ÚLTIMA MENSAGEM
Feedback Dúvidas, críticas ou sugestões sobre o Fórum FernandoK. Para demais questões utilize o fórum correto.	6	11	Re: O russo voltou por Ipmehi 01/Oct/2018, 08:25

FERNANDO K

	TÓPICOS	MENSAGENS	ÚLTIMA MENSAGEM
Arduino Projetos de arduino	31	79	skardy bogii por Soresorcem 05/Oct/2018, 10:55
ESP32 Projetos de ESP32	29	62	Dúvidas sobre como instalar a... por Marcos Sergio 04/Oct/2018, 15:52
ESP8266 O ESP8266 é um microcontrolador do fabricante chinês Espressif que inclui capacidade de comunicação por Wi-Fi.	24	51	Re: NodeMCU não conecta em qu... por ivanribeira 04/Oct/2018, 14:39
LoRa Projetos com LoRa	11	31	Projeto de irrigação de jardim por marlendo 04/Oct/2018, 21:30
STM32 Projetos com STM32	3	8	Re: Imprecisão de tempo de de... por biazoto 12/Sep/2018, 09:15
Motor Projetos com motor	5	11	Re: impressora 3d com motor dc por Magneton 24/Sep/2018, 19:05
Display Projetos com Display	4	11	Re: Alguém conhece o VIRTUINO... por Jod Luz 21/Sep/2018, 11:39

QUEM ESTÁ ONLINE
No total, há 4 usuários online :: 2 usuários registrados, 0 invitado e 2 visitantes (baseado em usuários ativos nos últimos 5 minutos)
O recorde de usuários online foi de 19 em 11/Sep/2018, 05:37

Usuários registrados: alberto, fernandokoyanagi
Legenda: Administradores, Moderadores globais

ANIVERSÁRIOS
Não há aniversários hoje

ESTATÍSTICAS
Total de mensagens 703 • Total de tópicos 114 • Total de membros 469 • Novo usuário: Soresorcem

[L...](#) [Anunciar](#)

Powered by phpBB® Forum Software © phpBB Limited
Traduzido por: Suporte phpBB
[Painel de Controle da Administração](#)



Instagram
fernandok_oficial



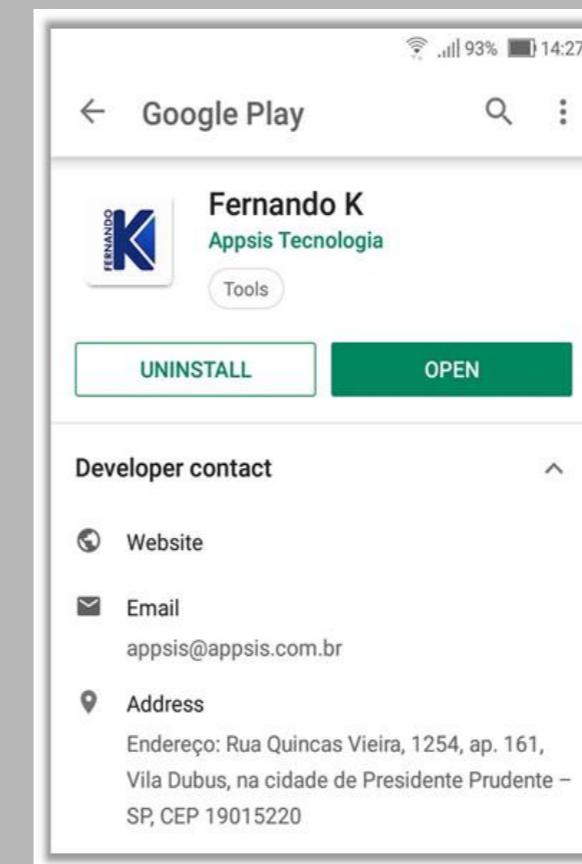
Telegram
fernandok_oficial



Link para download do Aplicativo

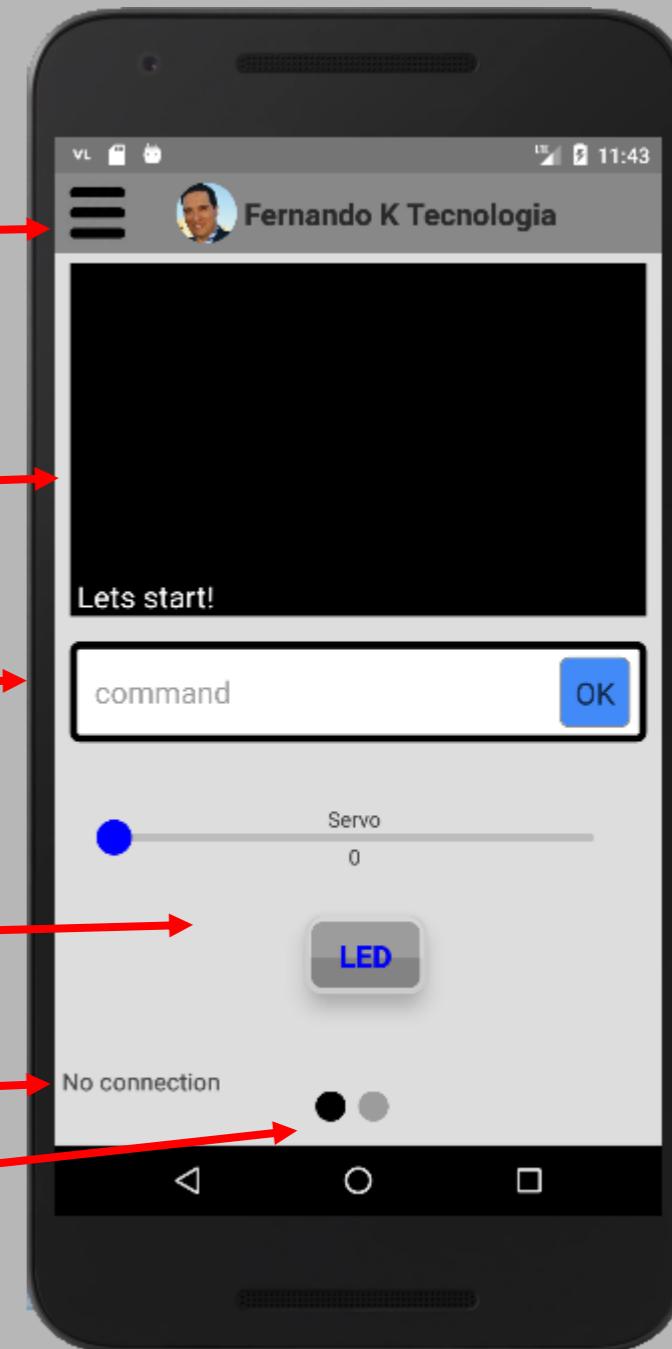
android: <https://play.google.com/store/apps/details?id=com.appfernandok>

iOS: <https://itunes.apple.com/us/app/fernando-k/id1449589064>



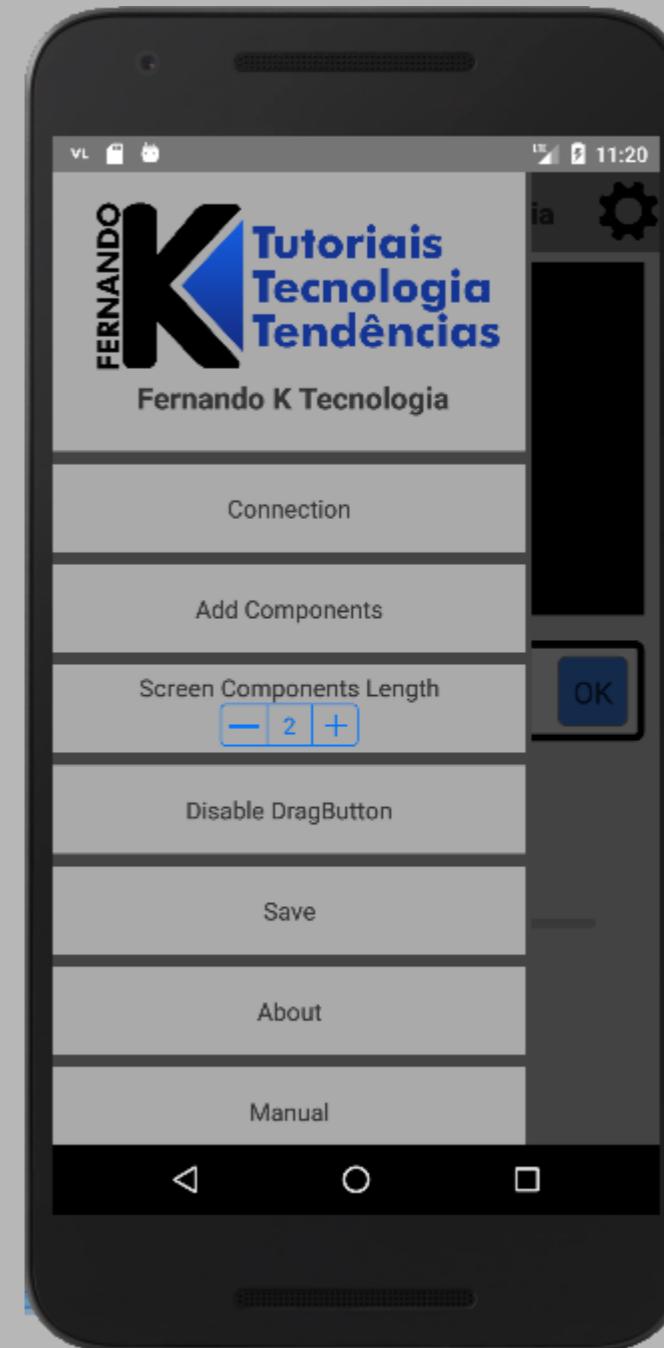
Tela inicial

- Action Bar:
 - Botão Menu
 - Fernando K Tecnologia
- Tela de entrada e saída:
 - Tela Informando as entradas e saídas de dados.
- Command:
 - Campo de texto do comando
 - Botão OK
- Área dos componentes
- Conexão Atual
- Paginação



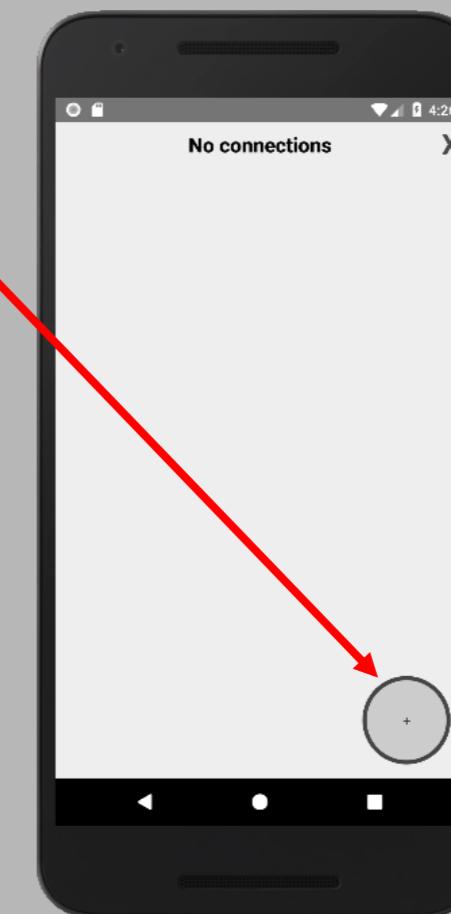
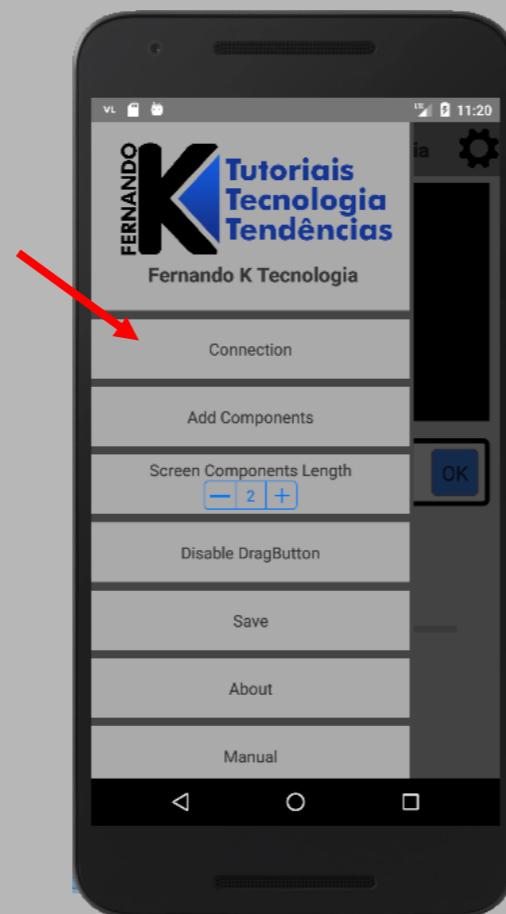
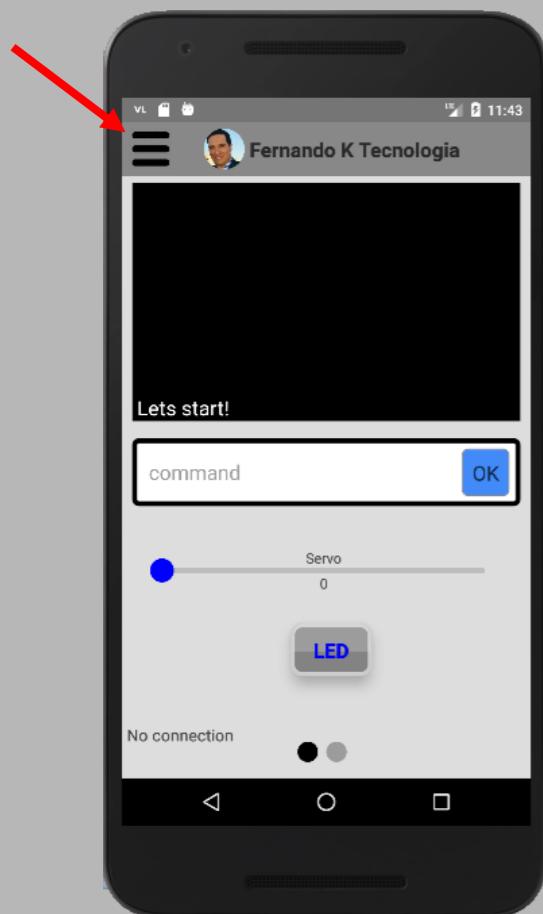
Menu Options

- Connection
- Add Components
- Screen Components Length
- Enable/Disable DragButton
- Save
- About
- Getting Buttons
- Create Buttons
- Save Buttons



Adicionar nova Conexão

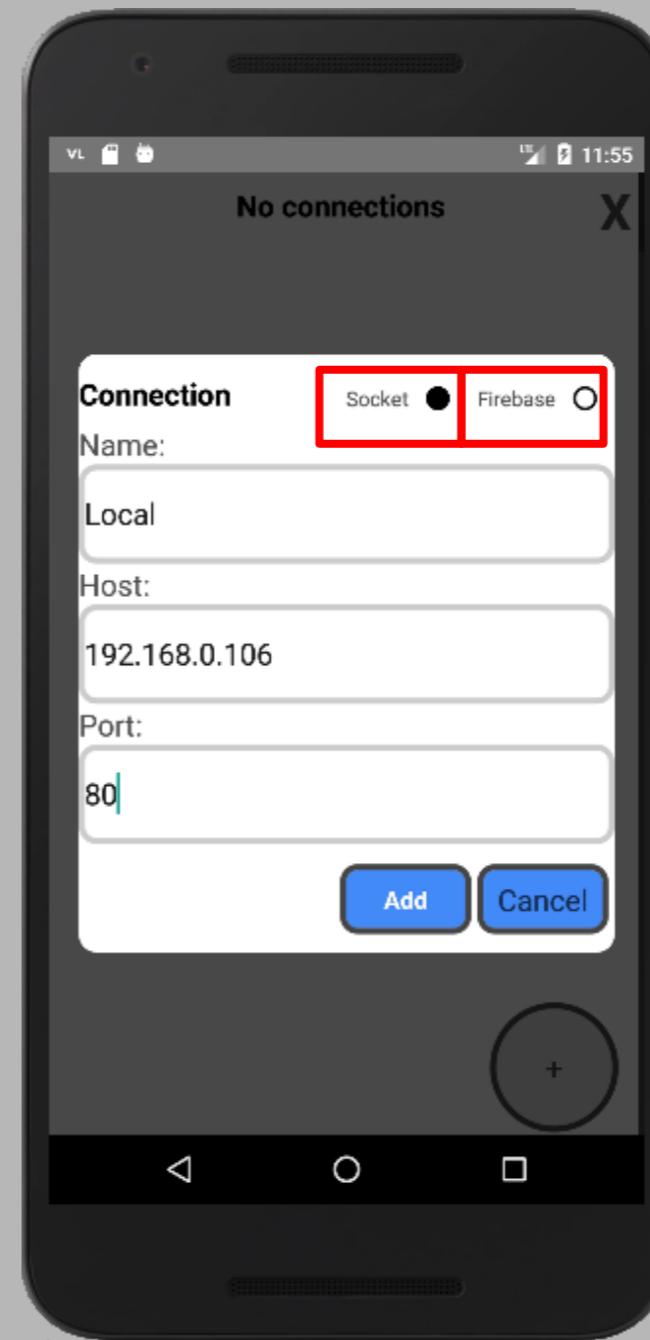
1. Clique no Botão Menu
2. Clique em Connection
3. Clique no botão “+”



Adicionar nova Conexão

Tipos de conexões:

