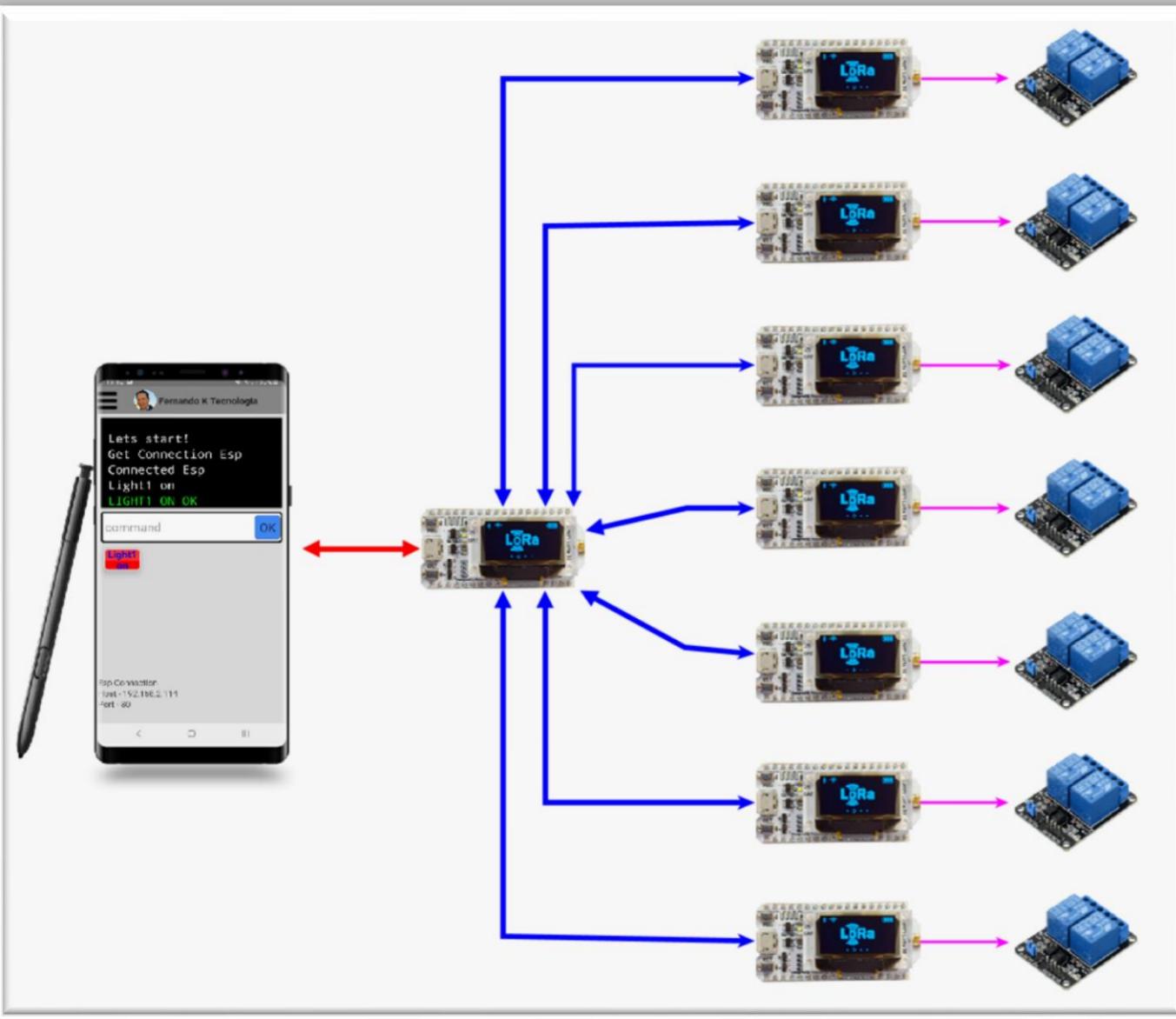


Automação LoRa e APP Fernando K

Esp32 LoRa – Multi Client – Gateway



Por Fernando Koyanagi

Intenção dessa aula

Modificar o estado de relês utilizando :

APP Fernando K

Esp32 LoRa Gateway

Esp32 LoRa Multi Client

Recursos usados

- Heltec WiFi LoRa 32
- Jumpers
- Relê
- Fonte





Robô de desenho XY

by Fernando K - 19 fevereiro

Temos aqui hoje um projeto de mecatrônica que é uma derivação de um vídeo que eu já lancei aqui: ROUTER E PLOTTER WIFI COM WEB SERVER ...

[Leia mais](#)



E se o seu link cair?

by Fernando K - 12 fevereiro

Neste vídeo vamos criar um sensor de queda de link com um ESP32 e um SIM800. Isso significa que, com este projeto, poderemos verificar ...

[Leia mais](#)



Tragédia de Brumadinho: sugestão para um sistema de alerta!

by Fernando K - 08 fevereiro

Como a Internet das Coisas pode auxiliar em tragédias como a de Mariana e Brumadinho? Hoje quero trazer uma sugestão de um sistema ...

[Leia mais](#)



Router e Plotter WiFi com Webserver em ESP32

by Fernando K - 05 fevereiro

Já gravei vídeos sobre plotter com o Raspberry Pi e com Laser, mas, hoje, quero falar de uma versão com GRBL e ESP32. Como acredito e...

[Leia mais](#)



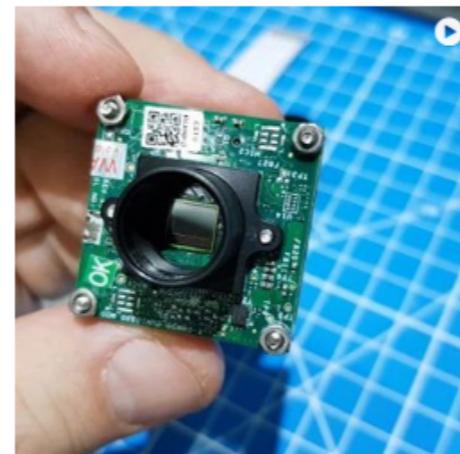
Links onde comprei os componentes

Em www.fernandok.com

Supergrupo de colaboração entre meus seguidores.

Conteúdo exclusivo, que não tem no Youtube!

INSTAGRAM @FERNANDOK_OFICIAL



FACEBOOK



INSCREVA-SE NO YOUTUBE

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 Sorororcem 05/Oct/2018, 10:55
ESP32 Projetos de ESP32	29	62	Dúvidas sobre como instalar a... por Marcos Sarge 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: Sorororcem

[L...](#) [☰](#)

Powered by phpBB® Forum Software © phpBB Limited
Traduzido por: Suporte phpBB
Painel de Controle da Administração



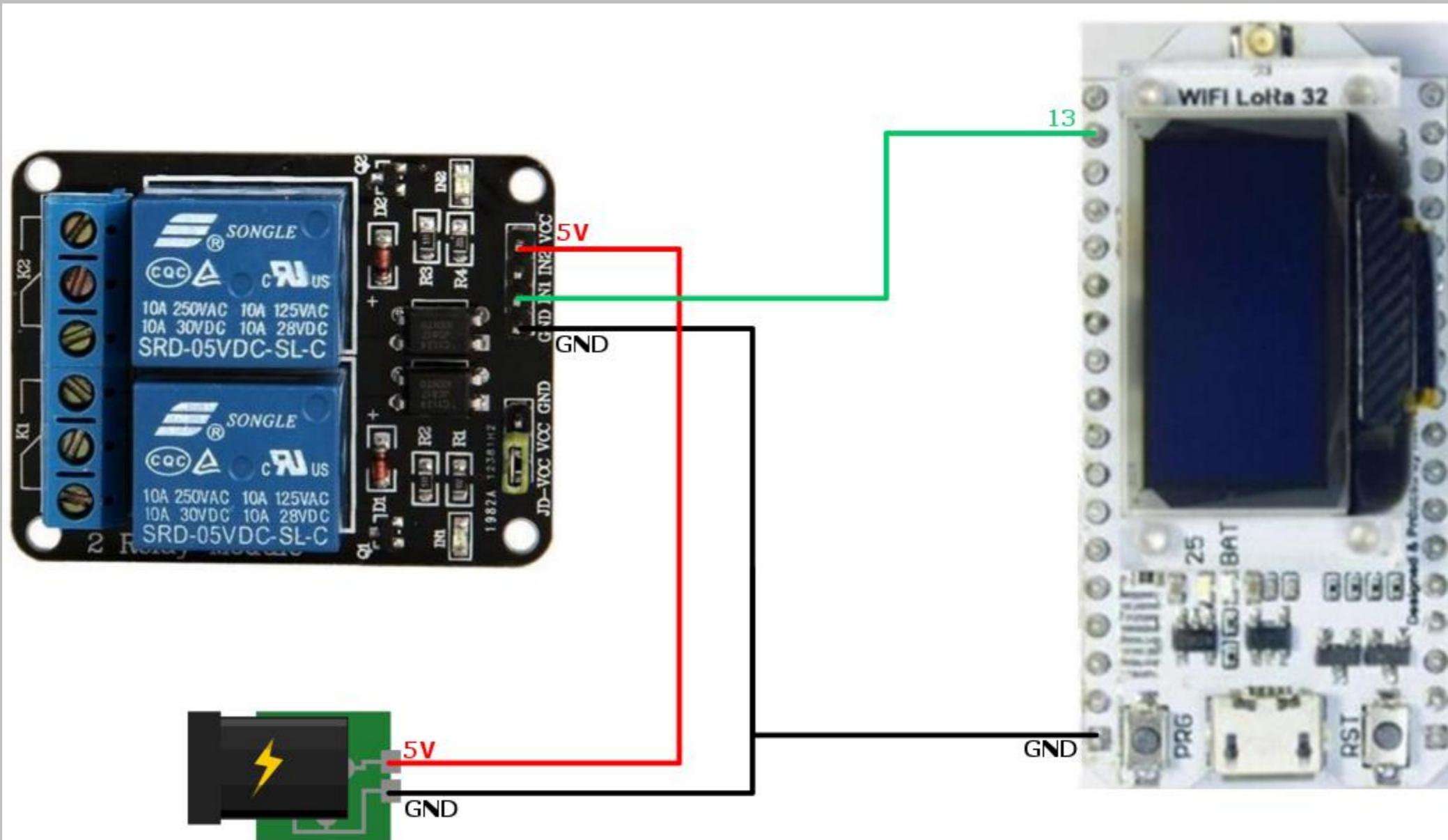
Instagram
fernandok_oficial



Telegram
fernandok_oficial



Montagem



Bibliotecas

Heltec ESP32 Dev-Boards

Gerenciador de Biblioteca X

Tipo: Todos Tópico: Todos Busca: heltec

Heltec ESP32 Dev-Boards
by Heltec Automation Versão 1.0.9 **INSTALLED**
Library for Heltec ESP32 (or ESP32+LoRa) based boards Include, WiFi Kit 32, WiFi LoRa 32, Wireless Stick, Wireless Shell, see more on <http://heltec.cn>
[More info](#)
[Selecionar versão](#) [Instalar](#)