1. Conexão por Socket.
2. Conexão por Firebase.



Conexão por **Socket**

Name: Informe o nome para identificar sua conexão

Host: Informar o Host da conexão

Port: Informar a Porta da conexão

Add: Adiciona uma nova conexão

O valor do Host e da Porta

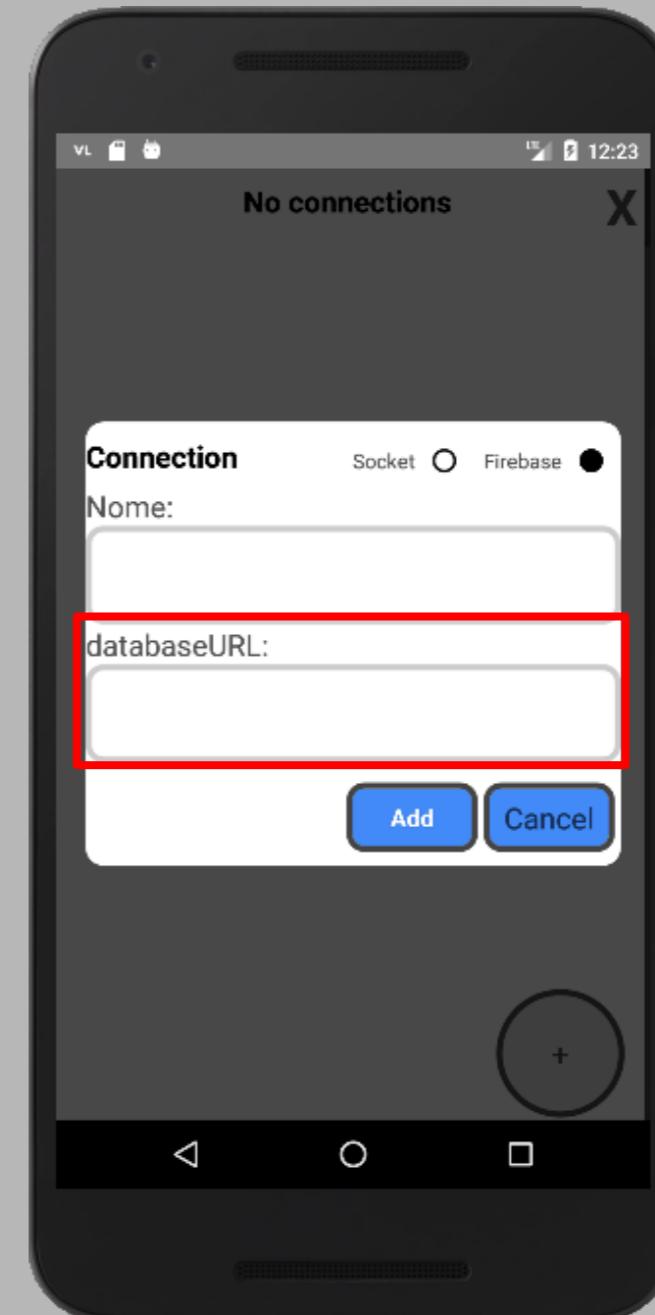
tem que ser os mesmos valores

identificados no ESP



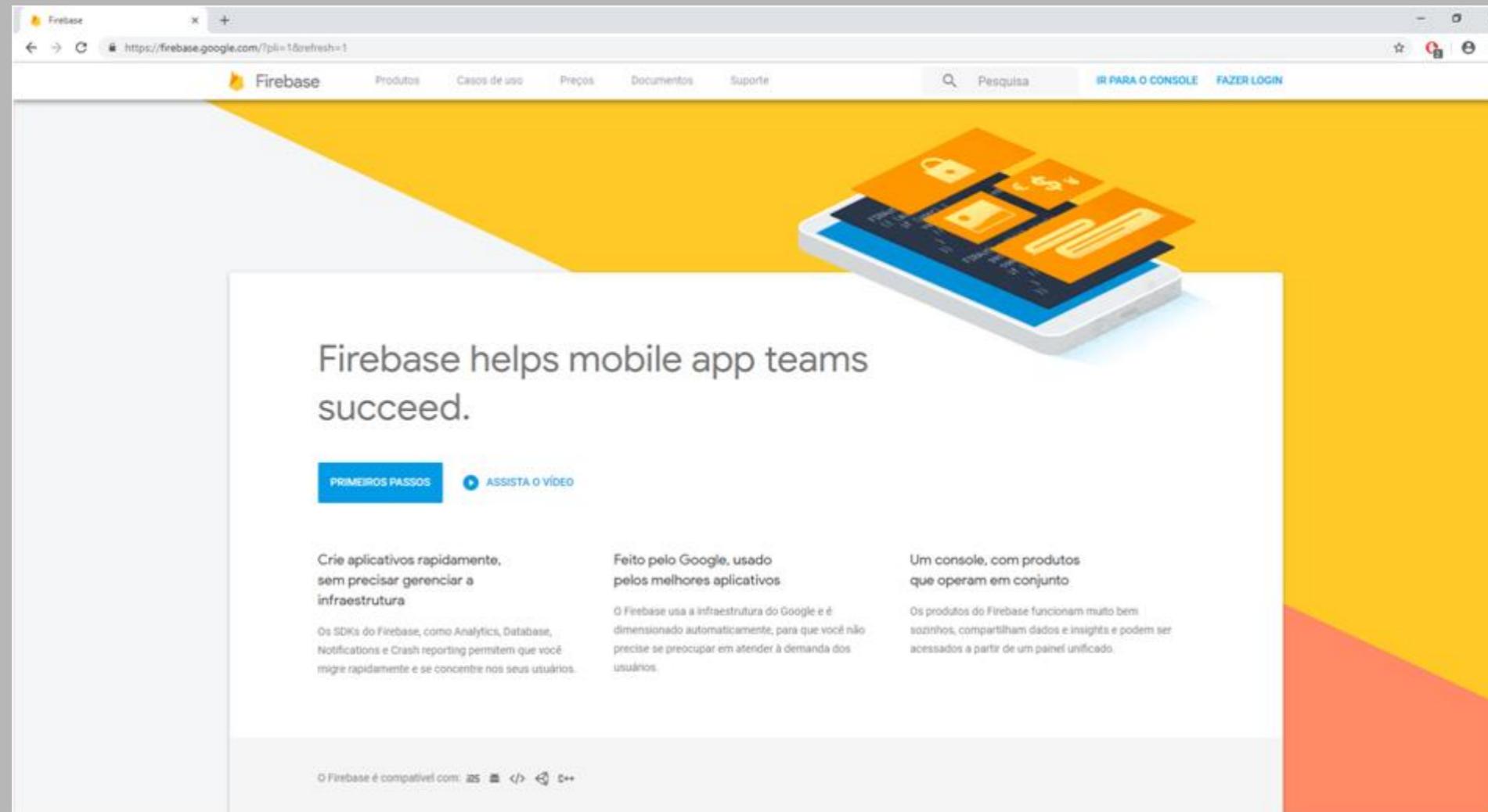
Conexão por Firebase

Para ter uma conexão com o Firebase, primeiramente é necessário criar uma conta, depois basta criar um banco de dados dentro do Firebase para obter a “databaseURL” utilizado para conectar ao aplicativo. Esses passos serão mostrados nos slides posteriores.



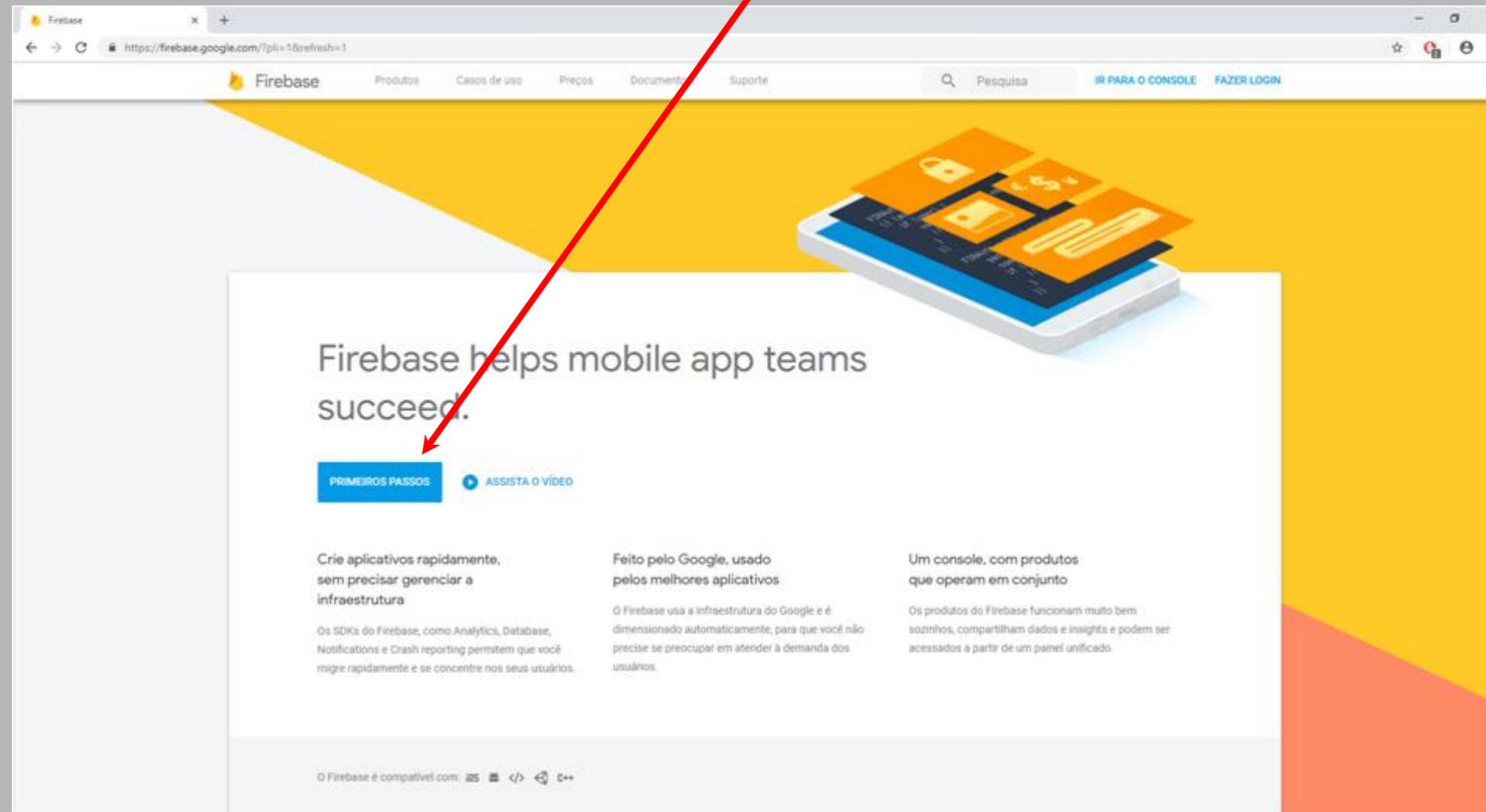
Criar banco de dados no **Firebase**

Primeiro acesse o site do Firebase. [Clique aqui](https://firebase.google.com/?pli=1&refresh=1) para acessá-lo.



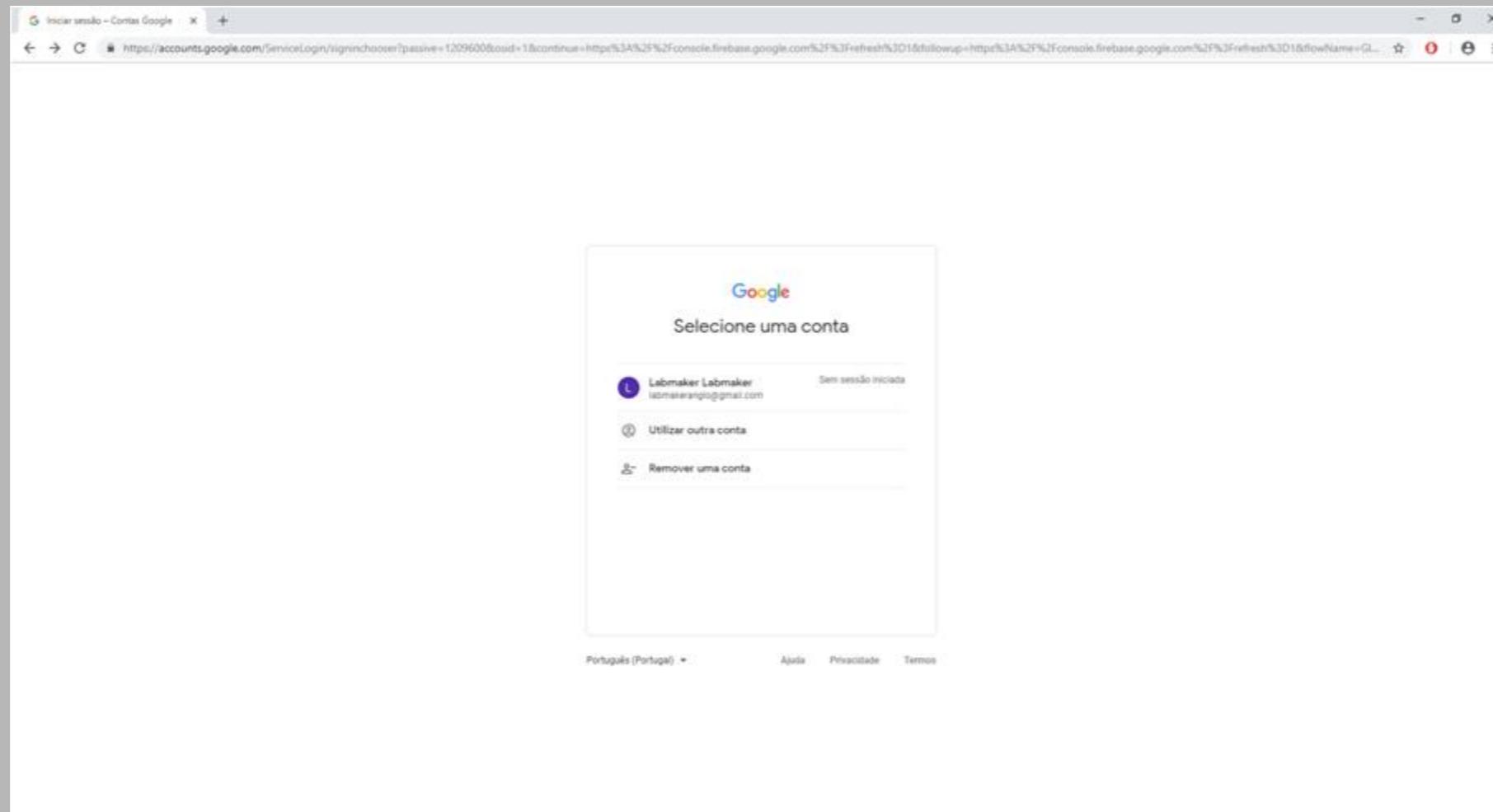
Criar banco de dados no Firebase

Clique no botão escrito “Primeiros Passos”.



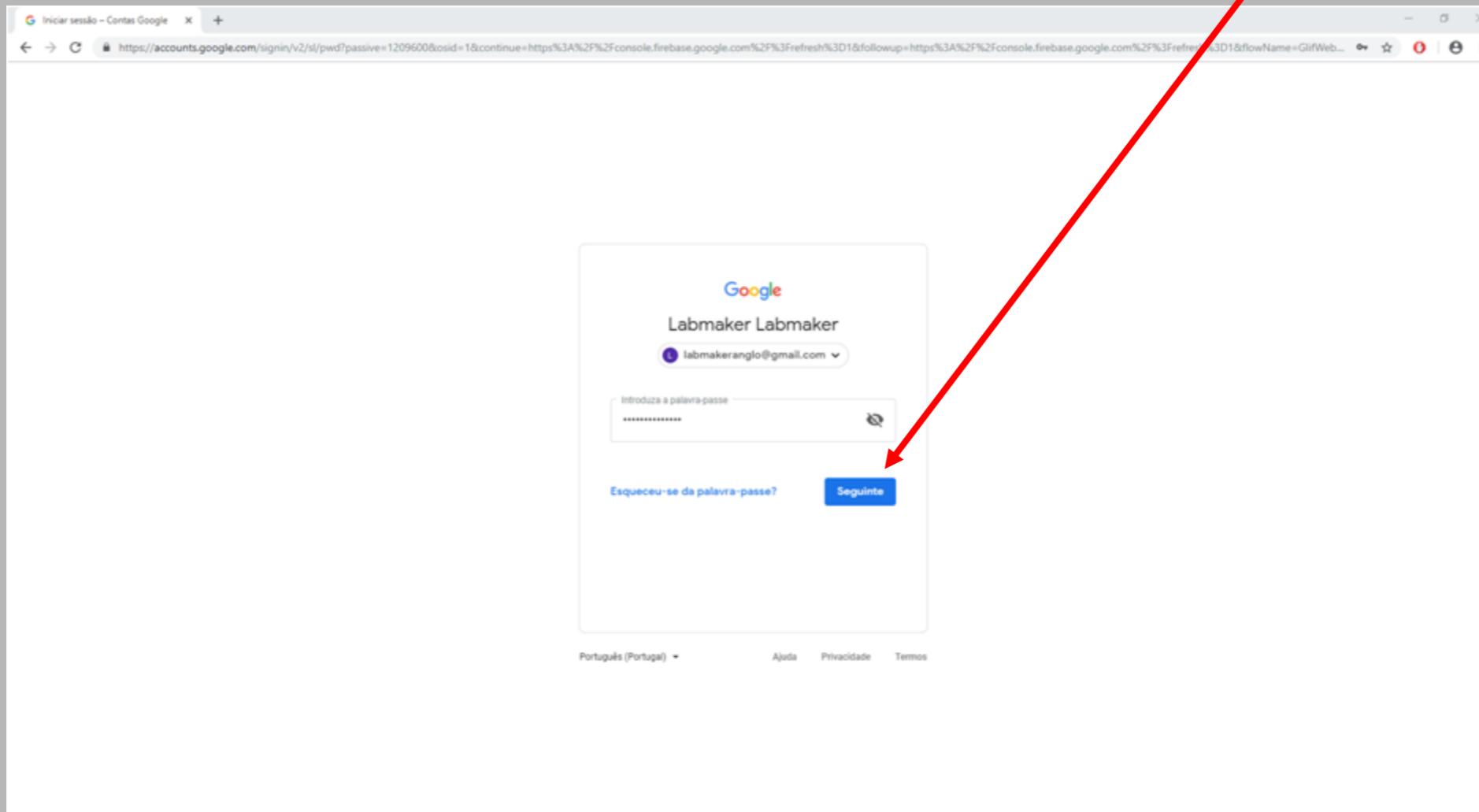
Criar banco de dados no **Firebase**

Para criar um banco de dados no Firebase é necessário ter uma conta do Google.



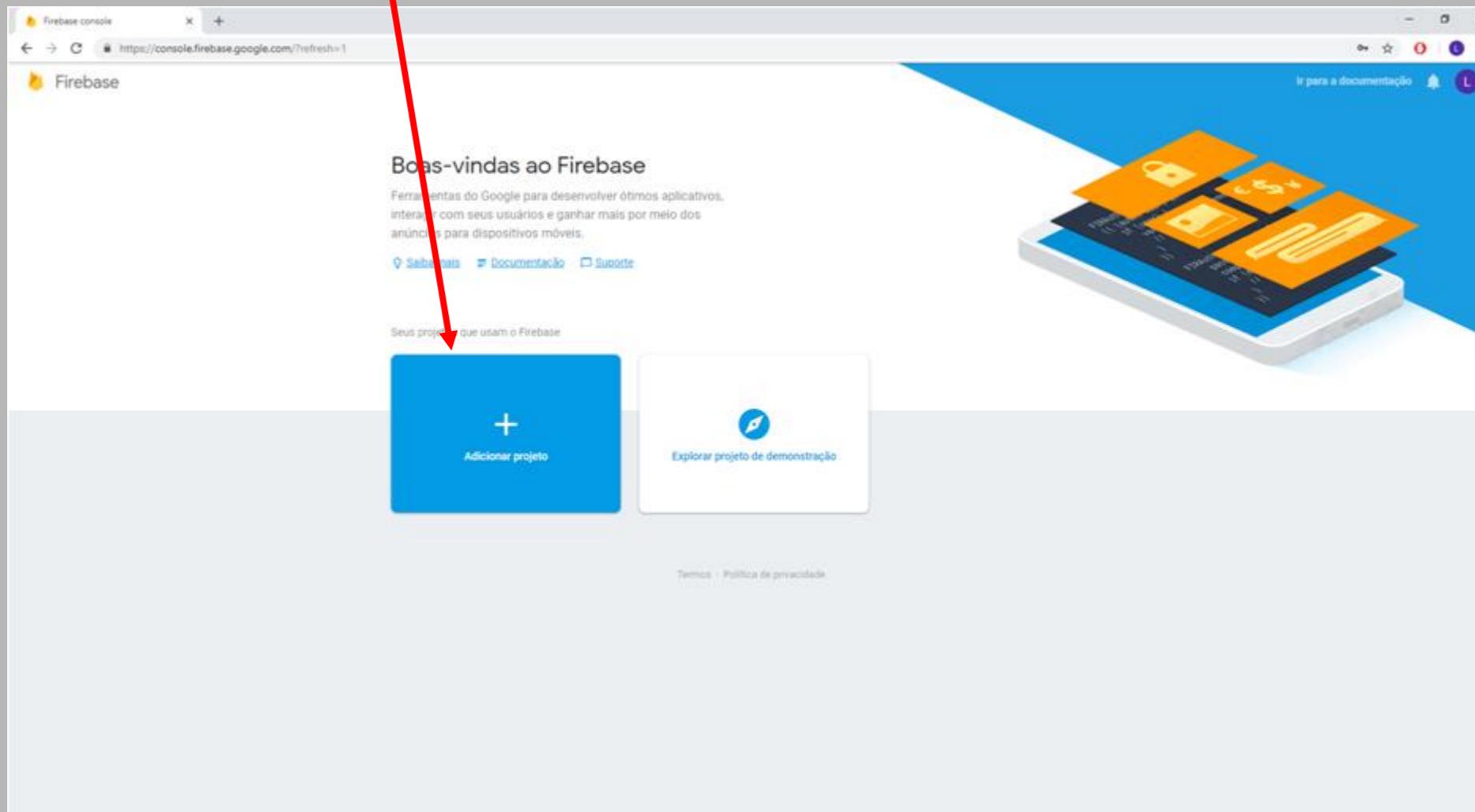
Criar banco de dados no **Firebase**

Faça o Login na sua conta e clique no botão “Seguinte”.



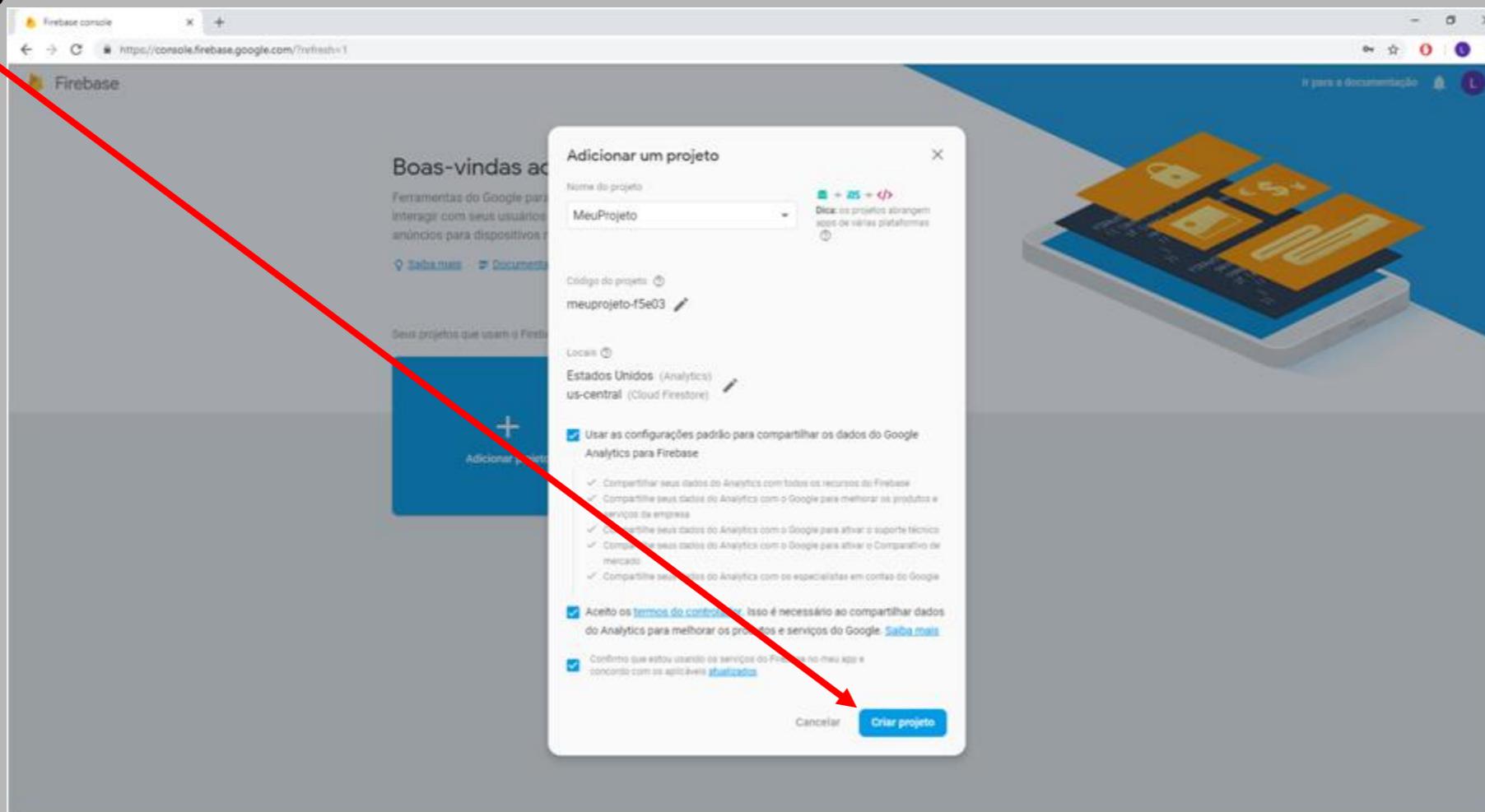
Criar banco de dados no **Firebase**

Clique em “Adicionar Projeto”.



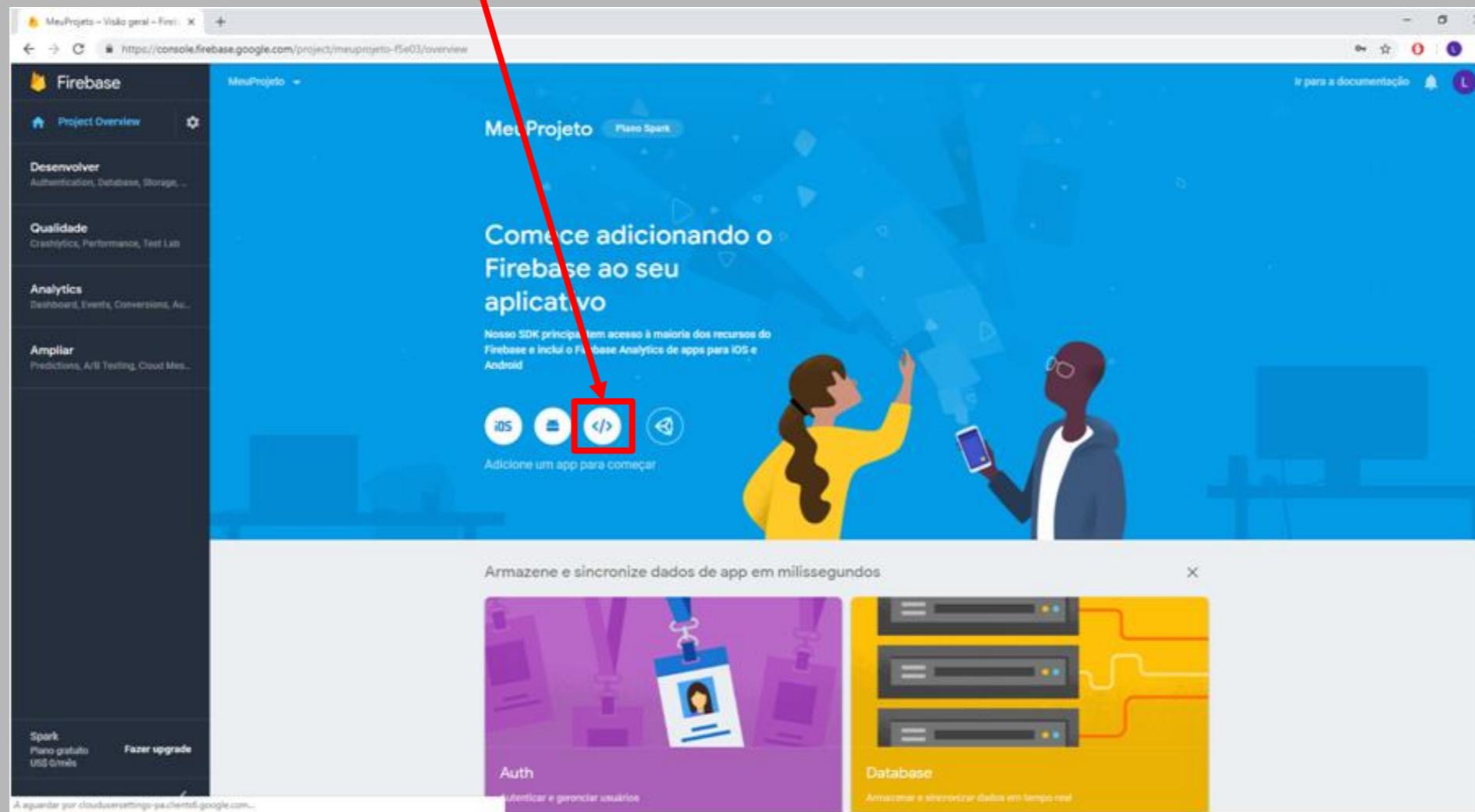
Criar banco de dados no **Firebase**

Insira um nome para seu projeto. Aceite os termos e clique em “Criar projeto”.



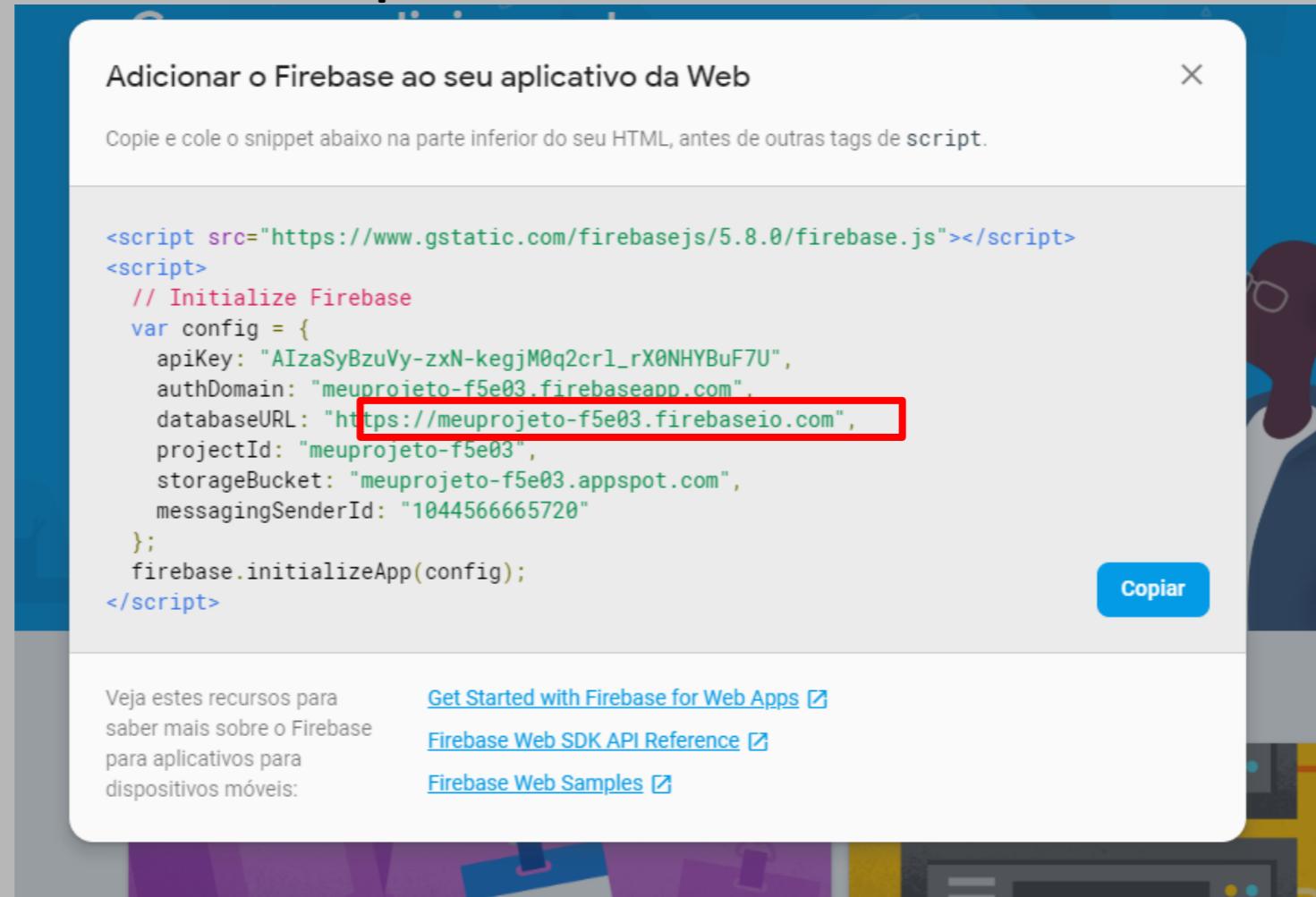
Criar banco de dados no Firebase

Após criar, clique em “</>” ao lado dos ícones do iOS e do Android.



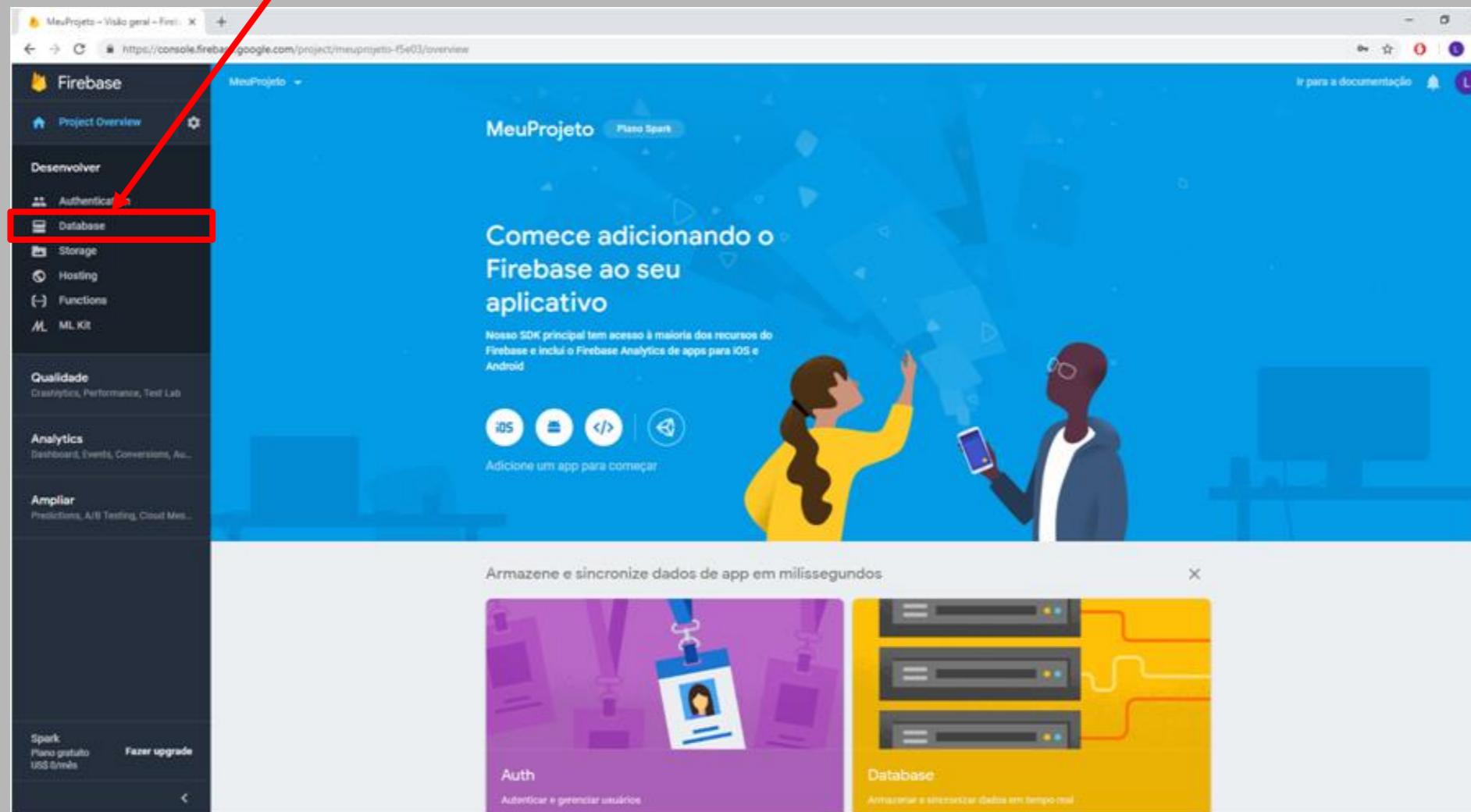
Criar banco de dados no **Firebase**

Copie e **salve** o conteúdo do “databaseURL” que será utilizado para fazer a conexão entre o aplicativo e o banco.



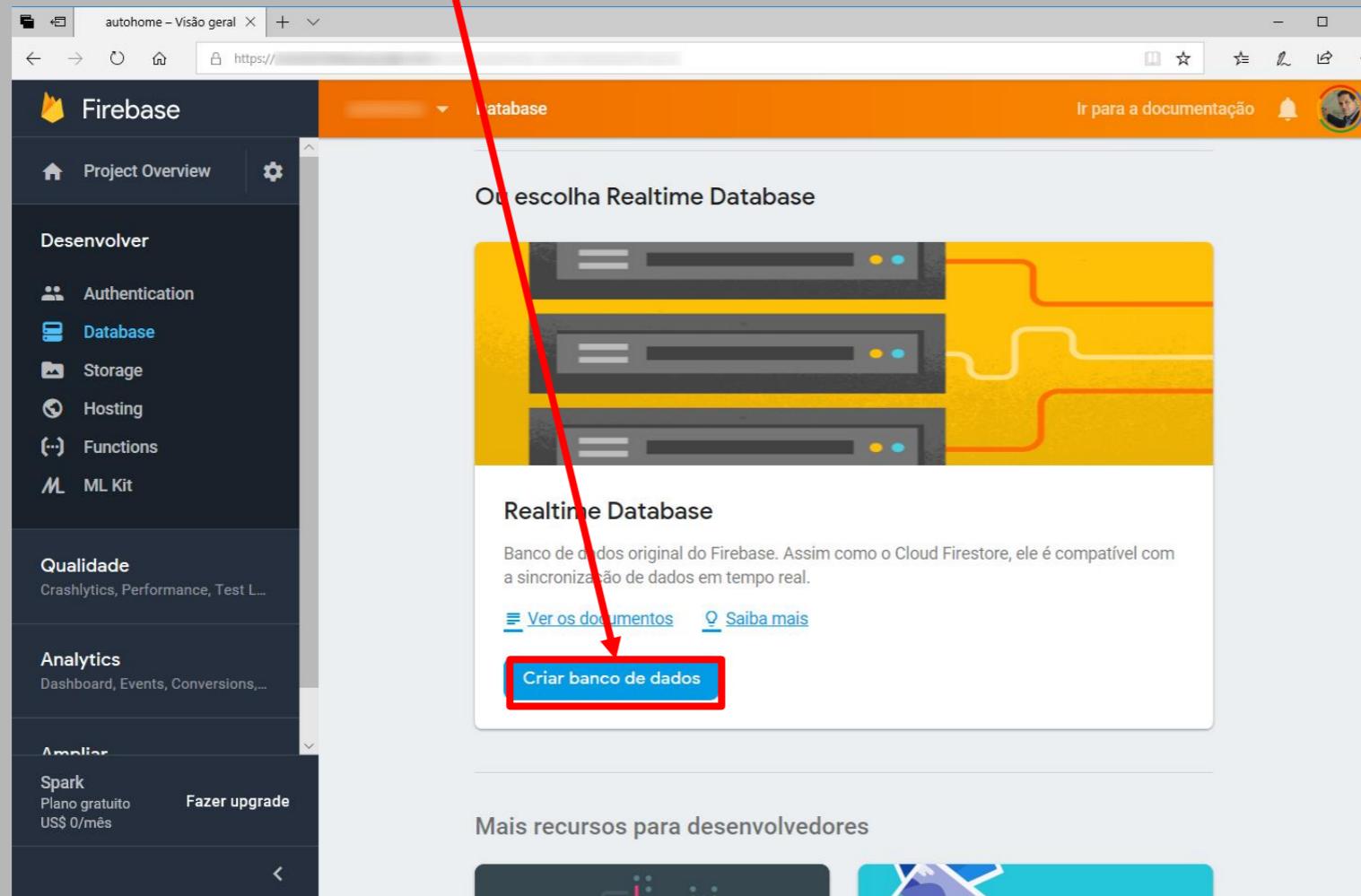
Criar banco de dados no Firebase

Clique em “Database” no lado esquerdo da tela.