Heltec ESP8266 Dev-Boards
by Heltec Automation
Library for Heltec ESP8266 based boards Include, WiFi Kit 8, see more on <http://heltec.cn>
[More info](#)

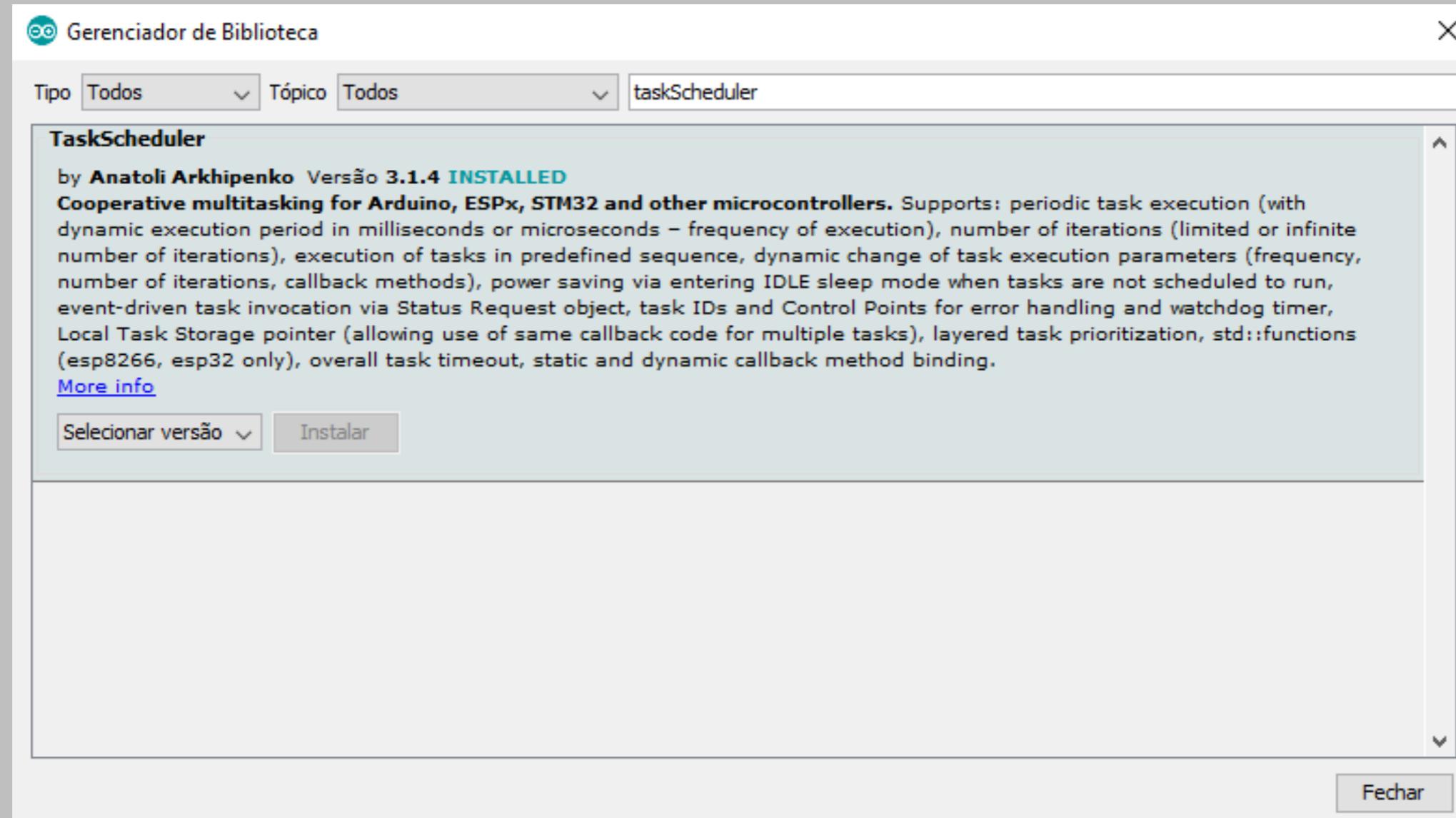
TTN_esp32
by Francois Riotte
ESP 32 port of the Arduino TheThingsNetwork library. Supports Heltec Wifi Lora 32 boards
[More info](#)

[Fechar](#)