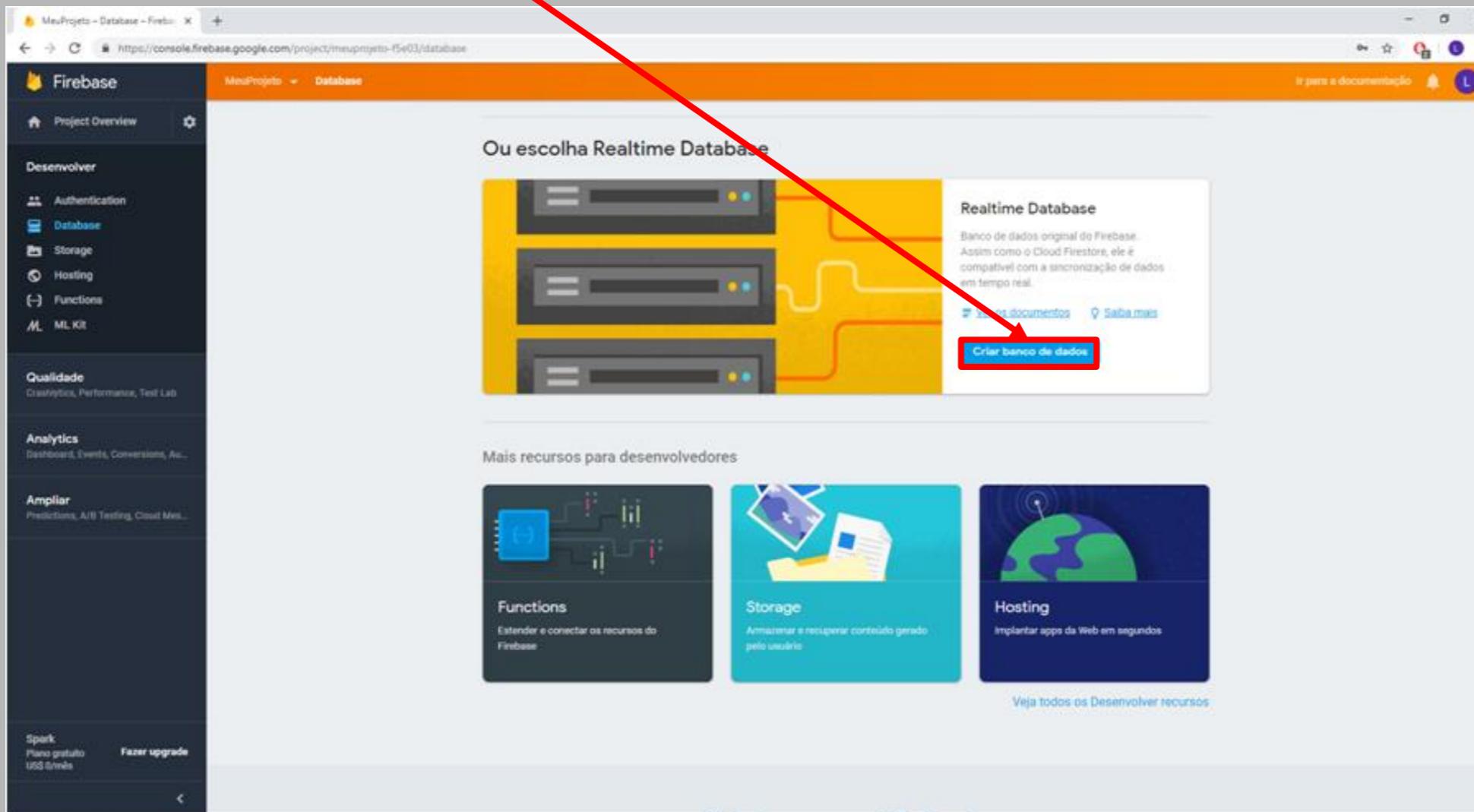
Criar banco de dados no Firebase

Clique em “Criar banco de dados”.



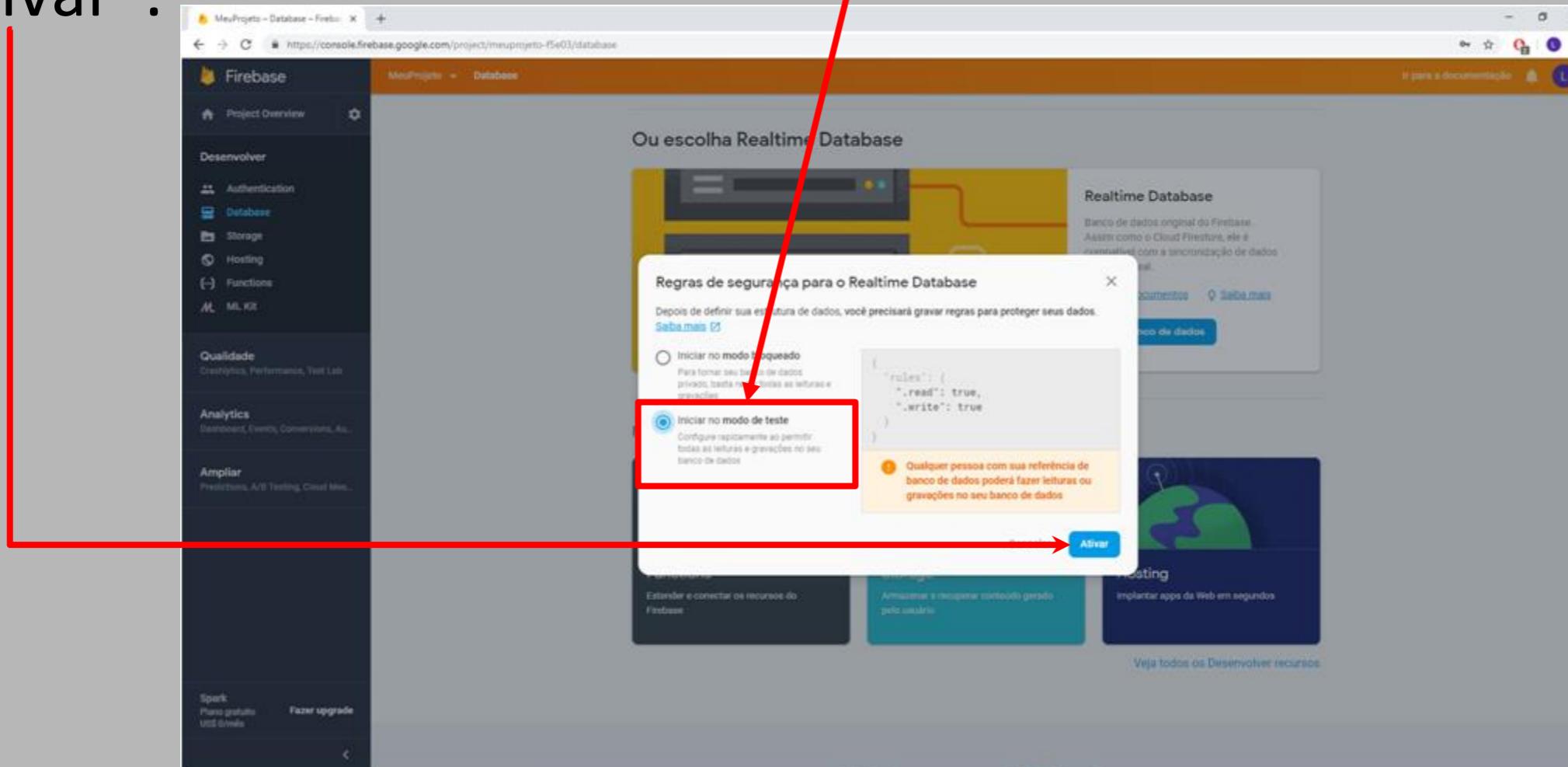
Criar banco de dados no Firebase

Clique em “Criar banco de dados” na seção Realtime Database.



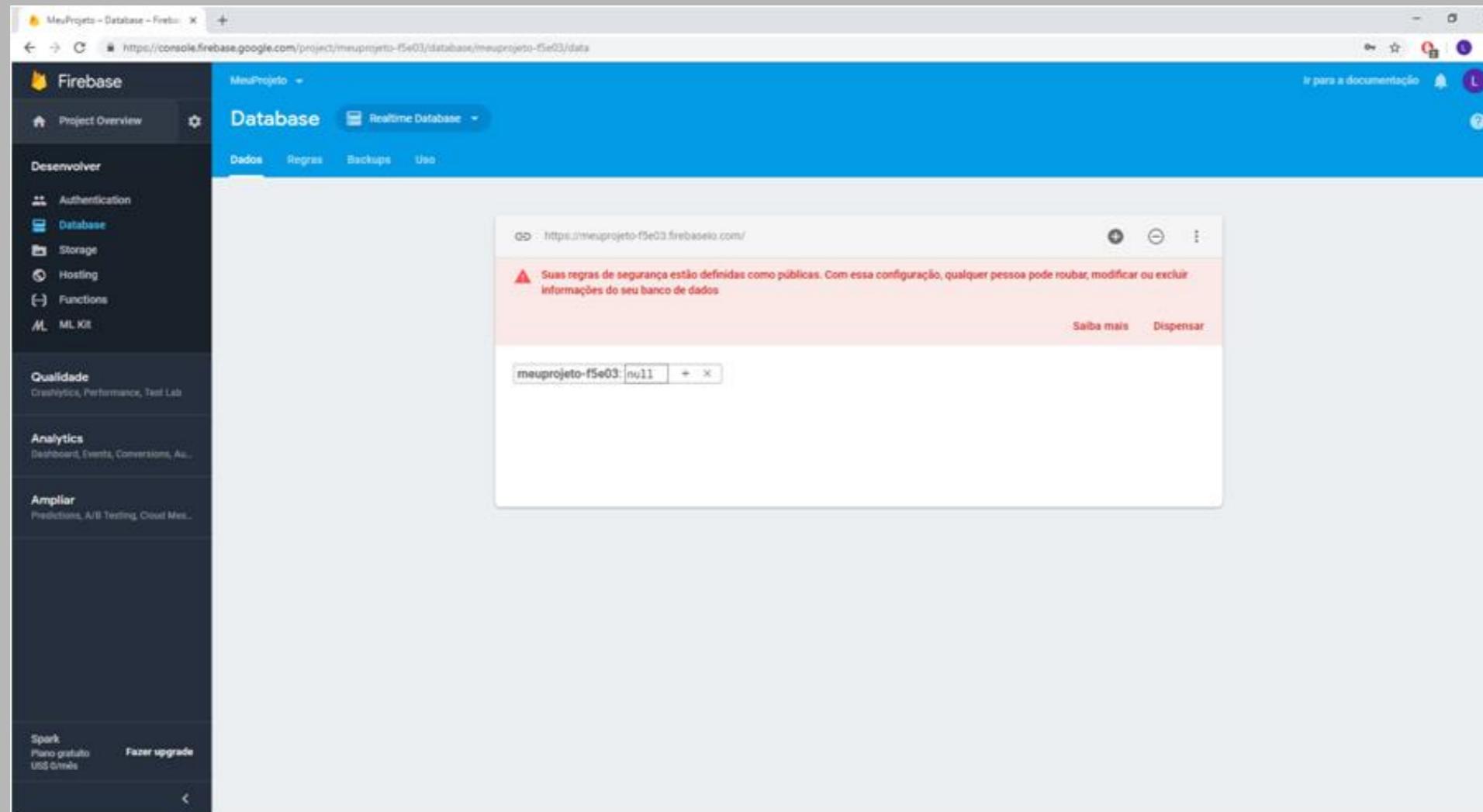
Criar banco de dados no Firebase

Selecione a opção “Iniciar no modo de teste” e depois clique em “Ativar”.



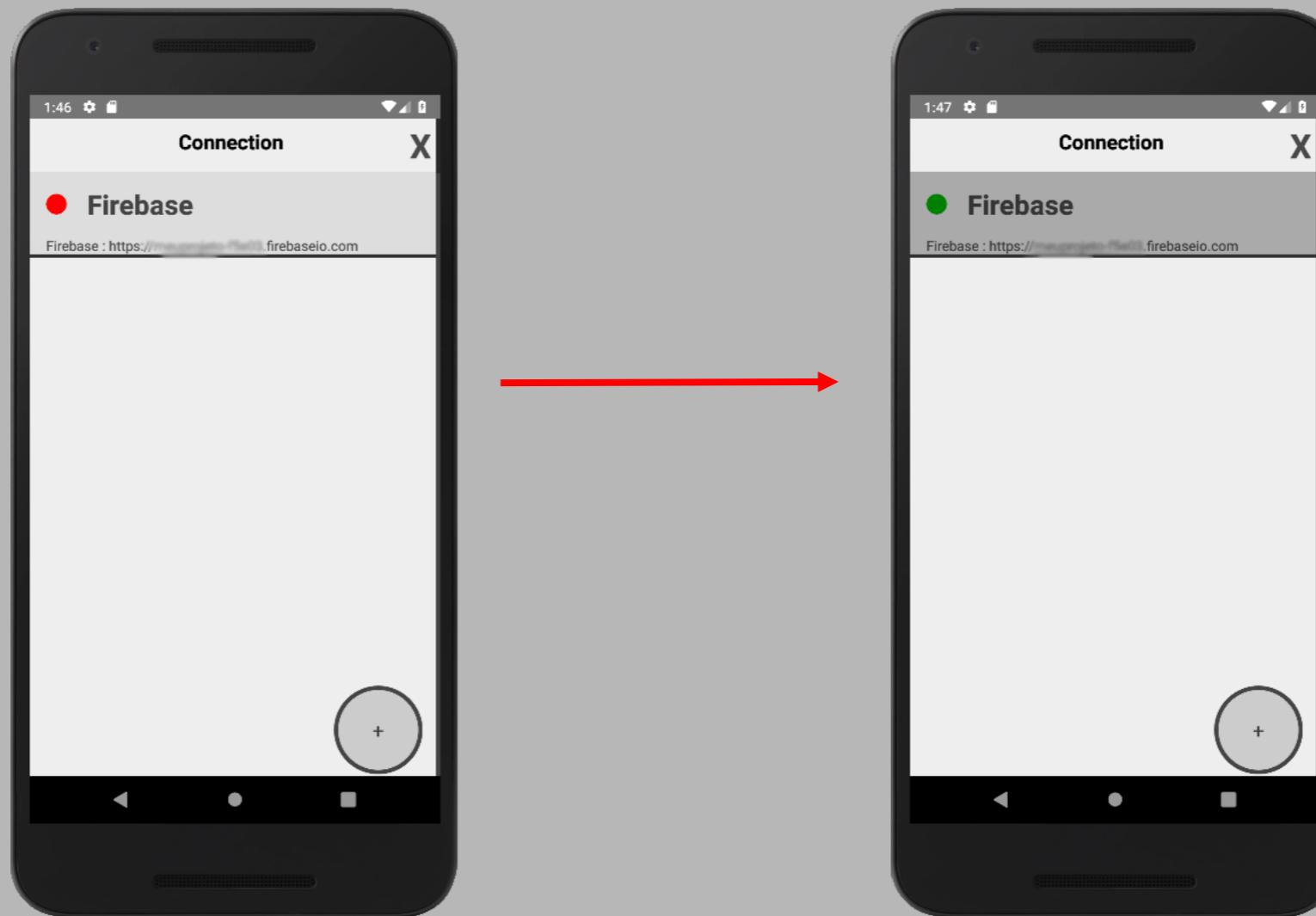
Criar banco de dados no Firebase

Pronto! O banco de dados está criado e em funcionamento.



Coneectar

Para conectar, clique na conexão.



Comunicação Aplicativo - ESP

Após estar conectado, é possível se comunicar com o ESP independente se a conexão for por Firebase ou Socket.

O Padrão de comunicação programado no ESP é o nome do comando, seguido por um valor (é possível alterar o padrão mudando o código do ESP).

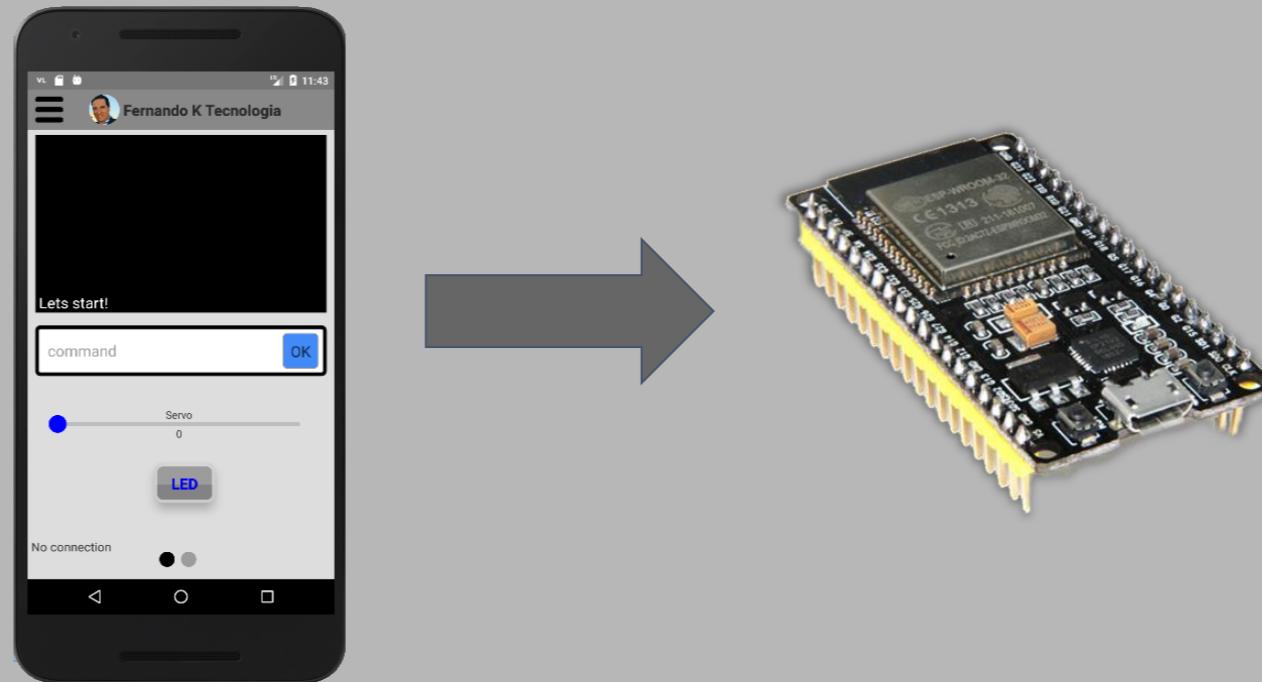
Exemplos:

- LED On
- LED Off
- Servo 0
- Servo 180

Comunicação Aplicativo - ESP

A comunicação pode ser feita através de 3 maneiras:

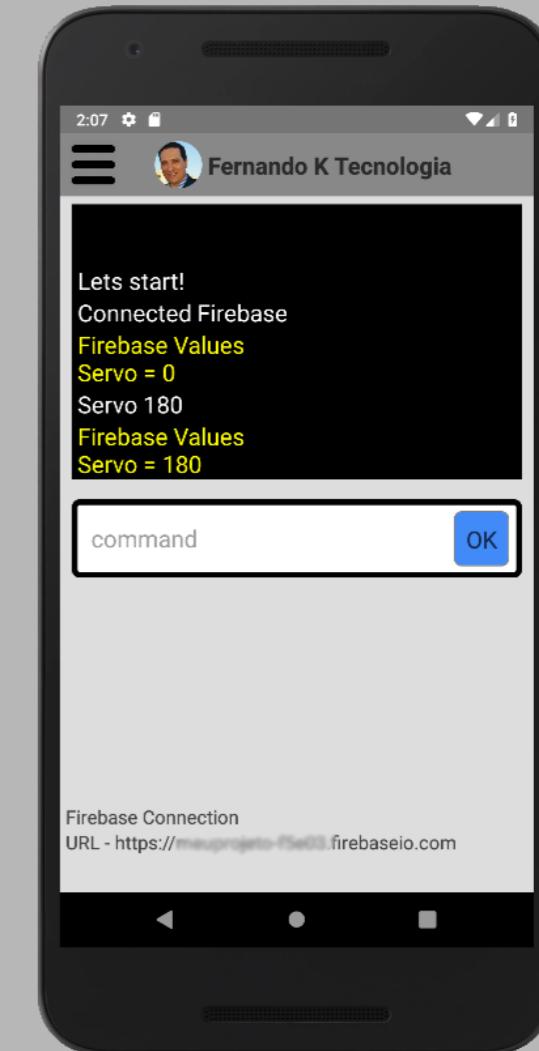
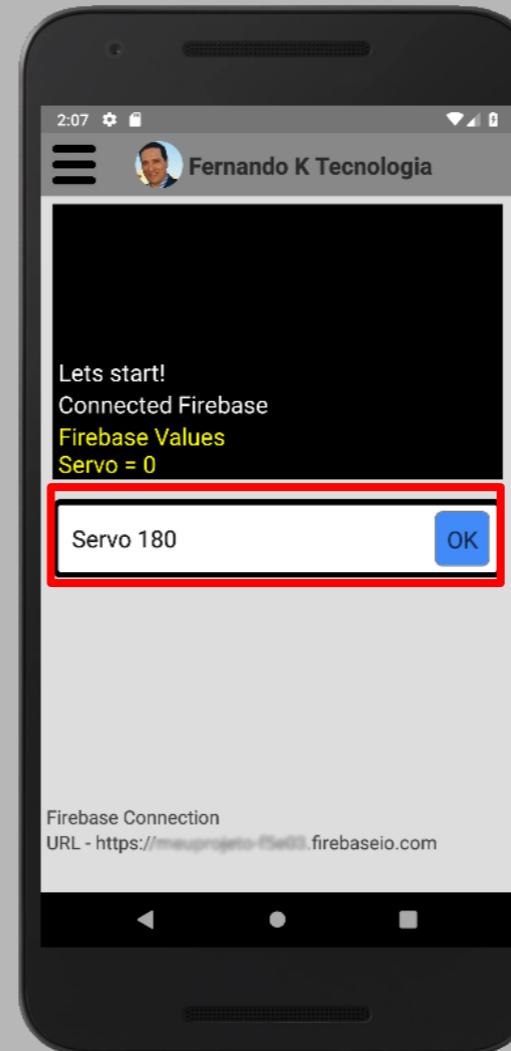
1. Terminal
2. Botão
3. Slider



Comunicação via Terminal

Digite um *comando no campo de texto e clique no botão **OK**.

*Os comandos devem estar devidamente programados no código fonte do ESP32.



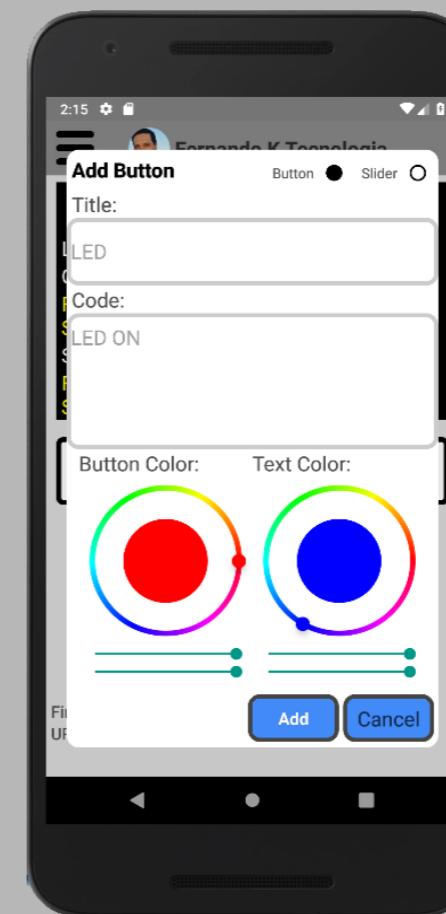
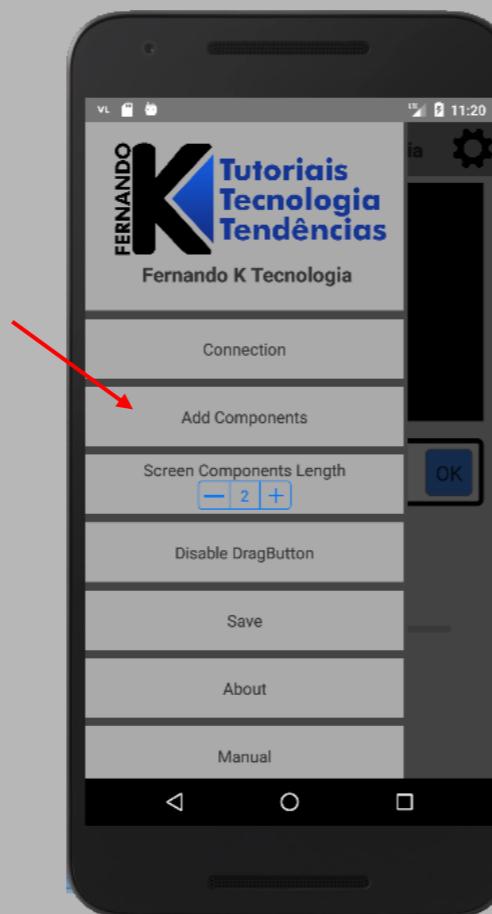
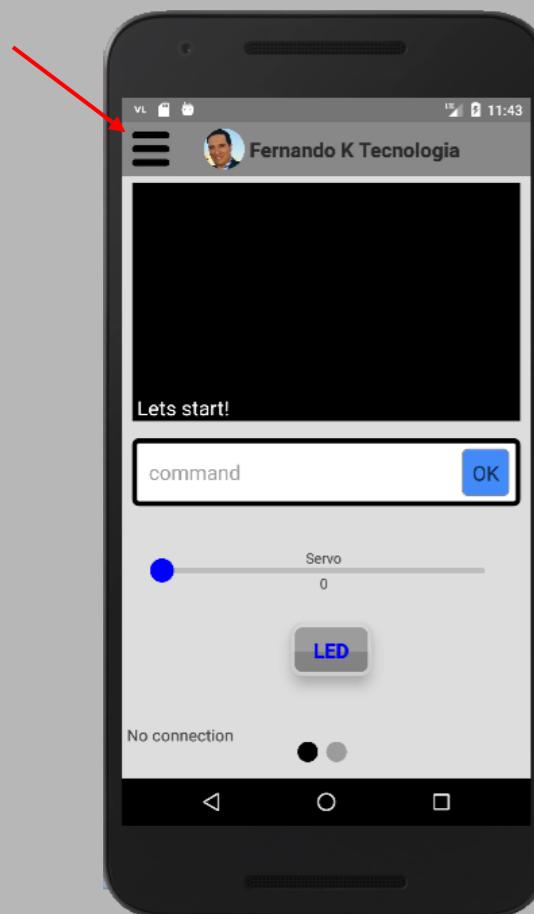
Comunicação clique Botão

O botão é um componente que facilita a comunicação com o ESP32, pois os comandos são previamente programados no botão e enviados quando ocorrer um clique.

O botão pode conter mais de um comando, como um script. Os comandos são separados pela quebra de linha e enviados em ordem.

Adicionar novo Botão

1. Clique no Botão Menu
2. Clique em Add Components



Adicionar novo Botão

Title: Informe o nome para identificar o botão

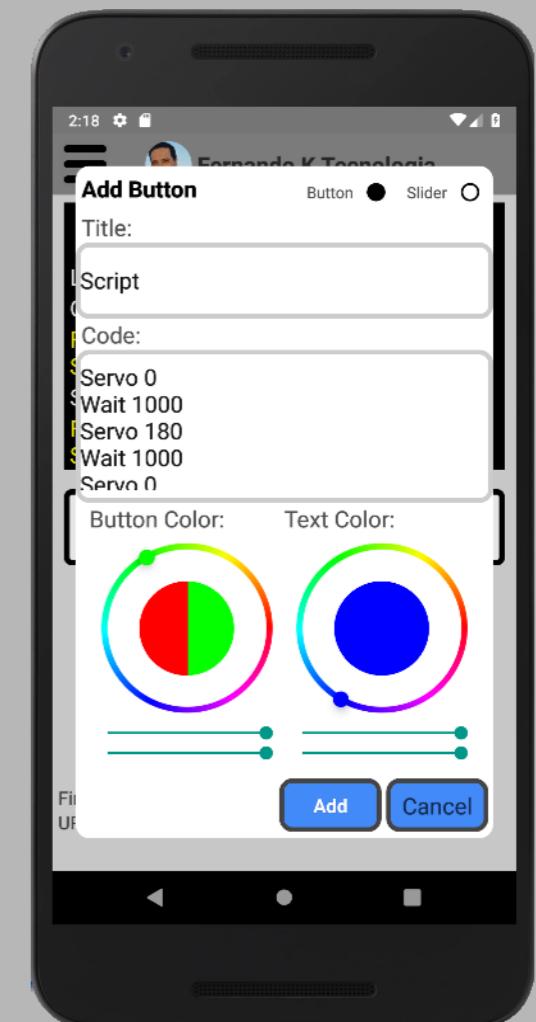
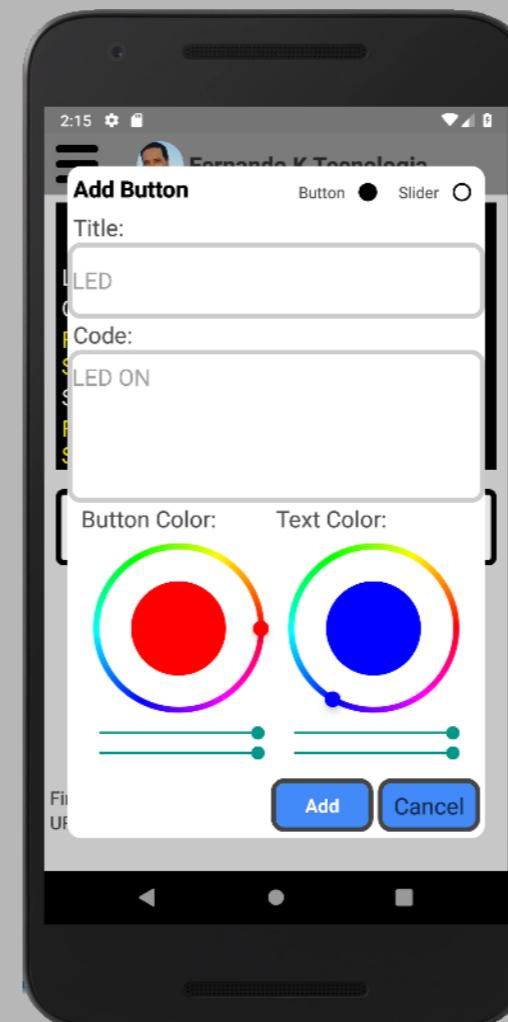
Code: *Informar o script.

Button Color: Cor Botão

Text Color: Cor do nome do Botão

Add: Adiciona o botão

*O Code pode ser um comando ou uma lista de comandos (Script).



Adicionar novo Botão

Após clicar em Add um novo botão será criado conforme a figura 2.

Figura 1

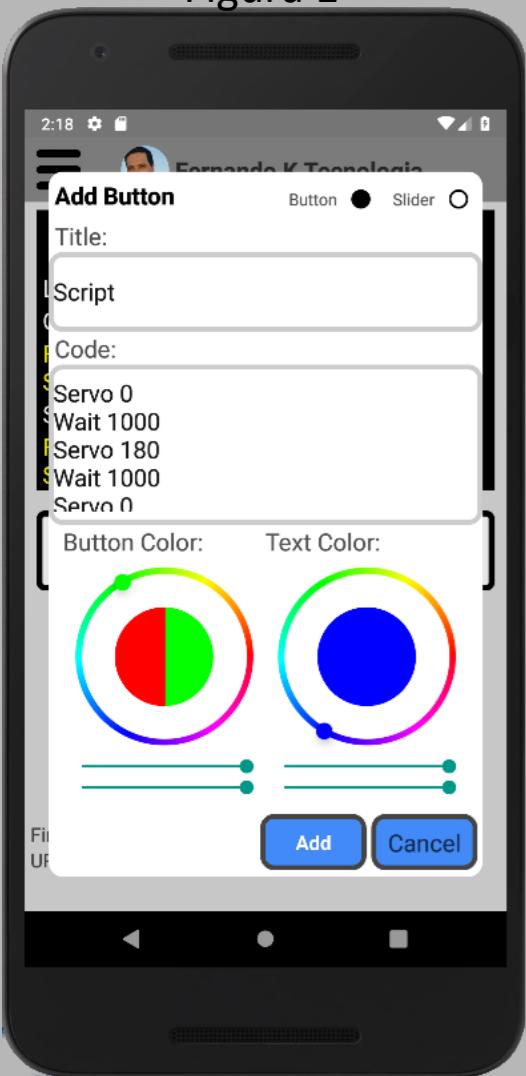
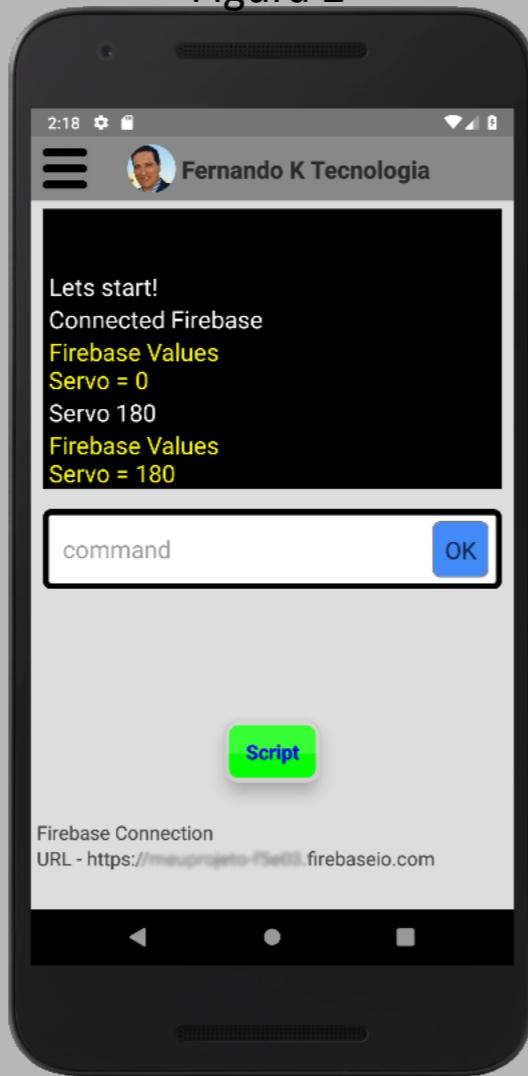


Figura 2



*A lista de comandos **reservados** (como o **Wait**) está disponível no *manual do usuário* do aplicativo.

Basta clicar no botão para executar o script colocado no Code.

Comunicação **Slider**

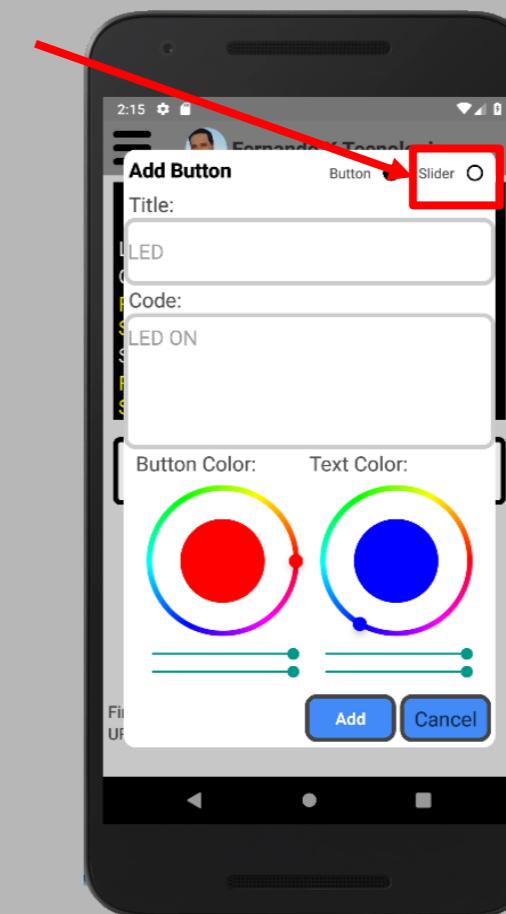
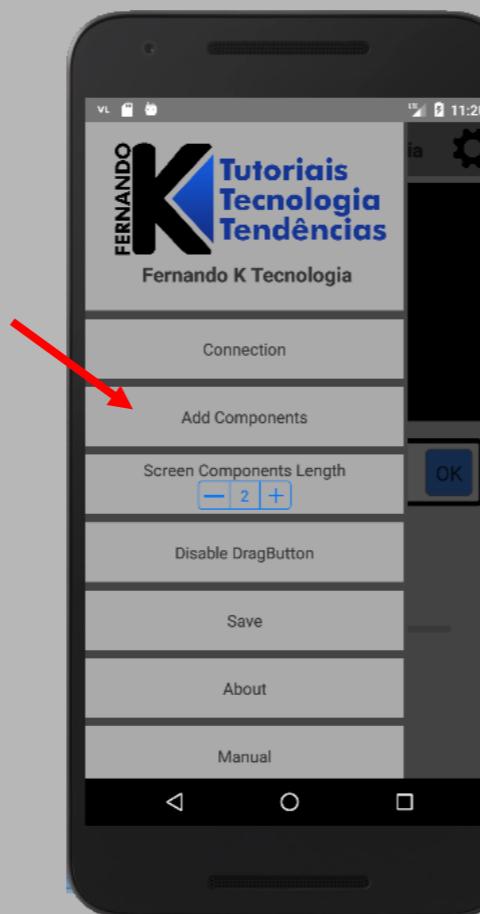
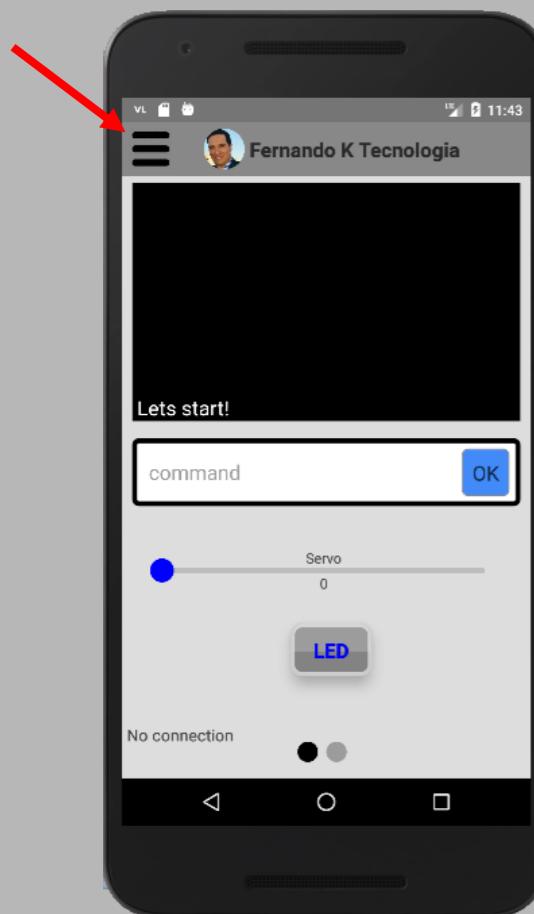
O slider é um componente que facilita a comunicação com o ESP32 quando o valor a ser enviado está dentro de um intervalo numérico.