https://github.com/HelTecAutomation/Heltec_ESP32

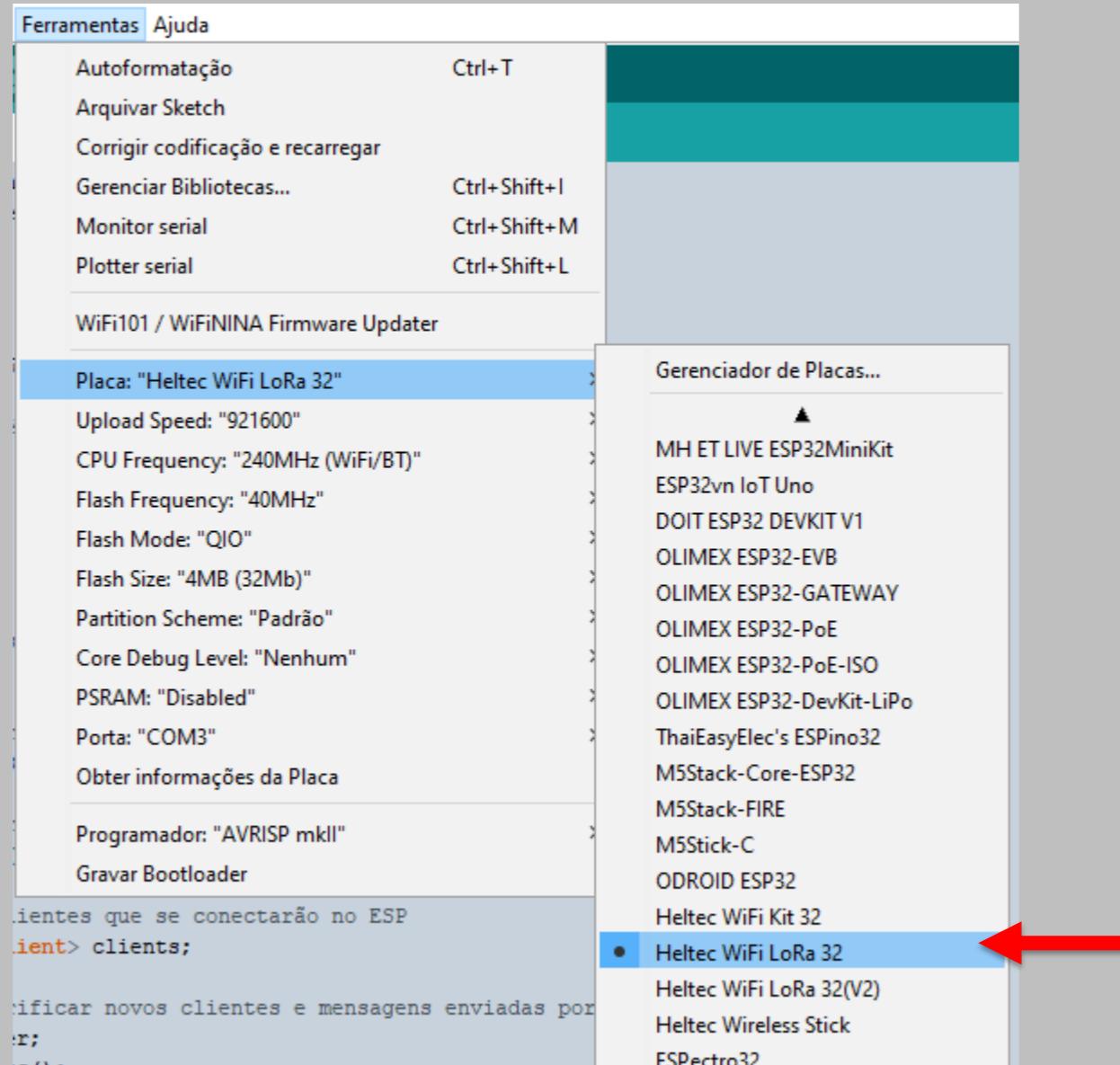
Biblioteca

TaskScheduler



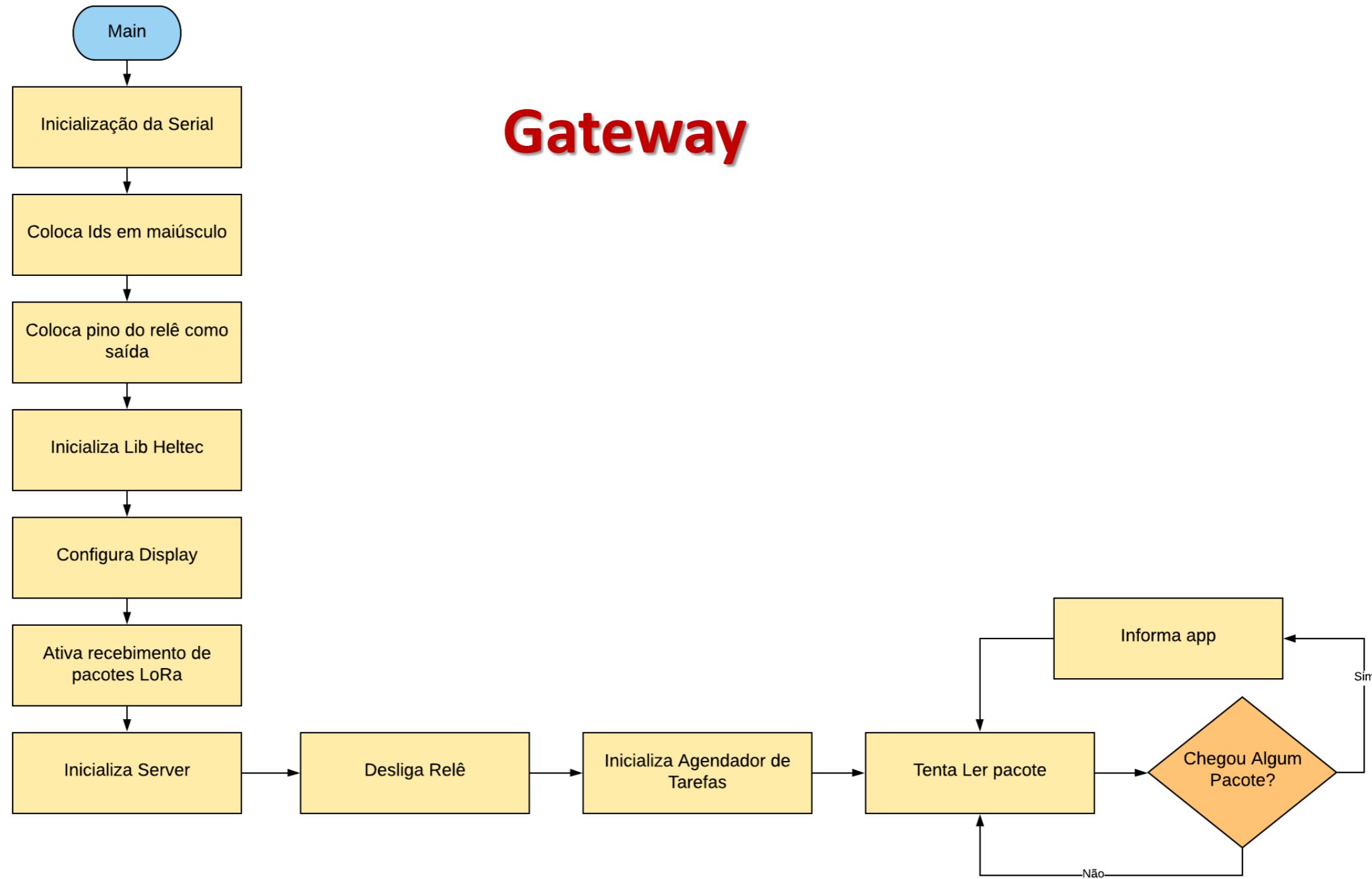
<https://github.com/arkhipenko/TaskScheduler>

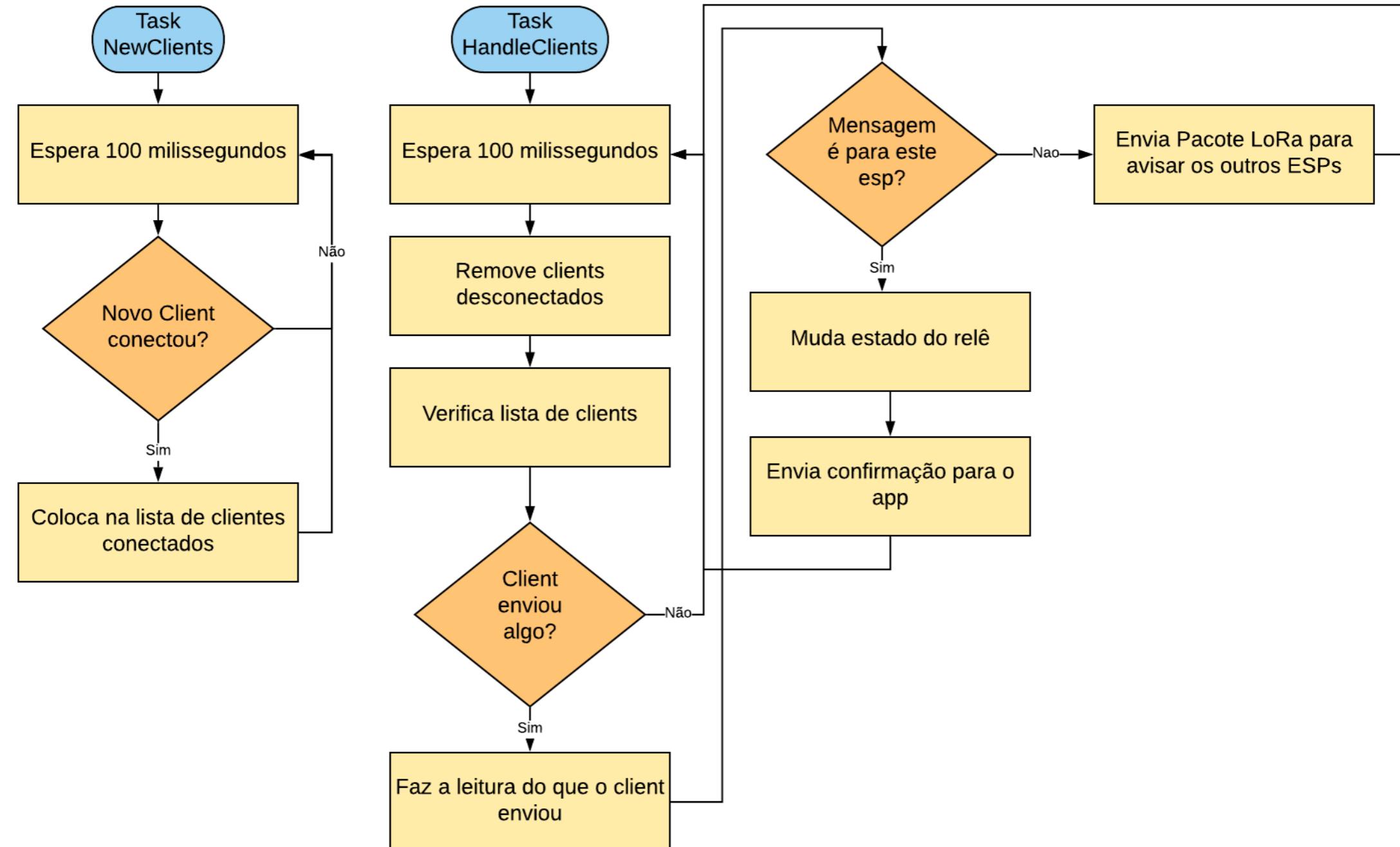
Seleção de Placa Ferramentas->Placa->Heltec WiFi LoRa 32



Código Gateway App *Fluxograma*

Gateway





Código Gateway App

Código Declarações e variáveis

```
#include <heltec.h>
#include <TaskScheduler.h>
#include <vector>
#include <WiFi.h>

//Frequência
#define BAND 433E6

//Pino onde o relê está
#define RELAY 13

//SSID e senha do roteador que vc usa em sua casa
#define SSID      "SSID"
#define PASSWORD "PASSWORD"

//Objeto que vamos utilizar para guardar o ip recebido
IPAddress myIP;

// Porta do server que vc vai utilizar para conectar pelo aplicativo
const int port = 80;

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

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

Código Declarações e variáveis

```
//Tarefas para verificar novos clientes e mensagens enviadas por estes
Scheduler scheduler;
void taskNewClients();
void taskHandleClients();
//Tarefa para verificar se uma nova conexão feita por aplicativo está sendo feita
Task t1(100, TASK_FOREVER, &taskNewClients, &scheduler, true);
//Tarefa para verificar se há novas mensagens vindas de aplicativo
Task t2(100, TASK_FOREVER, &taskHandleClients, &scheduler, true);

//Id e estados deste esp (altere para cada esp)
String ID = "LIGHT1";
String ID_ON = ID + " ON";
String ID_OFF = ID + " OFF";

//Variável para guardar o valor do estado atual do relê
String currentState = ID_OFF;
```

Código setup

```
void setup(){
    //Coloca tudo em maiúsculo
    ID_ON.toUpperCase();
    ID_OFF.toUpperCase();

    //Coloca o pino onde o relê está como saída
    pinMode(RELAY, OUTPUT);

    Heltec.begin(true /*Ativa o display*/, true /*Ativa lora*/, true /*Ativa informações pela serial*/, true /*Ativa PABOOST*/, BAND /*frequência*/);
    //Inicializa o display

    setupDisplay();
    //Ativa o recebimento de pacotes
    LoRa.receive();

    //Se conecta à rede WiFi
    setupWiFi();
    //Inicializa o server ao qual vc vai se conectar utilizando o aplicativo
    server.begin(port);
    //Coloca o estado do relê para desligado
    verifyAndSetRelayState(ID_OFF);

    //Inicializa o agendador de tarefas
    scheduler.startNow();
}
```

Código *setupDisplay*

```
//Inicializa o display
void setupDisplay() {
    Heltec.display->init();

    //Limpa o display
    Heltec.display->clear();

    //Modifica direcionamento do texto
    Heltec.display->flipScreenVertically();

    //Alinha o texto à esquerda
    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);

    //Altera a fonte
    Heltec.display->setFont(ArialMT_Plain_16);

    //Exibe no display
    Heltec.display->drawString(0, 0, "Display OK");
    Heltec.display->display();
}
```

Código *setupWiFi*

```
void setupWiFi() {
    Serial.print("Conectando");

    //Faz o ESP se conectar à rede WiFi
    WiFi.begin(SSID, PASSWORD);

    //Enquanto o ESP não se conectar à rede
    while (WiFi.status() != WL_CONNECTED)
    {
        //Esperamos 100 milisegundos
        delay(100);
        Serial.print(".");
    }

    //Se chegou aqui é porque conectou à rede, então mostramos no monitor serial para termos um feedback
    Serial.println("");
    Serial.println("Conectou");

    //Objeto que vamos utilizar para guardar o ip recebido
    myIP = WiFi.localIP();
    //Mostra o ip no monitor serial
    Serial.println(myIP);

    //Atualiza o display para exibir o ip
    refreshDisplay();
}
```

Código *refreshDisplay*

```
void refreshDisplay() {  
    //Limpa o display  
    Heltec.display->clear();  
  