Assim, basta arrastar o slider até o valor desejado e será enviado um comando com o nome do slider seguido do valor.

Exemplo: Se o nome do slider for Servo e estiver no valor 180. O comando enviado para o ESP será “Servo 180”.

Adicionar novo Slider

1. Clique no Botão Menu
2. Clique em Add Components
3. Clique em Slider



Adicionar novo **Slider**

Title: Informe o nome para identificar o Slider

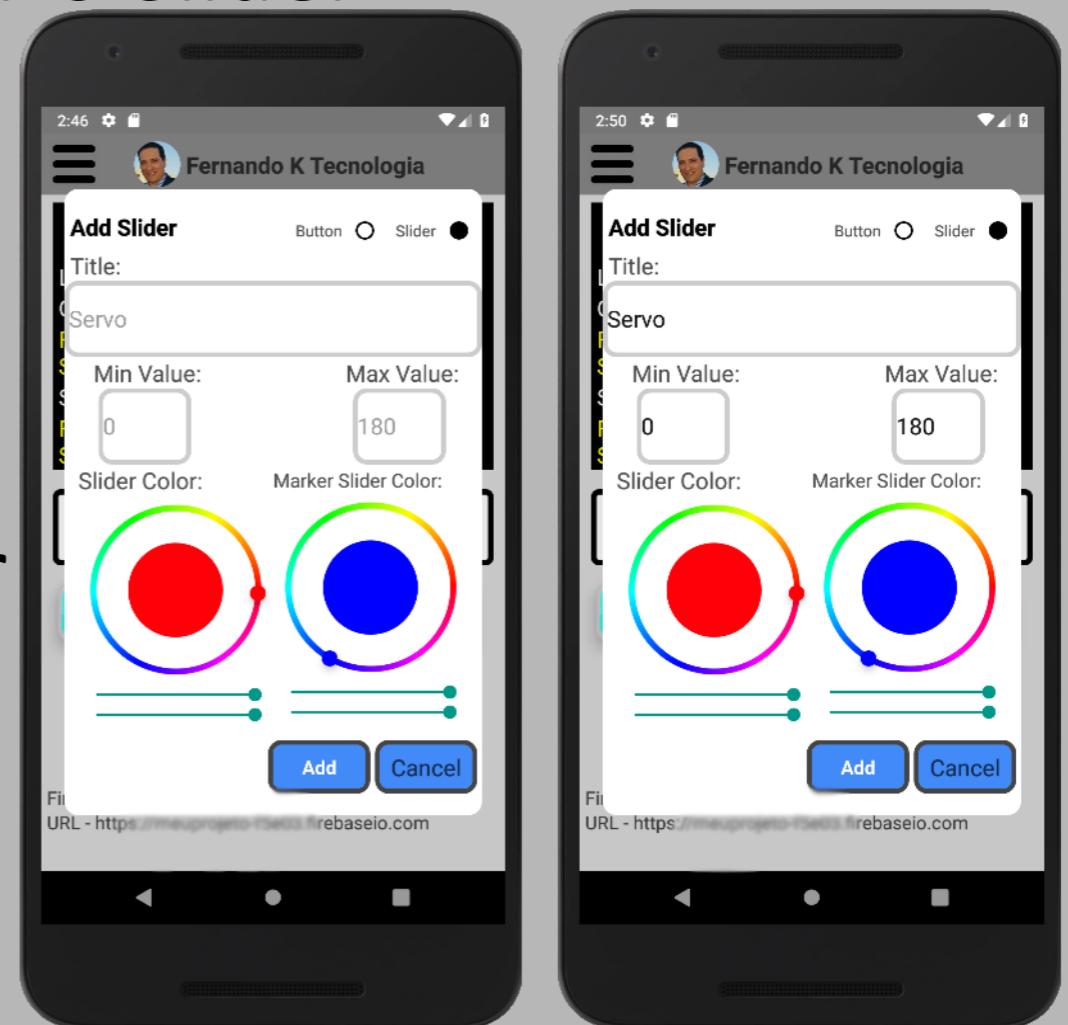
Min Value: Valor Mínimo.

Max Value: Valor Máximo.

Slider Color: Cor do Slider.

Marker Slider Color: Cor do Marcador

Add: Adiciona o slider.



Adicionar novo Slider

Após clicar em Add um novo slider será criado conforme a figura 2.

Figura 1

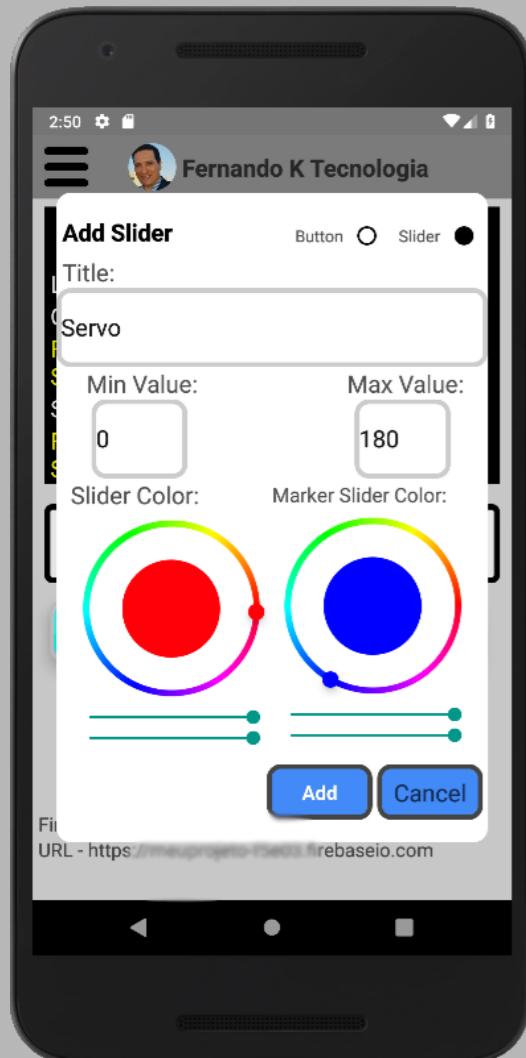
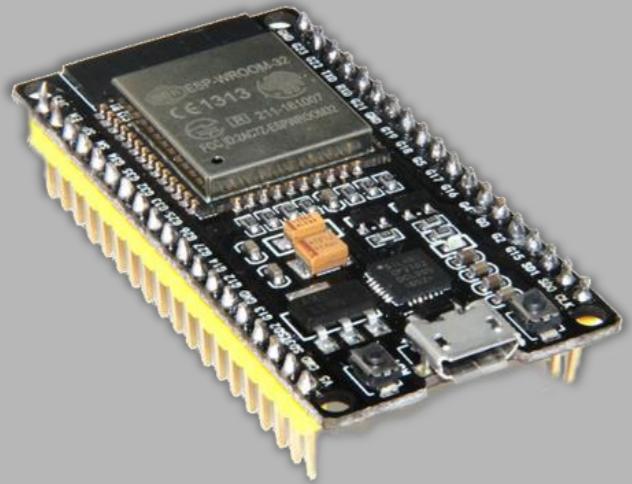


Figura 2



*O nome do Slider será mandando junto com o valor atual para o ESP como um comando quando sofrer alteração.



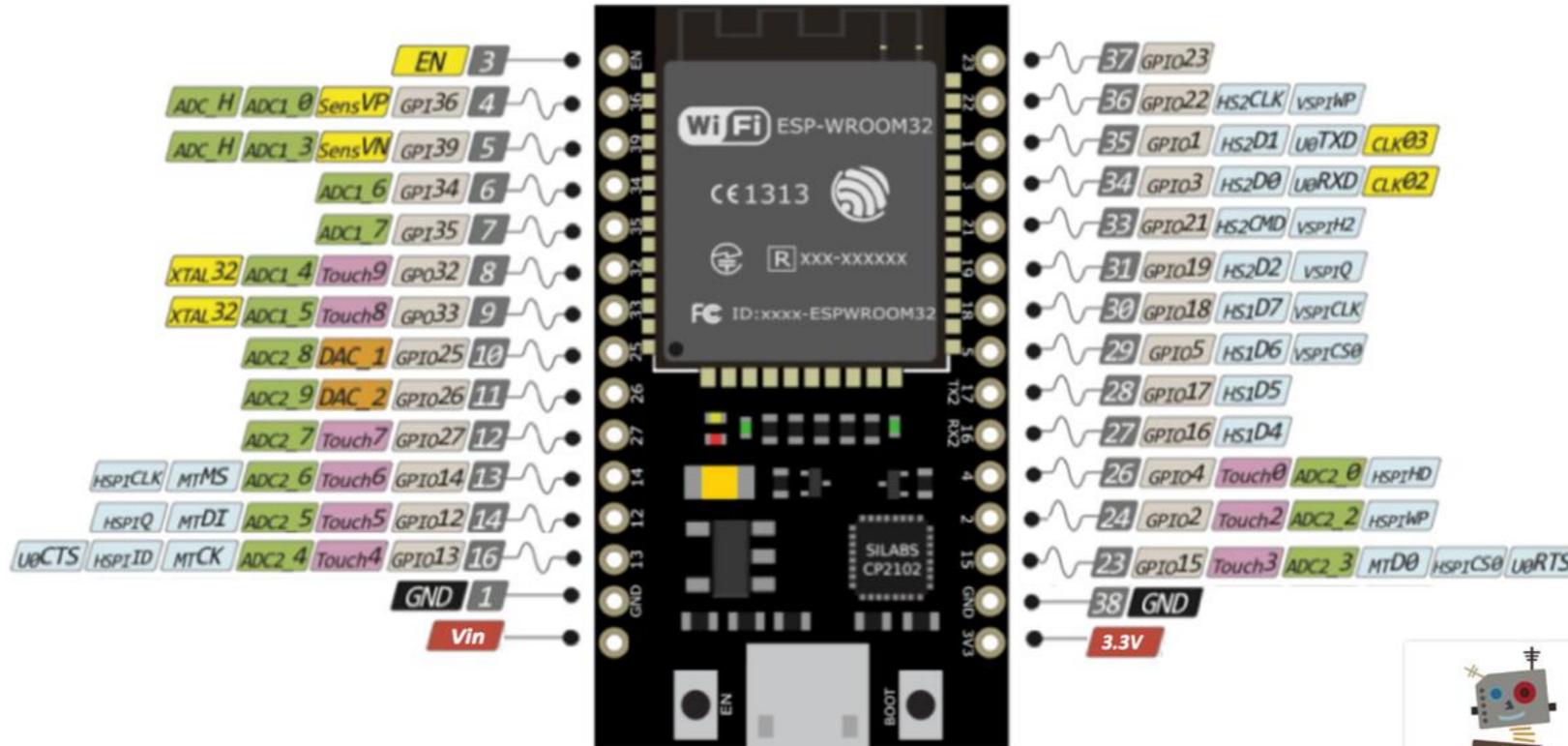
Montagem

Pinout 30 pinos

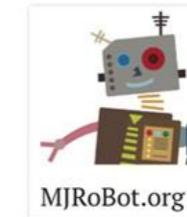
Modelo usado nesta aula



ESP32 DEVELOPMENT BOARD DUAL CORE ESP-32 & ESP-32S BOARD



Freely adapted by <https://MJRoBot.org>



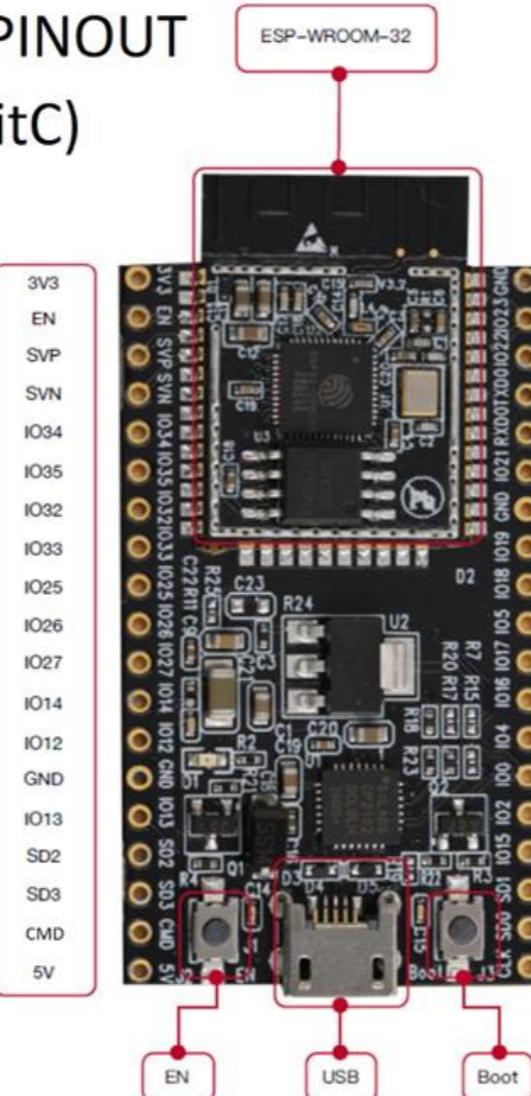
Pinout 38 pinos



ESP32-WROOM-32 PINOUT (aka ESP32-DevKitC)

	Power
	GND
	Serial Pin
	Analog Pin
	Control
	Physical Pin
	Port Pin
	Touch Pin
	DAC Pin
	PWM Pin

JTAG RESET		ChipPU	3.3V
ADC PA	RTC100	ADC1_0 SensVP	GPIO36 5
ADC PA	RTC103	ADC1_3 SensVN	GPIO39 8
RTC104	ADC1_6 VDET1	GPIO34 10	
RTC105	ADC1_7 VDET2	GPIO35 11	
XTAL32	Touch9 RTC109 ADC1_4	GPIO32 12	
XTAL32	Touch8 RTC108 ADC1_5	GPIO33 13	
DAC_1	RTC106 ADC2_8 EMACRXD0	GPIO25 14	
DAC_2	RTC107 ADC2_9 EMACRXD1	GPIO26 15	
Touch7	RTC1017 ADC2_7 EMACRXM	GPIO27 16	
TMS	HS2CLK SDCLK HSPICLK MTMS	Touch6 RTC1016 ADC2_6 EMACRXD2	GPIO14 17
TDI	HS2DATA2 SDATA2 HSPIDQ MTDI	Touch5 RTC1015 ADC2_5 EMACRXD3	GPIO12 18
TCK	HS2DATA3 SDATA3 HSPID	MTCK Touch4 RTC1014 ADC2_4 EMACRXE	GPIO13 20
		FLASH D2 SDATA2 HS1DATA2 U1RXD SPIHD GPIO9 28	
		FLASH D3 SDATA2 HS1DATA2 U1TXD SPINP GPIO10 29	
		FLASH CMD SDMD HS1CMD U1RTS SP1CS0 GPIO11 30	



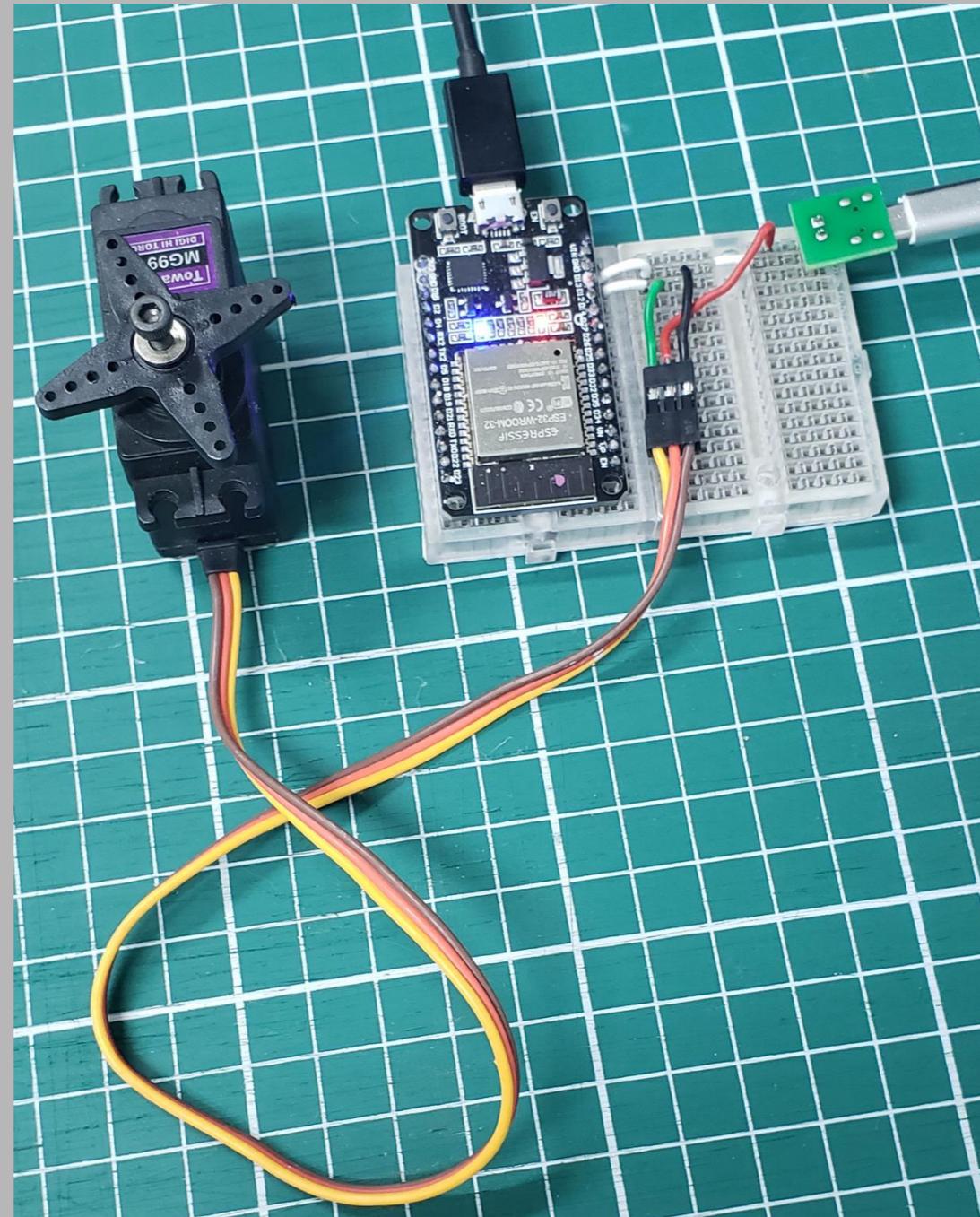
Adapted from data at:

http://www.espressif.com/sites/default/files/documentation/esp_wroom_32_datasheet_en.pdf
https://cdn-shop.adafruit.com/product-files/3269/pinout_wroom_pinout.png

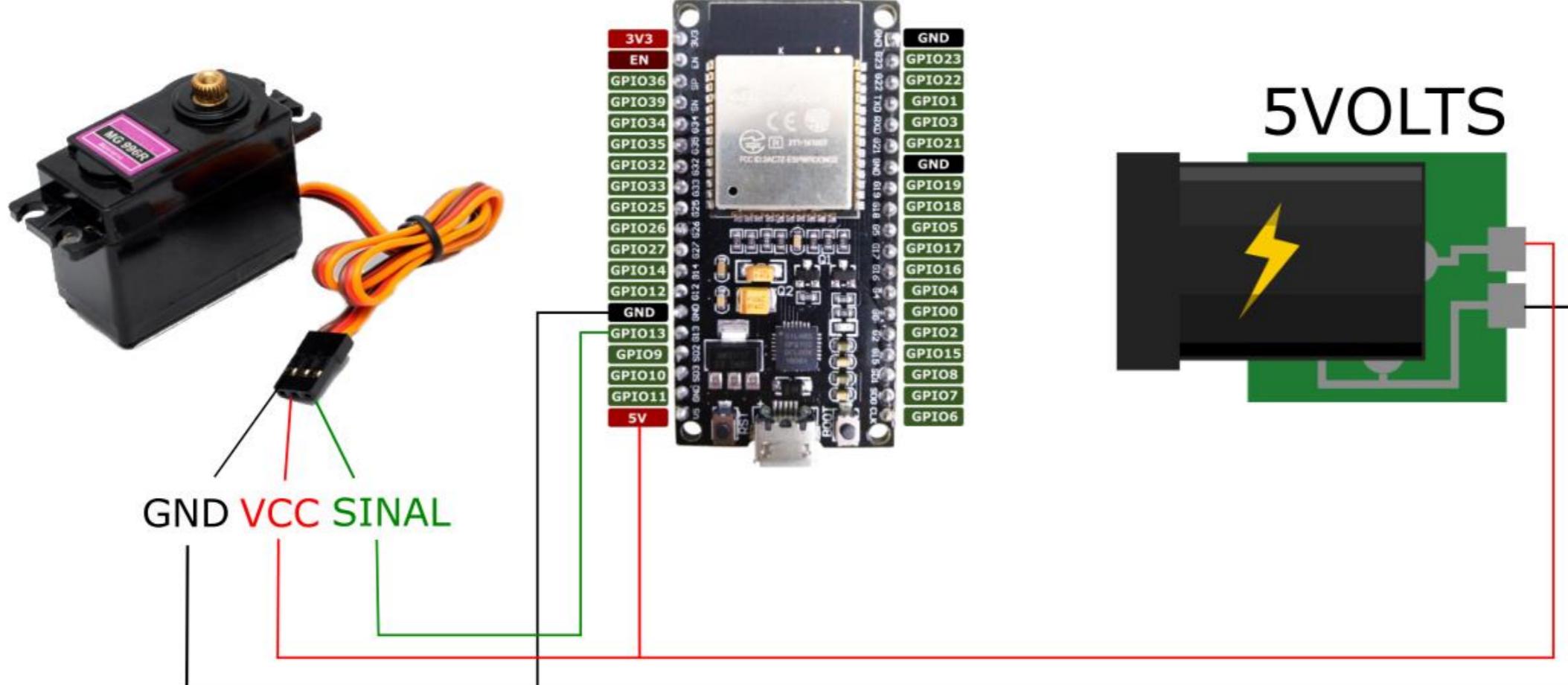
JTAG 20-pin		
Red	3V3	VTref
Gray	RSTn	TRST
Orange	IO12	TDI
Yellow	IO14	TMS
Blue	IO13	TCK
Ito GND	RTCK	
Purple	IO15	TDO
	RESET	
	DBGRQ	
	5V-Supply	
1 ● ● 2		---
3 ● ● 4	GND	GND
5 ● ● 6	GND	Black
7 ● ● 8	GND	
9 ● ● 10	GND	Green
11 ● ● 12	GND	White
13 ● ● 14	GND	
15 ● ● 16	GND	
17 ● ● 18	GND	
19 ● ● 20	GND	
Pins 14, 16, 18, 20: On some models like the high-end model J-Link PRO, these pins may not be connected to GND but are reserved for future use/extension. In case of doubt, leave open on target hardware.		

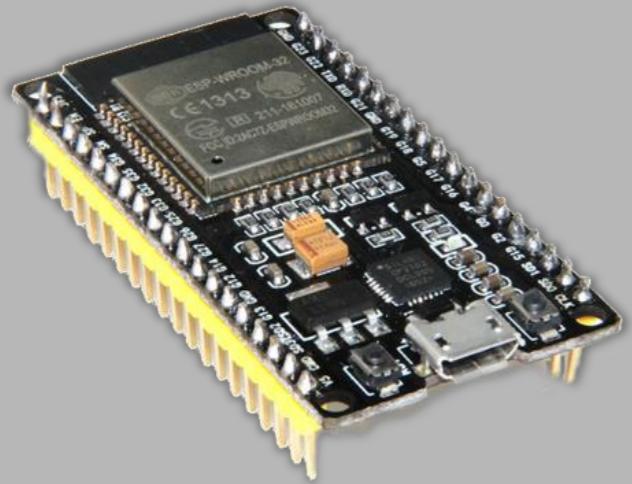
GND	●	GND
IO23	●	SPI MOSI
IO22	●	Wire SCL
TXD0	●	PROGRAM Port
RXD0	●	
IO21	●	Wire SDA
GND	●	NC
IO19	●	SPI MISO
IO18	●	SPI SCK
IO5	●	SPI SS
IO17	●	
IO16	●	
IO4	●	
IO10	●	
IO2	●	
IO15	●	TDO
SD1	●	FLASH D1
SD0	●	FLASH D0
CLK	●	FLASH SCK

Montagem



Montagem

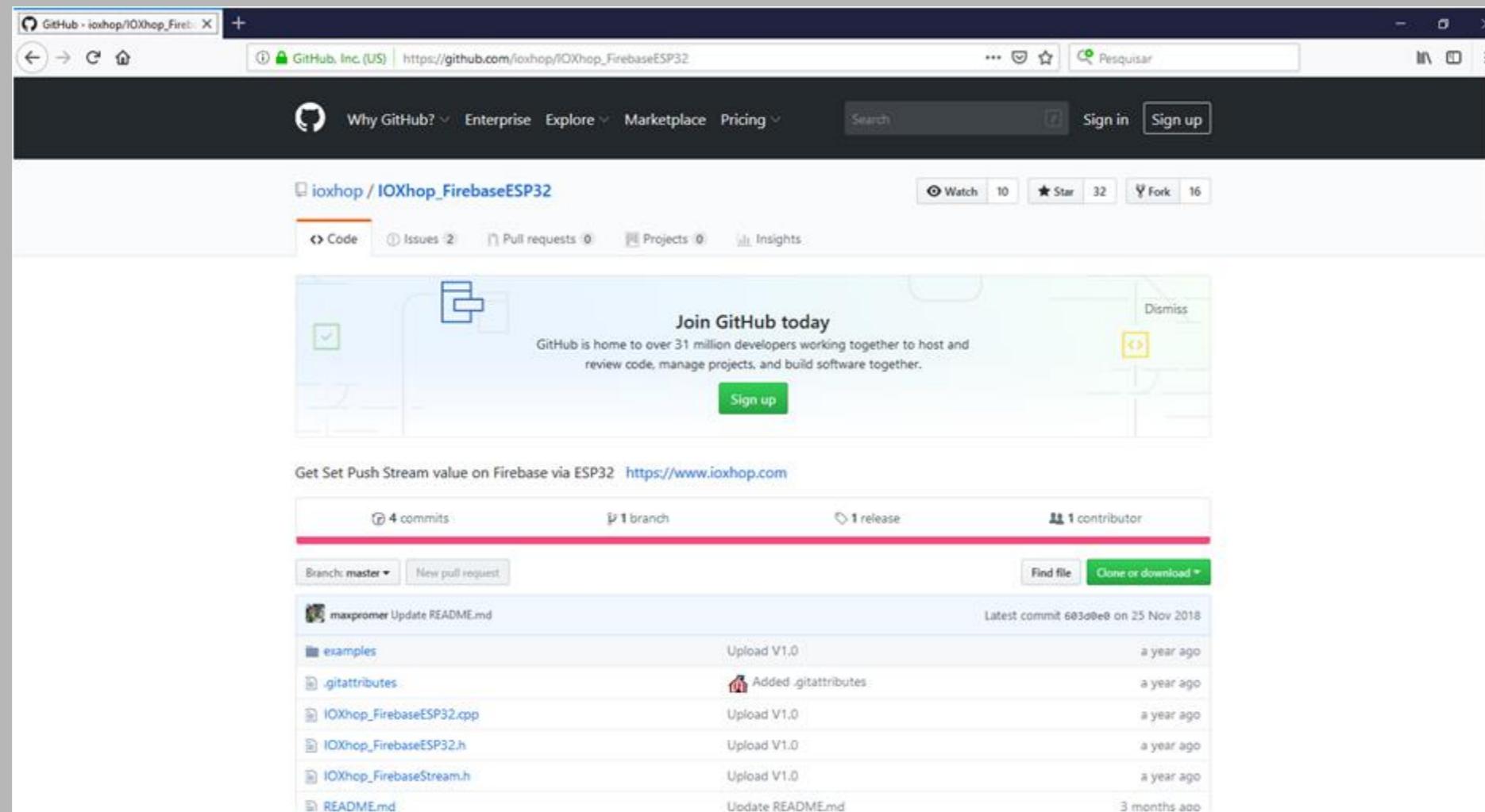




Instalação das Bibliotecas

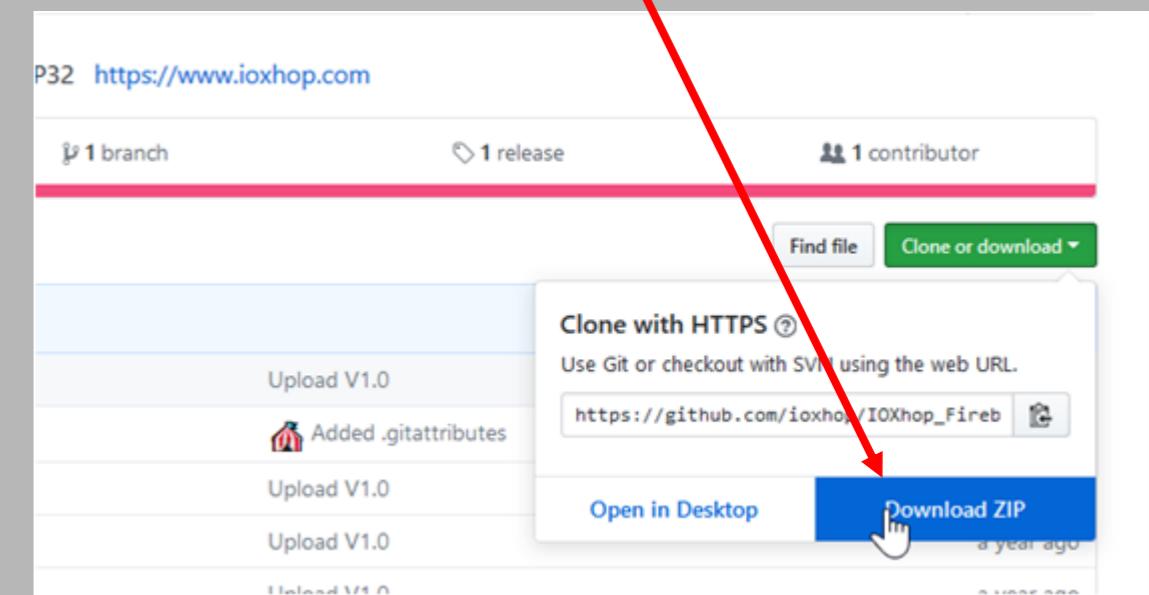
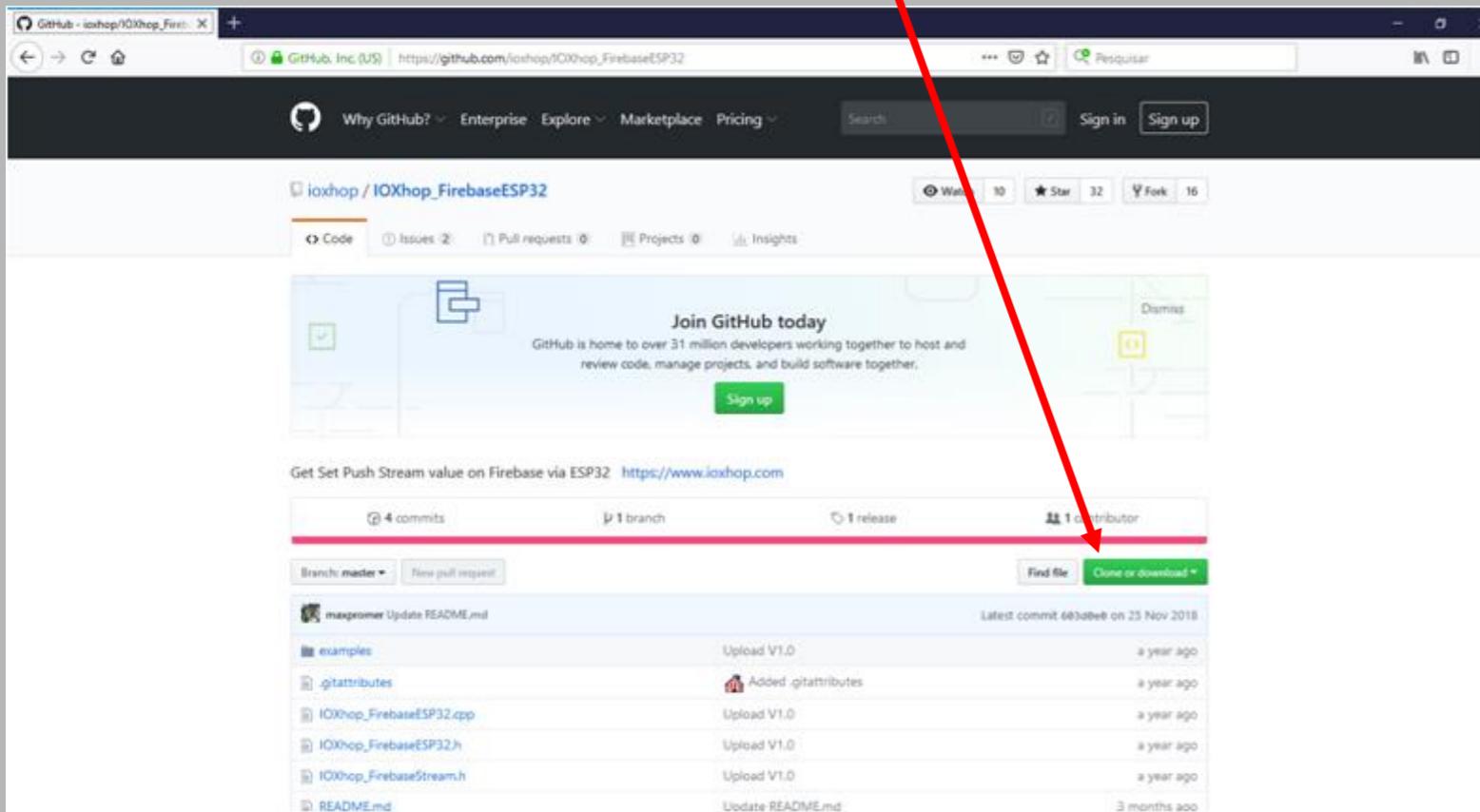
Instalação da Biblioteca Firebase

É necessário instalar a biblioteca do Firebase para o ESP32. Entre no link clicando [aqui](#).



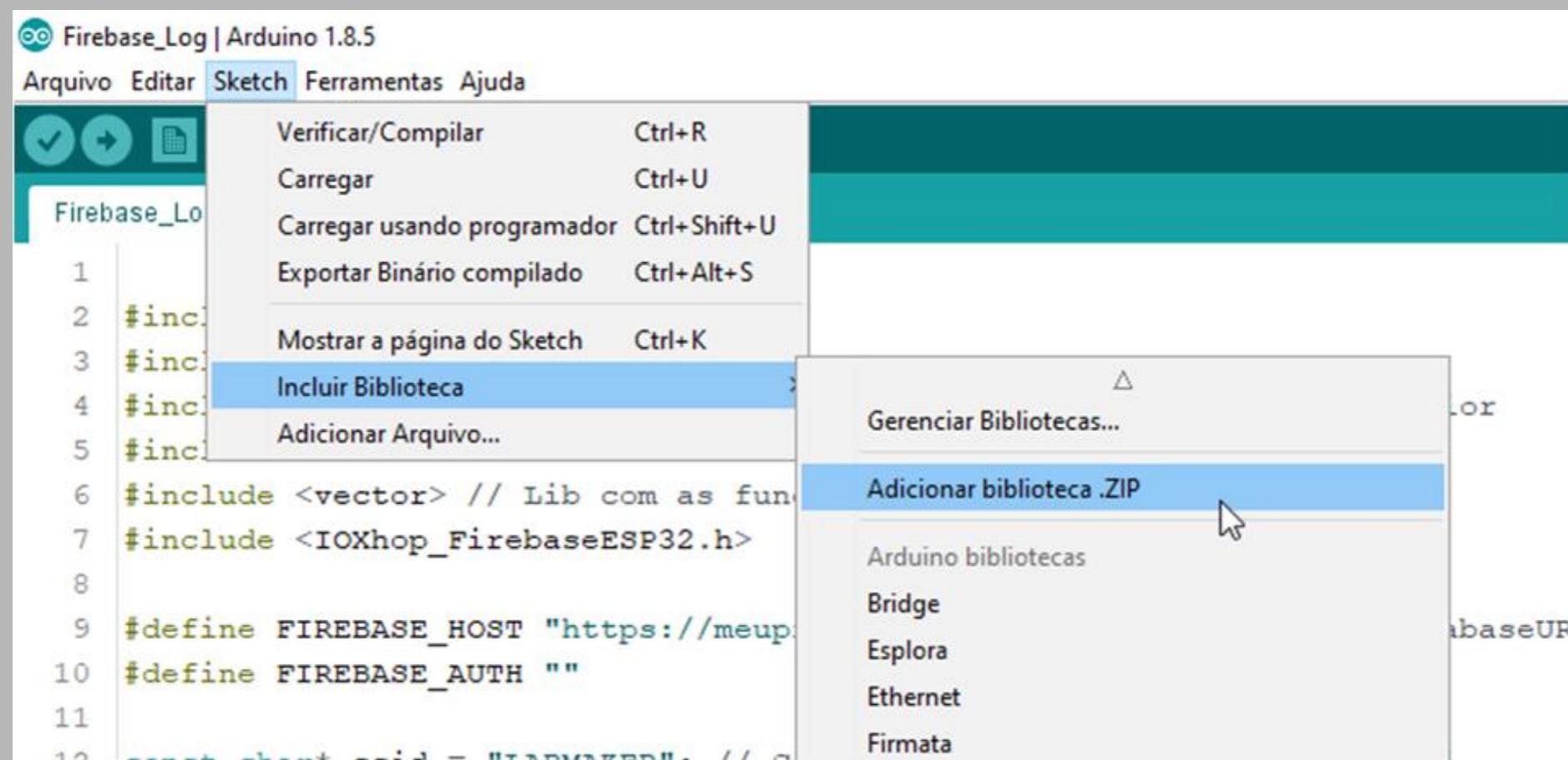
Instalação da Biblioteca Firebase

Clique em “Clone or Download” e depois “Download ZIP” para baixar o ZIP.