    //Exibe o estado atual do relê  
    Heltec.display->drawString(0, 0, currentState);  
  
    //Exibe o ip deste esp para ser utilizado no aplicativo  
    Heltec.display->drawString(0, 15, myIP.toString());  
    Heltec.display->display();  
}
```

Código *taskNewClients*

```
// Task que insere novos clientes conectados no vector
void taskNewClients(){
    // Se existir um novo client atribuimos para a variável
    WiFiClient 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()));
    }
}
```

Código *taskHandleClients*

```
// Função que verifica se o app enviou um comando
void taskHandleClients()
{
    // String que receberá o comando vindo do aplicativo
    String cmd;

    // Atualizamos o vector deixando somente os clientes conectados
    refreshConnections();

    // Percorremos o vector
    for(int i=0; i<clients.size(); i++){
        // Se existir dados a serem lidos
        if(clients[i].available()){
            // Recebemos a String até o '\n'
            cmd = clients[i].readStringUntil('\n');
            // Verificamos o comando, enviando por parametro a String cmd
            handleCommand(cmd);
        }
    }
}
```

Código *refreshConnections*

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

```
// Função que verifica o comando vindo do app
void handleCommand(String cmd)
{
    // Se a String estiver vazia não precisamos fazer nada
    if (cmd.equals(""))
        return;

    //Coloca todos os caracters em maiúsculo
    cmd.toUpperCase();
    // Exibimos o comando recebido no monitor serial
    Serial.println("Received from app: " + cmd);
    //Verifica se a mensagem é para este esp e modifica o estado do relê de acordo com o que foi enviado
    bool forMe = verifyAndSetRelayState(cmd);

    //Se a mensagem é para este esp
    if(forMe) {
        //Envia mensagem de confirmação de volta para os aplicativos conectados
        String confirmationMessage = currentState + " OK";
        sendToClients(confirmationMessage);
        Serial.println("Changed Relay status: " + confirmationMessage);
    }
    //Se não é para este esp
    else {
        //Envia o comando para os outros esp através de um pacote LoRa
        sendLoRaPacket(cmd);
    }
}
```

Código *verifyAndSetRelayState*

```
//Verifica se estado é valido para este esp e modifica o estado do relê de acordo
//Retorna true se a mensagem for para este esp e false caso contrário
bool verifyAndSetRelayState(String state) {
    //Se a mudança de estado pertence ao id vinculado a este esp
    if (state == ID_ON || state == ID_OFF) {
        //Guarda o estado atual
        currentState = state;

        //Modificamos o estado do relê de acordo com o estado enviado
        digitalWrite(RELAY, currentState == ID_ON ? LOW : HIGH);

        //Atualizamos o display com o estado atualizado
        refreshDisplay();
        return true;
    }
    return false;
}
```

Código *sendToClients* e *sendLoRaPacket*

```
//Função que envia mensagem para todos os apps conectados
void sendToClients(String msg) {
    for(int i=0; i<clients.size(); i++){
        clients[i].print(msg);
    }
}

//Envia um pacote LoRa
void sendLoRaPacket(String str) {
    //Inicializa o pacote
    LoRa.beginPacket();
    LoRa.setTxPower(14,RF_PA_CONFIG_PASELECT_PABOOST);
    //Coloca a string no pacote
    LoRa.print(str);
    //Finaliza e envia o pacote
    LoRa.endPacket();
}
```

Código *loop*

```
void loop() {
    //Faz a leitura do pacote
    String packet = readLoRaPacket();

    //Se uma mensagem chegou
    if(!packet.equals("")) {
        //As mensagens enviadas dos outros ESPs são apenas de feedback
        //Então enviamos a mensagem como confirmação para os aplicativos conectados
        sendToClients(packet);
    }

    //Executa as tarefas que foram adicionadas ao scheduler
    scheduler.execute();
}
```

Código *readLoRaPacket*

```
//Faz a leitura de um pacote (se chegou algum)
String readLoRaPacket() {
    String packet = "";

    //Verifica o tamanho do pacote
    int packetSize = LoRa.parsePacket();

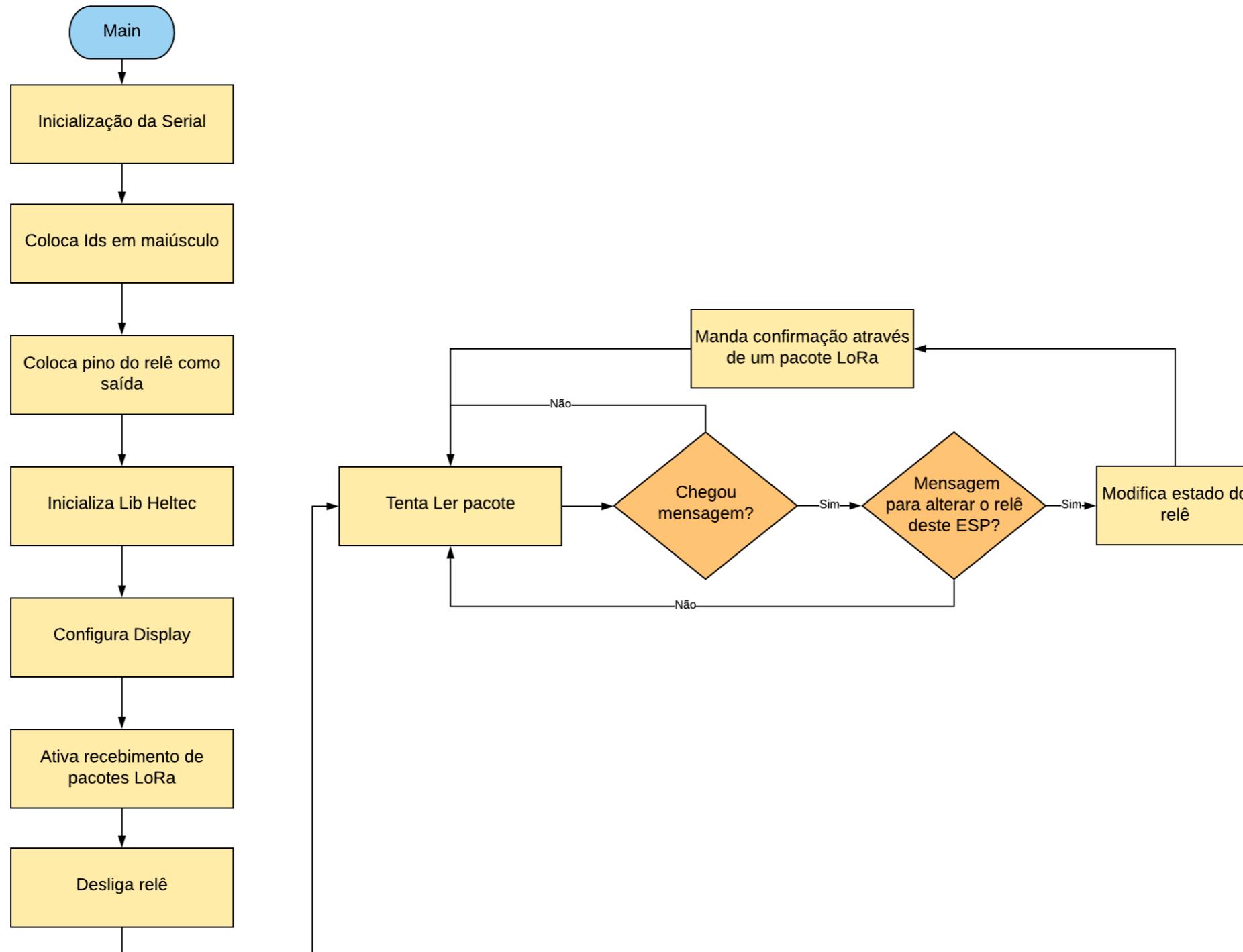
    //Lê cada caractere e concatena na string
    for (int i = 0; i < packetSize; i++) {
        packet += (char) LoRa.read();
    }

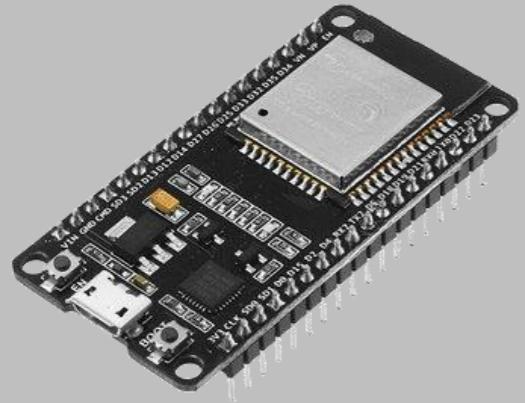
    return packet;
}
```



Código Receivers

Fluxograma





Código Receivers

Código Imports e Variáveis

```
#include "heltec.h"

//Frequência
#define BAND 433E6

//Pino onde o relê está
#define RELAY 13

//Id e estados deste esp (altere para cada esp)
String ID = "LIGHT2";
String ID_ON = ID + " ON";
String ID_OFF = ID + " OFF";

//Variável para guardar o valor do estado atual do relê
String currentState = ID_OFF;
```

Código *setup*

```
void setup() {  
    //Coloca tudo em maiúsculo  
    ID_ON.toUpperCase();  
    ID_OFF.toUpperCase();  
  
    //Coloca o pino onde o relê está como saída  
    pinMode(RELAY, OUTPUT);  
  
    Heltec.begin(true /*Ativa display*/, true /*Ativa LoRa*/, true /*Ativa informações pela  
serial*/, true /*Ativa PABOOST*/, BAND /*frequência*/);  
  
    //Inicializa o display  
    setupDisplay();  
  
    //Ativa o recebimento de pacotes  
    LoRa.receive();  
  
    //Coloca o estado do relê como desligado  
    verifyAndSetRelayState(ID_OFF);  
}
```

Código *setupDisplay*

```
//Inicializa o display
void setupDisplay() {
    Heltec.display->init();

    //Limpa o display
    Heltec.display->clear();

    //Modifica direcionamento do texto
    Heltec.display->flipScreenVertically();

    //Alinha o texto à esquerda
    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);

    //Altera a fonte
    Heltec.display->setFont(ArialMT_Plain_16);