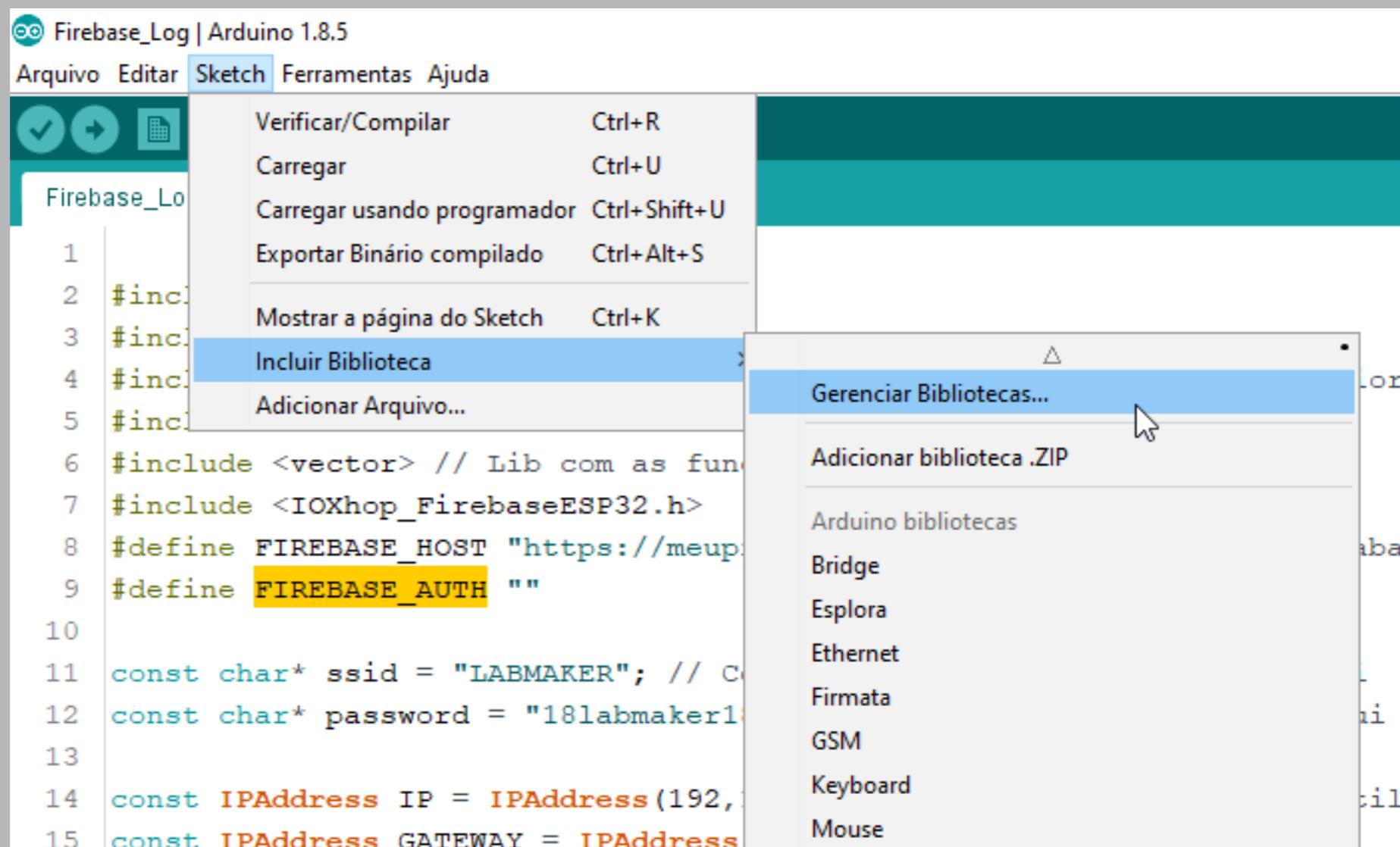
Instalação da Biblioteca Firebase

Na IDE do Arduino vá em Sketch->Incluir Biblioteca->Adicionar biblioteca .ZIP. Navegue até a pasta que baixou a biblioteca e adicione ela.



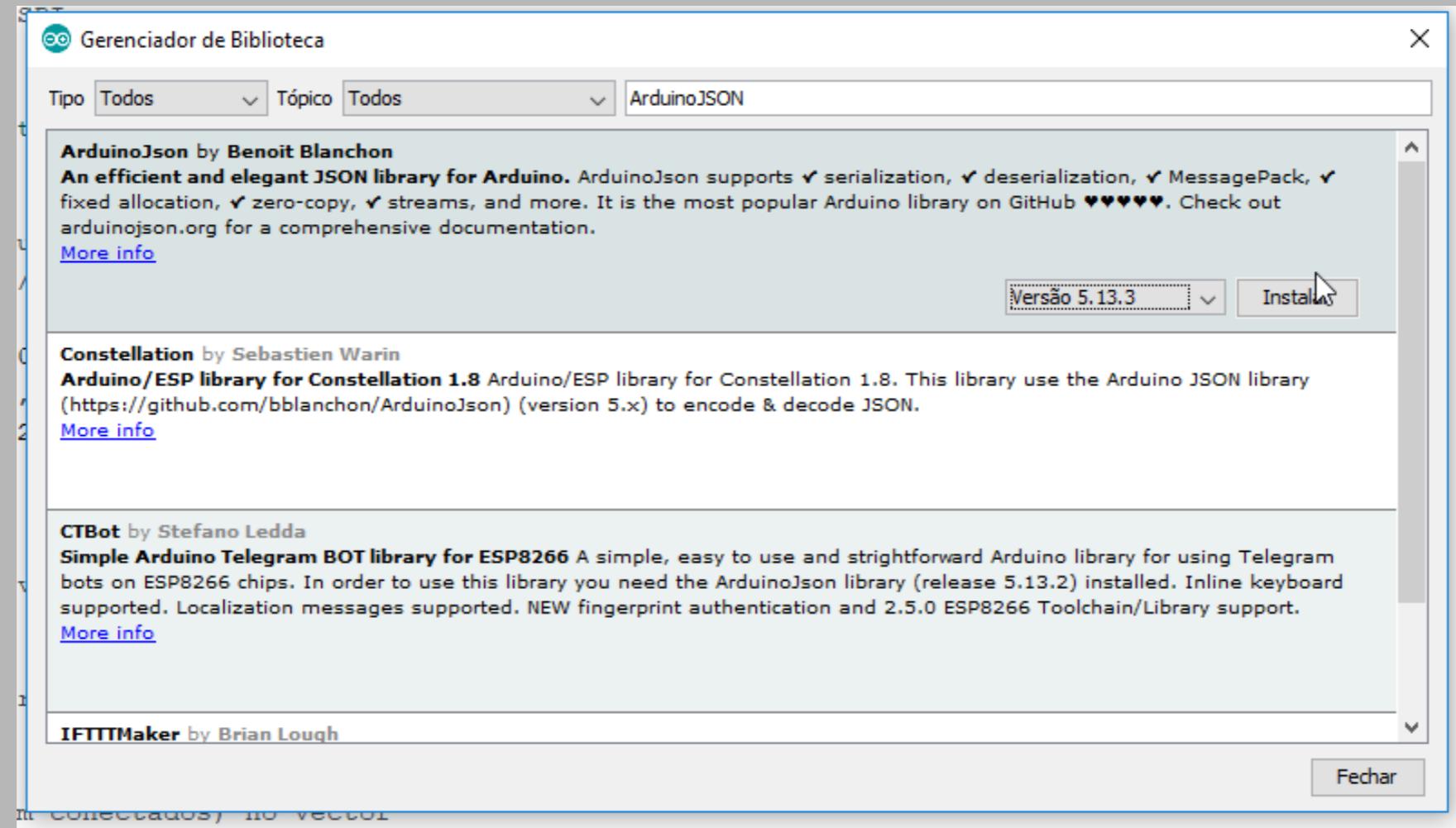
Instalação da Biblioteca ArduinoJSON

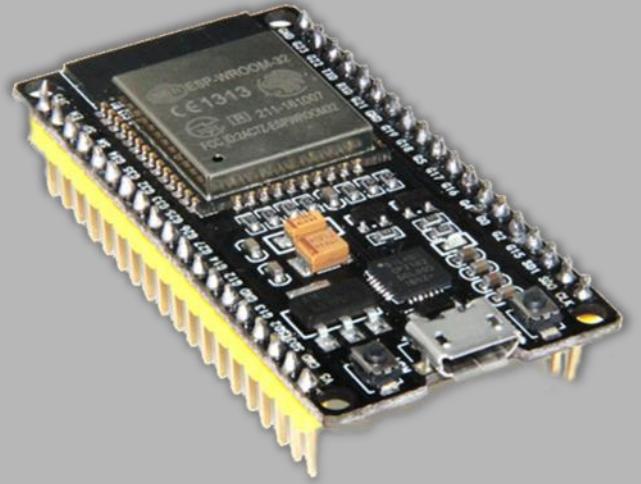
Na IDE do Arduino vá em Sketch->Incluir Biblioteca->Gerenciar bibliotecas



Instalação da Biblioteca ArduinoJSON

Pesquise a biblioteca **ArduinoJSON**, selecione a primeira feita por **Benoit Blanchon**, selecione a versão **5.13.3** e clique em **Instalar**.





Código

Código Declarações e variáveis

```
#include <Arduino.h> //Lib Arduino
#include <WiFi.h> // Lib WiFi
#include <Wire.h> // Necessário apenas para o Arduino 1.6.5 e posterior
#include <SPI.h> // Lib de comunicação SPI
#include <vector> // Lib com as funções de vetor (vector)
#include <IOXhop_FirebaseESP32.h> //Lib Firebase
#define FIREBASE_HOST "https://NomeDoProjeto.firebaseio.com" //databaseURL fornecido pelo
Firebase
#define FIREBASE_AUTH ""
#include "esp_task_wdt.h"
const char* ssid = "NomeRedeWifi"; // Coloque o nome da sua rede wifi aqui
const char* password = "Senha1234"; // Coloque a sua senha wifi aqui

const IPAddress IP = IPAddress(192,168,0,141); // IP fixo que o ESP utilizará
const IPAddress GATEWAY = IPAddress(192,168,0,1); // Gateway
const IPAddress SUBNET = IPAddress(255,255,255,0); // Máscara
// Google Public DNS represents two IP addresses for IPv4 - 8.8.8.8 and 8.8.4.4
const IPAddress PRIMARY_DNS(8, 8, 8, 8);
const IPAddress SECONDARY_DNS(8, 8, 4, 4);

const int port = 80; // Porta
```

Código Declarações e variáveis (Continuação)

```
// Objeto WiFi Server, o ESP será o servidor
WiFiServer server(port);

// Vetor com os clientes que se conectarão no ESP
std::vector<WiFiClient> clients;

// Task que insere novos clientes (recém conectados) no vector
void taskNewClients(void *);

// Task que recebe executa comandos dos clientes
void handleClients(void *);

int LED_BUILTIN = 2;
const int freq = 50;
const int canal = 0;
const int resolucao = 12;

const int pinoAtuacao = 13;
int ciclo = 200;
```

Código Setup

```
void setup()
{
    // Iniciamos a serial com 115200 de velocidade
    Serial.begin(115200);

    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(pinoAtuacao, OUTPUT);
    ledcSetup(canal, freq, resolucao);
    ledcAttachPin(pinoAtuacao, canal);
    //Iniciamos o valor do servo com 150, que se equivale a 0.
    ledcWrite(canal, 150);

    // Iniciamos o servidor (WiFi Server)
    serverBegin();
    // Criamos 3 tasks (mais detalhes no escopo da função)
    createTasks();
    //Inicia a conexão com o Firebase
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    //Muda o valor do Servo no firebase para 0 para iniciar.
    Firebase.setInt("Servo", 0);
```

Código Setup (Continuação)

```
//Callback quando tem alguma alteração no firebase
Firebase.stream("/", [](FirebaseStream stream){
    //Entra quando o evento que vem do callback é de update(put)
    if (stream.getEvent() == "put")
    {
        //Verifica se o nome é o mesmo do Servo. O nome tem que ser igual com o nome no Firebase e no App.
        if (stream.getPath() == "/Servo")
        {
            //Muda o valor do Servo de acordo com o que foi recebido e é feito o mapeamento do valor.
            ledcWrite(canal, map(stream.getDataInt(),0,180,150,450));
        }
        else
        {
            if(stream.getPath() == "/Led")
            {
                String cmd;
                cmd= stream.getDataString(); //Pega o valor recebido
                cmd.toUpperCase(); //Colocar tudo em maiúsculo
```

Código Setup (Continuação)

```
if(cmd=="ON"){//Se o comando for On, ligar o LED
    digitalWrite(LED_BUILTIN, HIGH);
}
else{
    if(cmd=="OFF"){//Se o comando for Off, Desligar o LED
        digitalWrite(LED_BUILTIN, LOW);
    }
}
}
});
```

Código Loop

```
void loop()
{
    // Exibimos o total de clientes conectados
    showClients();
    delay(5000);
}
```

Código ServerBegin

```
// Conectamos no WiFi e iniciamos o servidor
void serverBegin()
{
    WiFi.begin(ssid, password); // Iniciamos o WiFi
    Serial.println("Connecting to WiFi");
    // Enquanto não estiver conectado exibimos um ponto
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(1000);
    }
    Serial.println("OK");// Exibimos na serial
    //Configuramos o WiFi com o IP definido anteriormente
    if (!WiFi.config(IP, GATEWAY, SUBNET, PRIMARY_DNS, SECONDARY_DNS))
    {
        Serial.println("STA Failed to configure");
        while(1);
    }
    server.begin(port);// Iniciamos o servidor
}
```

Código *Create Tasks*

```
// Criamos as 2 Tasks
void createTasks()
{
    //Criamos a task que insere os novos clientes no vector
    xTaskCreatePinnedToCore(taskNewClients, "taskNewClients", 10000, NULL, 2, NULL, 1);

    //Criamos a task que recebe e executa os comandos dos clients conectados
    xTaskCreatePinnedToCore(handleClients, "handleClients", 10000, NULL, 2, NULL, 1);
}
```

Código Task 2: Core 0 - taskNewClients

```
// Task que insere novos clientes conectados no vector
void taskNewClients(void *p)
{
    // Objeto WiFiClient que receberá o novo cliente conectado
    WiFiClient newClient;
    // Tempo esperado no delay da task (1 ms)
    TickType_t taskDelay = 1 / portTICK_PERIOD_MS;
    while(true)
    {
        // Se existir um novo client atribuimos para a variável
        newClient = server.available();
        // Se o client for diferente de nulo
        if(newClient)
        {
            // Inserimos no vector
            clients.push_back(newClient);
            // Exibimos na serial indicando novo client e a quantidade atual de clients
            Serial.println("New client! size:"+String(clients.size()));
        }
        // Aguardamos 1ms
        vTaskDelay(taskDelay);
    }
}
```

Verificação de novos clientes

Recepção e inserção de novos clientes (Vector)

Código Task 3: Core 0 - handleClients

```
// Função que verifica se um cliente enviou um comando
void handleClients(void *p)
{
    String cmd; // String que receberá o comando
    TickType_t taskDelay = 1 / portTICK_PERIOD_MS; // Tempo aguardado pela task (1 ms)
    while(true) // Loop infinito
    {
        // Atualizamos o vector deixando somente os clientes conectados
        → refreshConnections();
        // Percorremos o vector
        for(int i=0; i<clients.size(); i++)
        {
            if(clients[i].available()) // Se existir dados a serem lidos
            {
                → cmd = clients[i].readStringUntil('\n'); // Recebemos a String até o '\n'
                // Executamos um comando, enviando por parâmetro a String cmd, e o client que nos enviou o comando
                → executeCommand(cmd, clients[i]);
            }
            esp_task_wdt_reset(); // Reset Whatchdog
        }
        vTaskDelay(taskDelay); // Delay de 1 ms
    }
}
```

Atualização de conexões

Recepção de comando

Execução de comando

Envio de respostas

Código *refreshConnections* (função chamada pela task *handleClients*)

```
// Função que verifica se um ou mais clients se desconectaram do server e, se sim, estes clients serão retirados
// do vector
void refreshConnections()
{
    bool flag = false; // Flag que indica se pelo menos um client se desconectou
    std::vector<WiFiClient> newVector; // Objeto que receberá apenas os clients conectados
    for(int i=0; i<clients.size(); i++) // Percorremos o vector
    {
        if(!clients[i].connected()) // Verificamos se o client está desconectado
        {
            // Exibimos na serial que um cliente se desconectou e a posição em que ele está no vector
            Serial.println("Client disconnected! ["+String(i)+"]");
            clients[i].stop(); // Desconectamos o client
            flag = true; // Setamos a flag como true indicando que o vector foi alterado
        }
        else
            newVector.push_back(clients[i]); // Se o client está conectado, adicionamos no newVector
    }
    // Se pelo menos um client se desconectou, atribuimos ao vector "clients" os clients de "newVector"
    if(flag)
        clients = newVector;
}
```

Código executeCommand (função chamada pela task handleClients)

```
// Função que executa um comando de acordo com a String "cmd"
void executeCommand(String cmd, WiFiClient client)
{
    // String que guarda a resposta que será enviada para o client
    String response = "";

    // Se a String estiver vazia, abortamos a função
    if (cmd.equals(""))
        return;

    // Exibimos o comando recebido na serial e no
    Serial.println("Recebido: ["+cmd+"]");
    // Deixamos a string toda em maiúsculo
    cmd.toUpperCase();
    //Verifica se o comando é destinado ao Servo
    if(cmd.indexOf("SERVO")==0)
    {
```

Código executeCommand (função chamada pela task handleClients)

```
//Recebe o grau de acordo com o valor após o espaço " "
int grau = atoi(cmd.substring(cmd.indexOf(" ")+1).c_str()));

//Mapeia o valor recebido para o valor do Servo
ciclo = map(grau,0,180,150,450);
ledcWrite(canal, ciclo);

response = "ok";
}

else{
    if(cmd.indexOf("LED ON")==0){
        digitalWrite(LED_BUILTIN, HIGH);
        response = "ok";
    }
    else{
        if(cmd.indexOf("LED OFF")==0){
            digitalWrite(LED_BUILTIN, LOW);
            response = "ok";
        }
    }
}
```

Código *executeCommand* (função chamada pela task *handleClients*)

```
else{
    response = "Invalid Command";
}
}

// Enviamos para o client passado por parâmetro e exibimos sucesso ou falha na serial
if(client.print(response)>0)
    Serial.println("Resposta enviada");
else
    Serial.println("Erro ao enviar resposta");
}
```

Em www.fernandok.com

Download arquivos PDF e INO do código fonte



Inscreva-se