    //Exibe no display
    Heltec.display->drawString(0, 0, "Display OK");
    Heltec.display->display();
}
```

Código *verifyAndSetRelayState*

```
//Verifica se estado é valido para este esp e modifica o estado do relê de acordo
//Retorna true se a mensagem for para este esp e false caso contrário
bool verifyAndSetRelayState(String state) {
    //Se a mudança de estado pertence ao id vinculado a este esp
    if (state == ID_ON || state == ID_OFF) {
        //Guarda o estado atual
        currentState = state;

        //Modificamos o estado do relê de acordo com o estado enviado
        digitalWrite(RELAY, currentState == ID_ON ? LOW : HIGH);

        //Atualizamos o display com o estado atualizado
        refreshDisplay();
        return true;
    }
    return false;
}
```

Código *refreshDisplay*

```
//Atualiza dados que aparecem no display
void refreshDisplay() {
    //Limpa o display
    Heltec.display->clear();

    //Mostra o estado atual do relê
    Heltec.display->drawString(0 , 0 , currentState);

    //Mostra os dados
    Heltec.display->display();
}
```

Código *loop*

```
void loop() {
    //Faz a leitura do pacote
    String packet = readLoRaPacket();

    //Se uma mensagem chegou
    if(!packet.equals("")) {
        //Verifica se a mensagem é para este esp e modifica o estado do relê
        //de acordo com o que foi enviado
        bool forMe = verifyAndSetRelayState(packet);

        //Se a mensagem é para este esp
        if(forMe) {
            //Envia de volta um pacote de confirmação
            sendLoRaPacket(currentState + " OK");
        }
    }
}
```

Código *readLoRaPacket*

```
//Faz a leitura de um pacote (se chegou algum)
String readLoRaPacket() {
    String packet = "";

    //Verifica o tamanho do pacote
    int packetSize = LoRa.parsePacket();

    //Lê cada caractere e concatena na string
    for (int i = 0; i < packetSize; i++) {
        packet += (char) LoRa.read();
    }

    return packet;
}
```

Código *sendLoRaPacket*

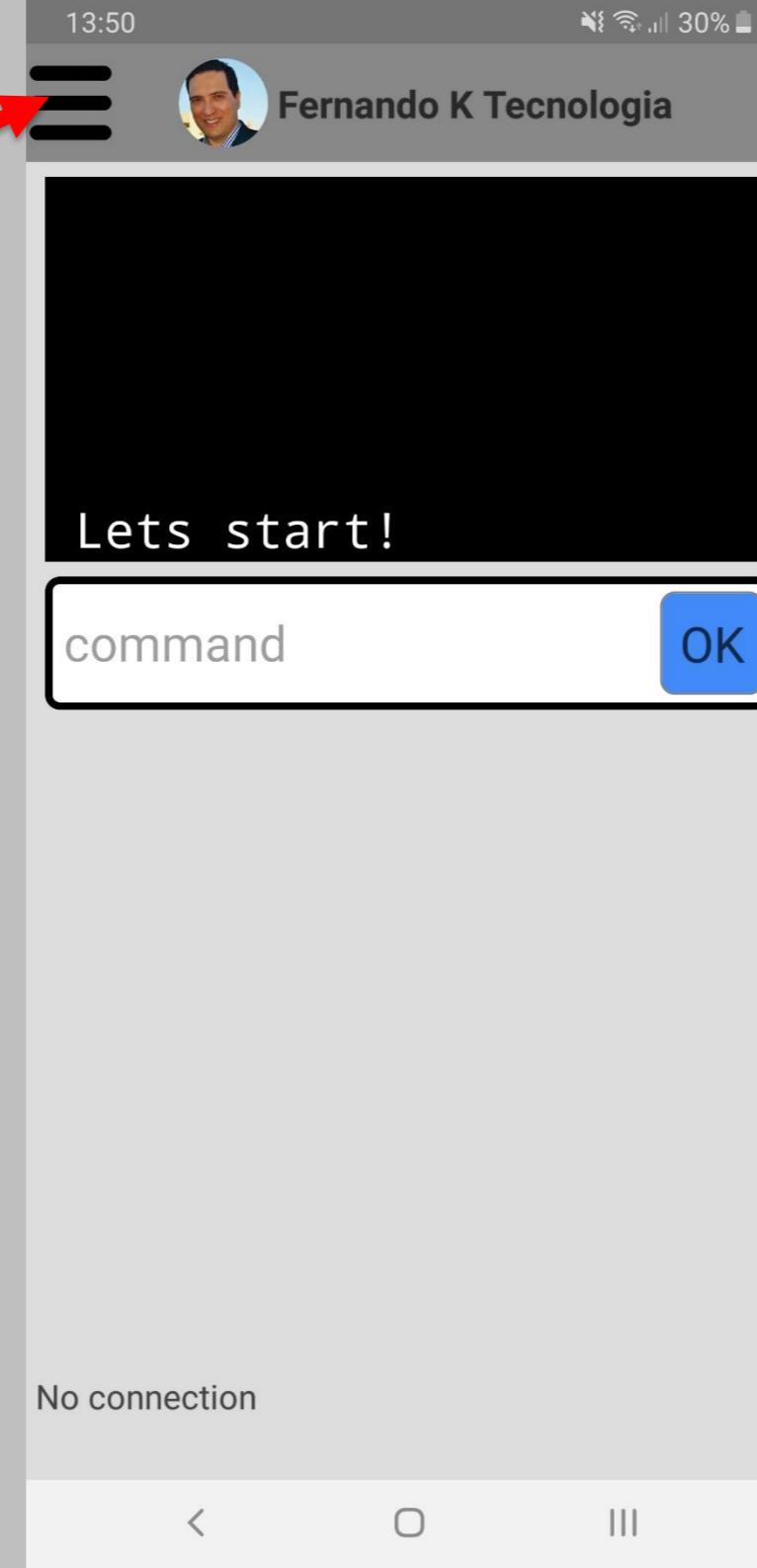
```
//Envia um pacote LoRa
void sendLoRaPacket(String str) {
    //Inicializa o pacote
    LoRa.beginPacket();

    //Coloca a string no pacote
    LoRa.print(str);

    //Finaliza e envia o pacote
    LoRa.endPacket();
}
```

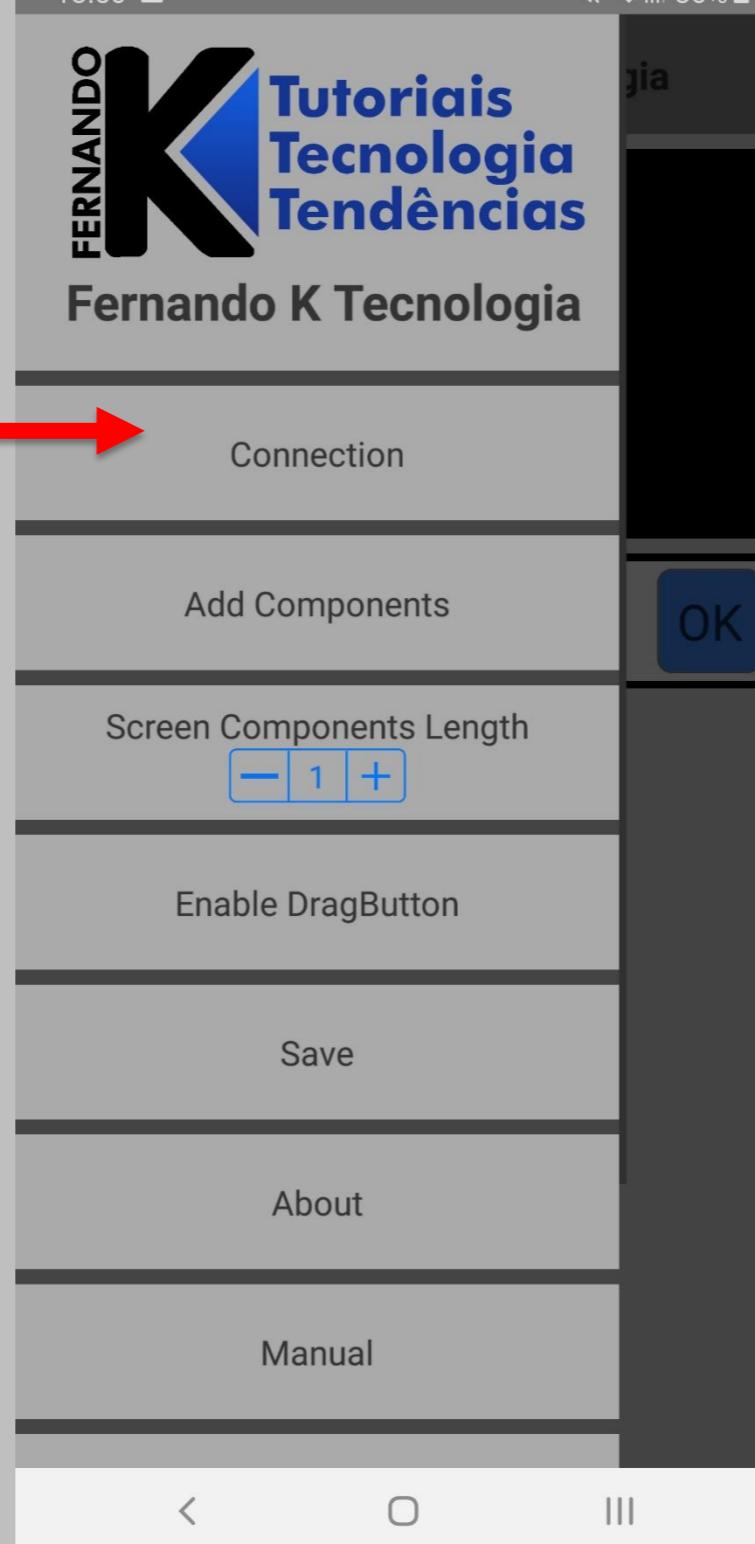
App Fernando K

Clique aqui

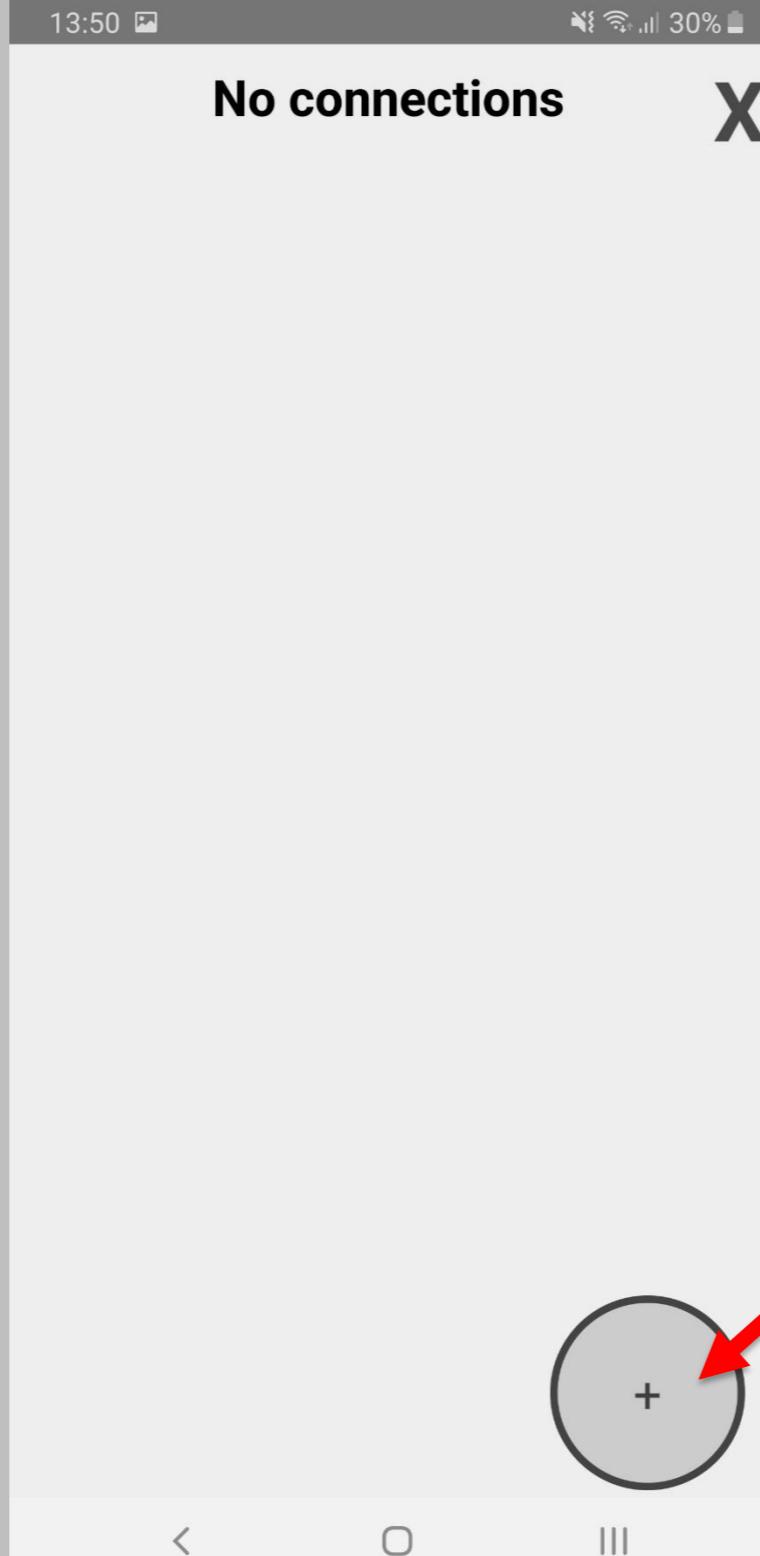


AppFernandoK

Clique em
Connection

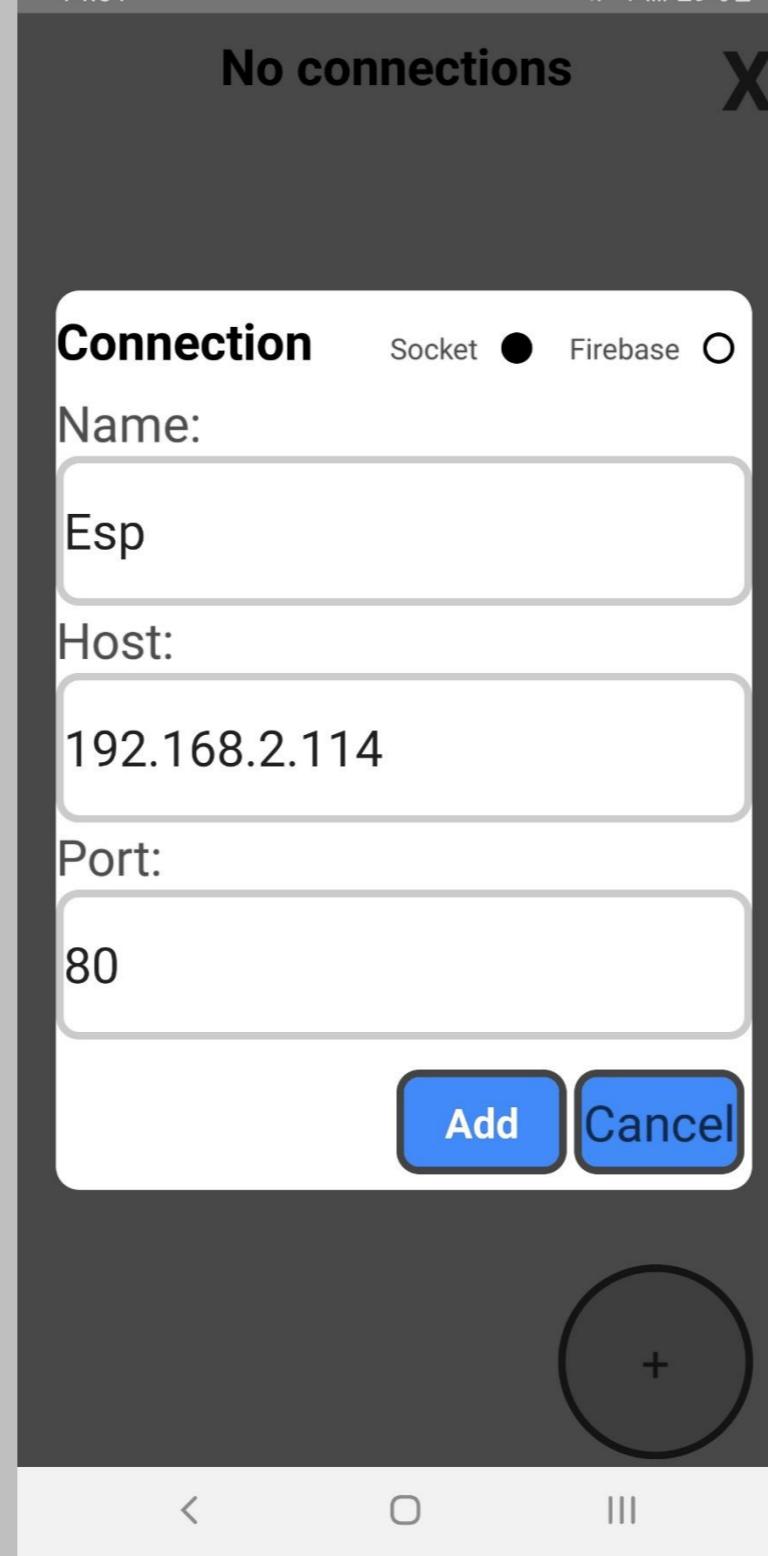


AppFernandoK



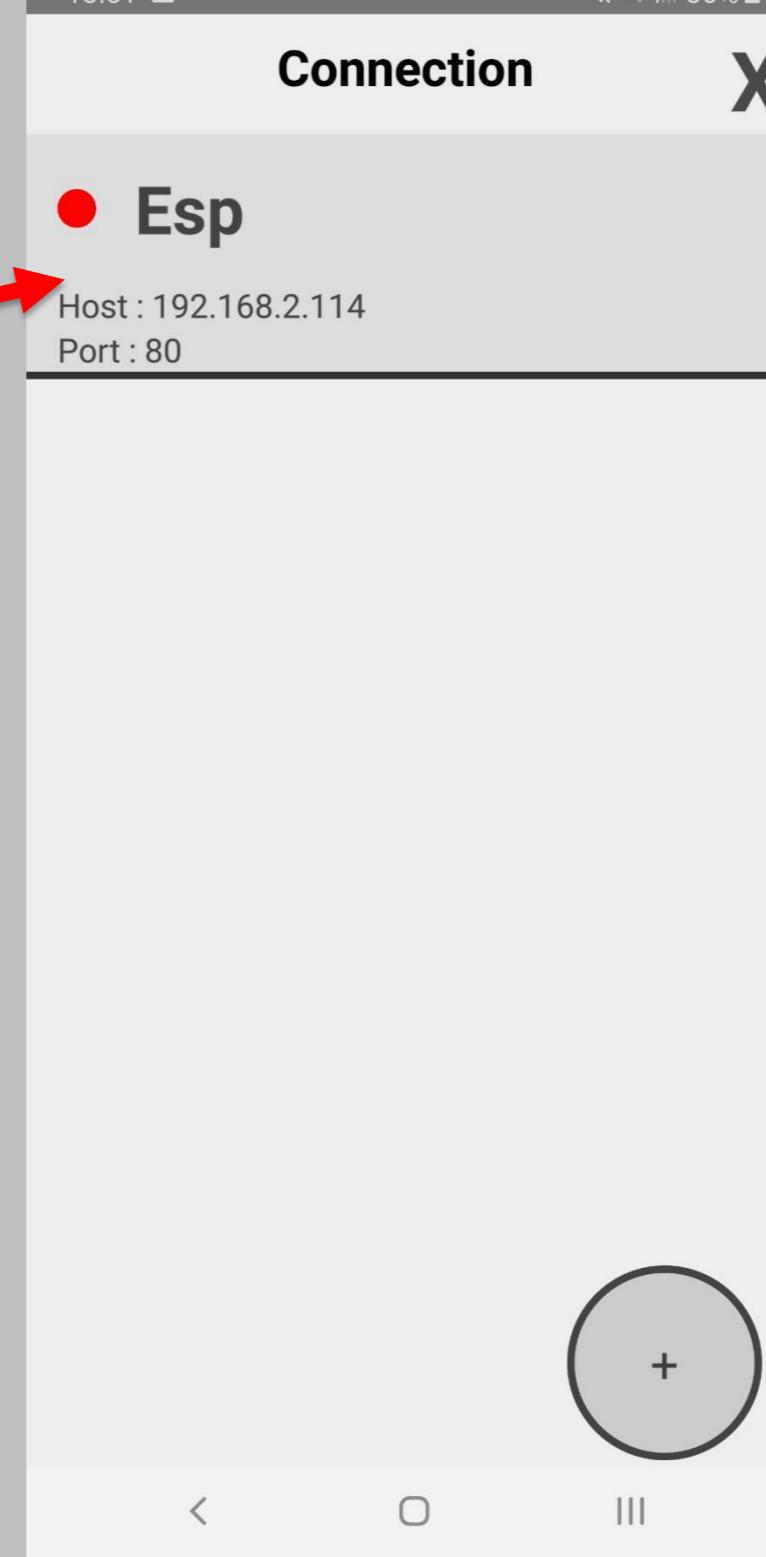
AppFernandoK

**Adicione nome
para a conexão,
IP, porta e clique
em Add**



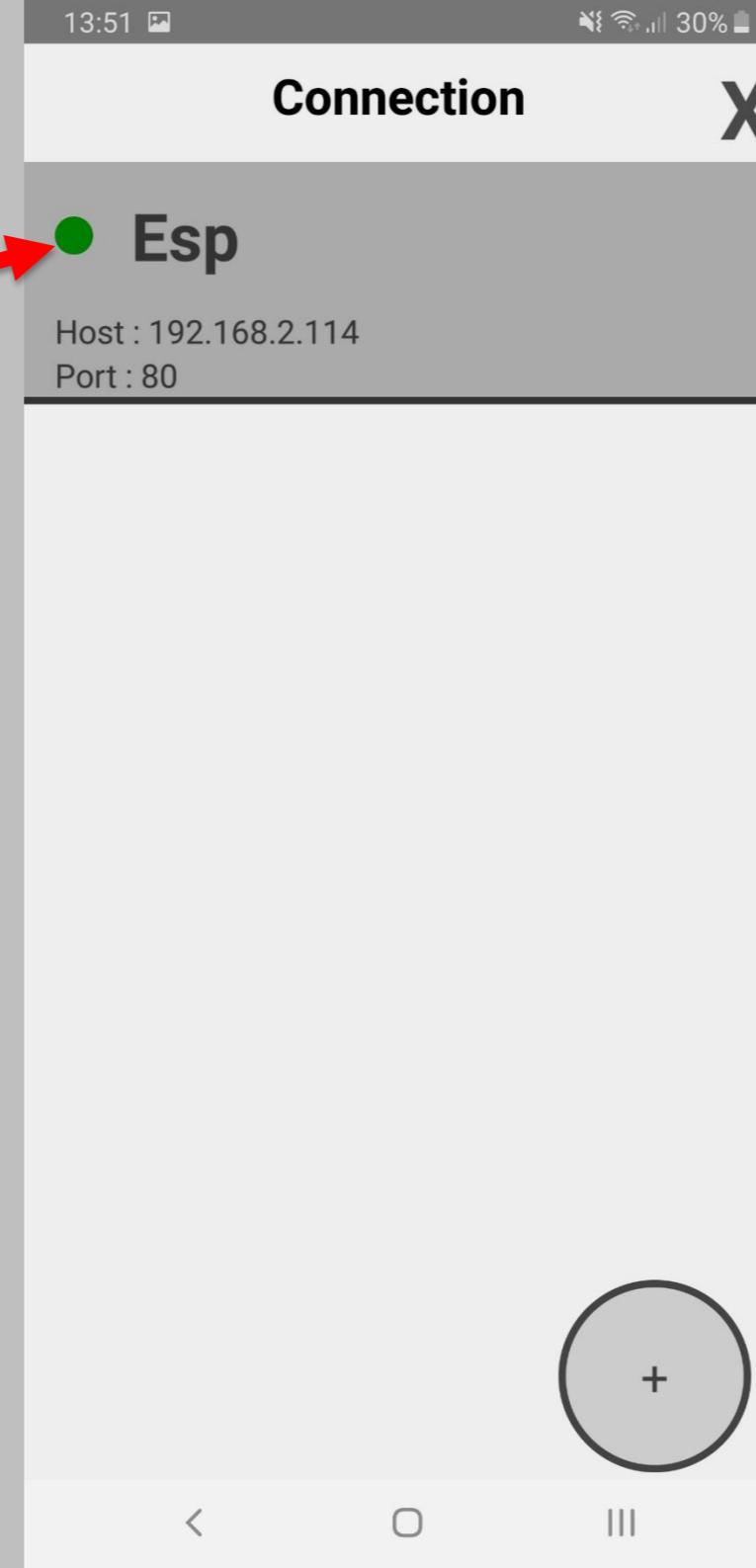
AppFernandoK

Clique na conexão
que acabou de criar



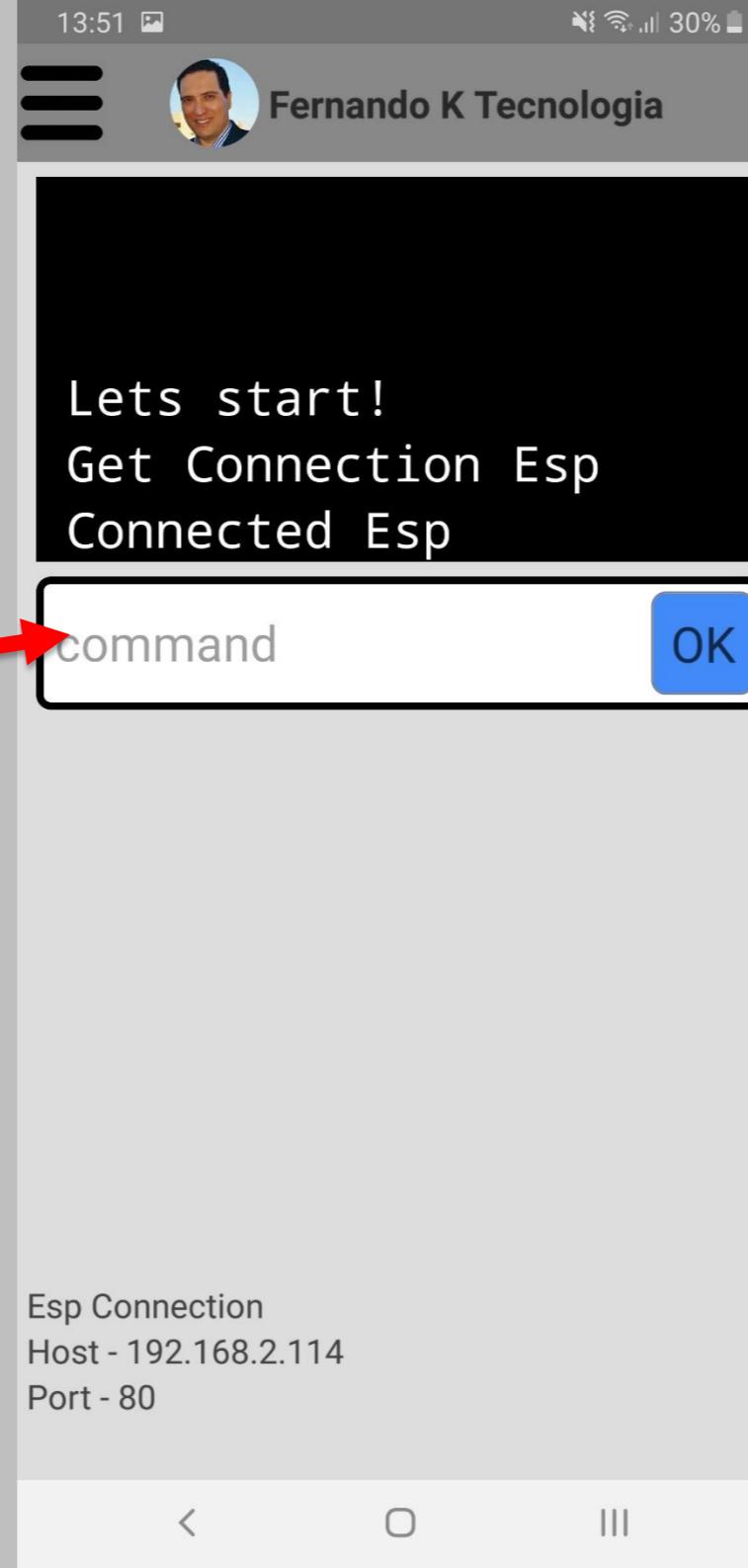
AppFernandoK

O círculo deve estar
verde se a conexão
foi bem sucedida



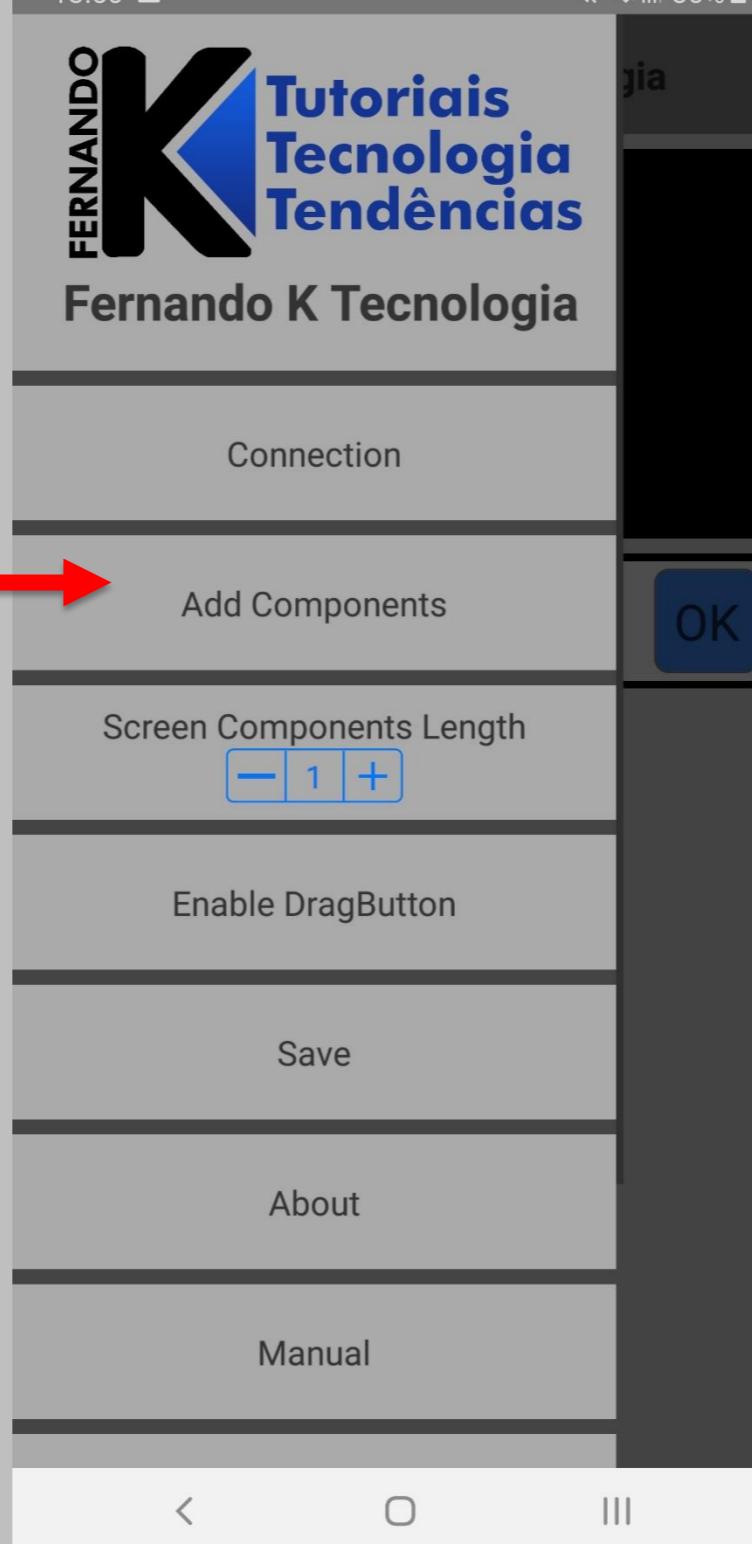
AppFernandoK

Você pode enviar
os comandos
digitando aqui



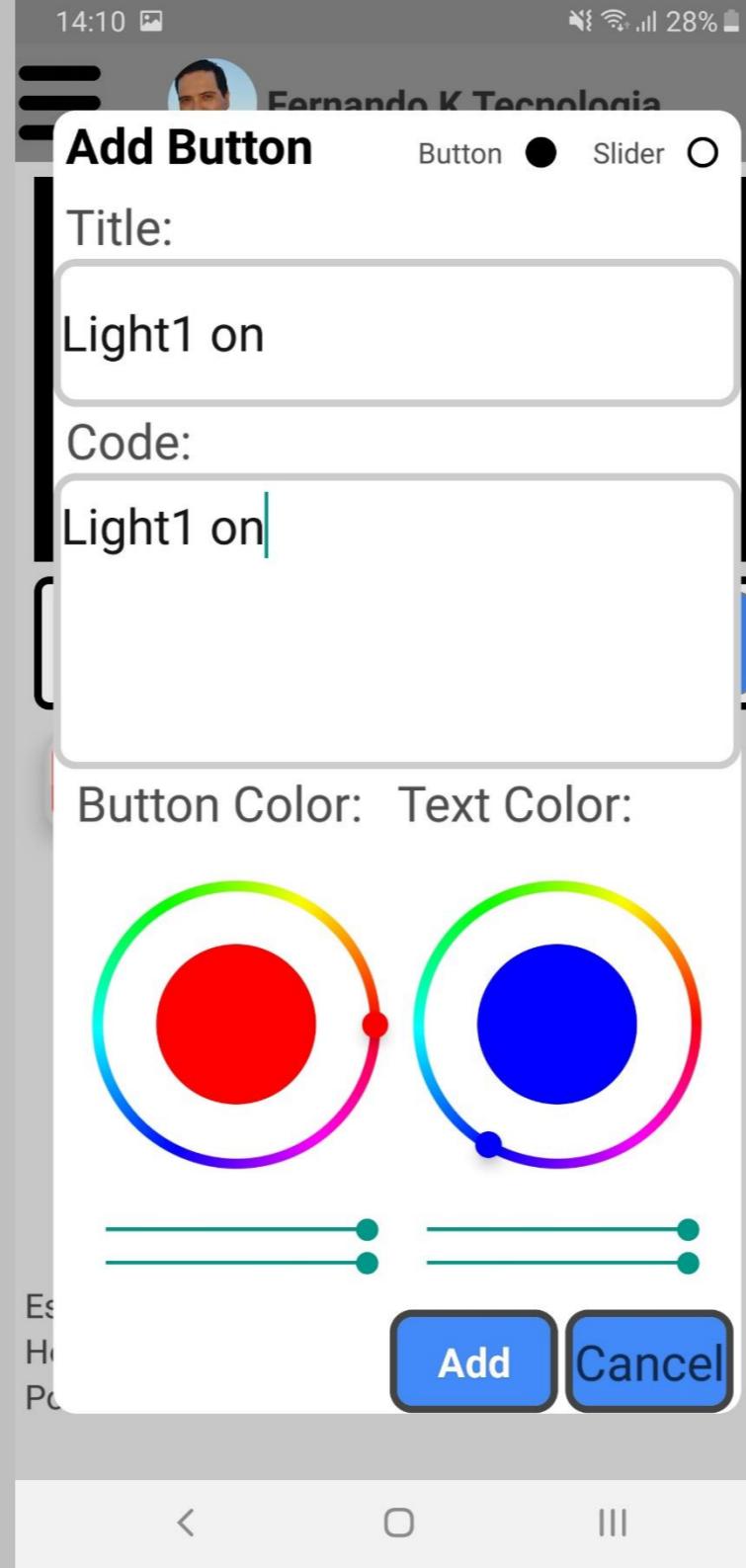
AppFernandoK

Ou pode
criar botões
clicando
aqui



AppFernandoK

Digite o título,
comando do botão
e clique em Add



AppFernandoK

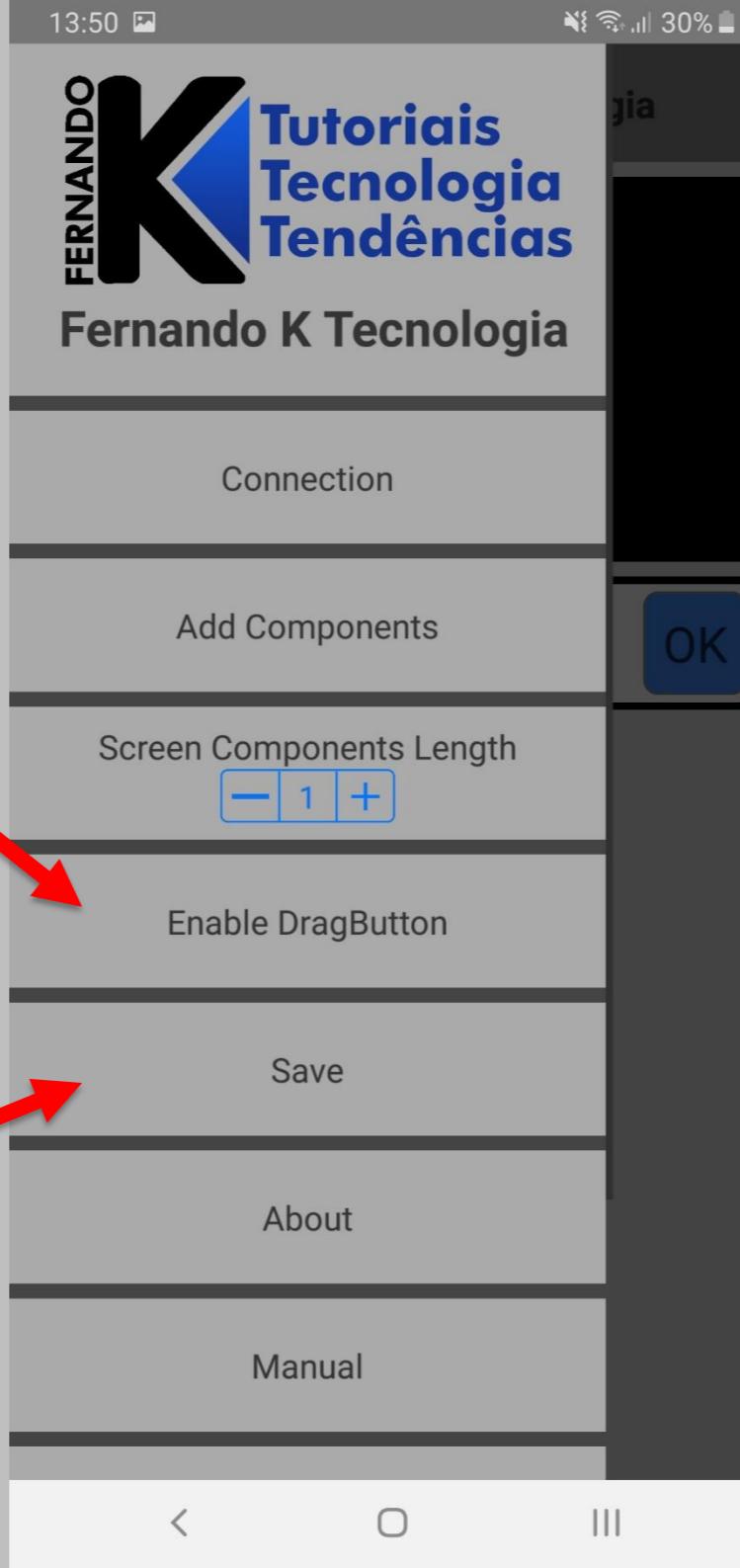
**Clique no botão
e verá uma
mensagem de
confirmação no
terminal
se tudo deu certo**



AppFernandoK

Você pode
mover os
botões
clicando
aqui

Lembre-se
de salvar



Em www.fernandok.com

Download arquivos PDF e INO do código fonte

