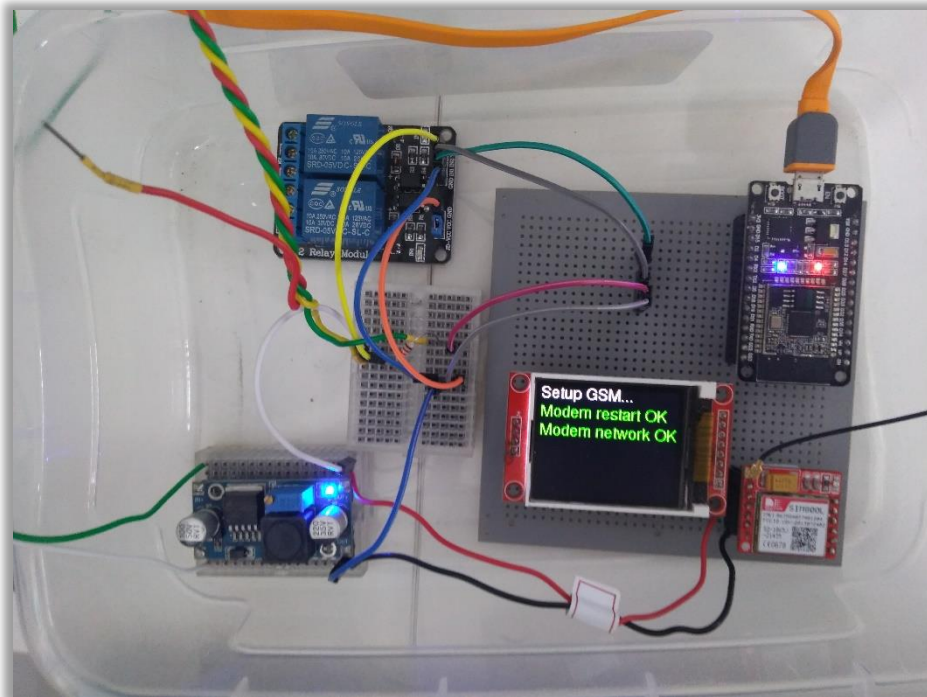
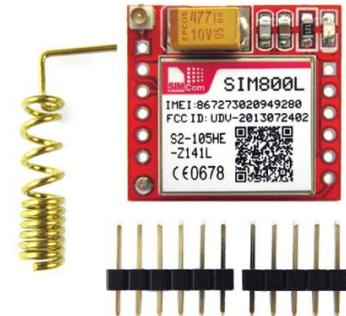
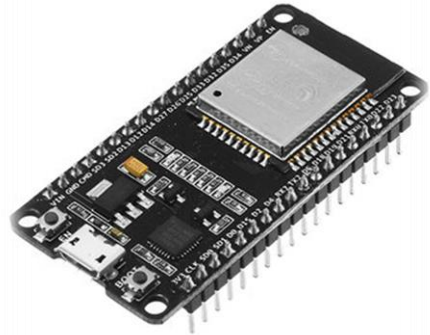


# ESP32 - SIM800L

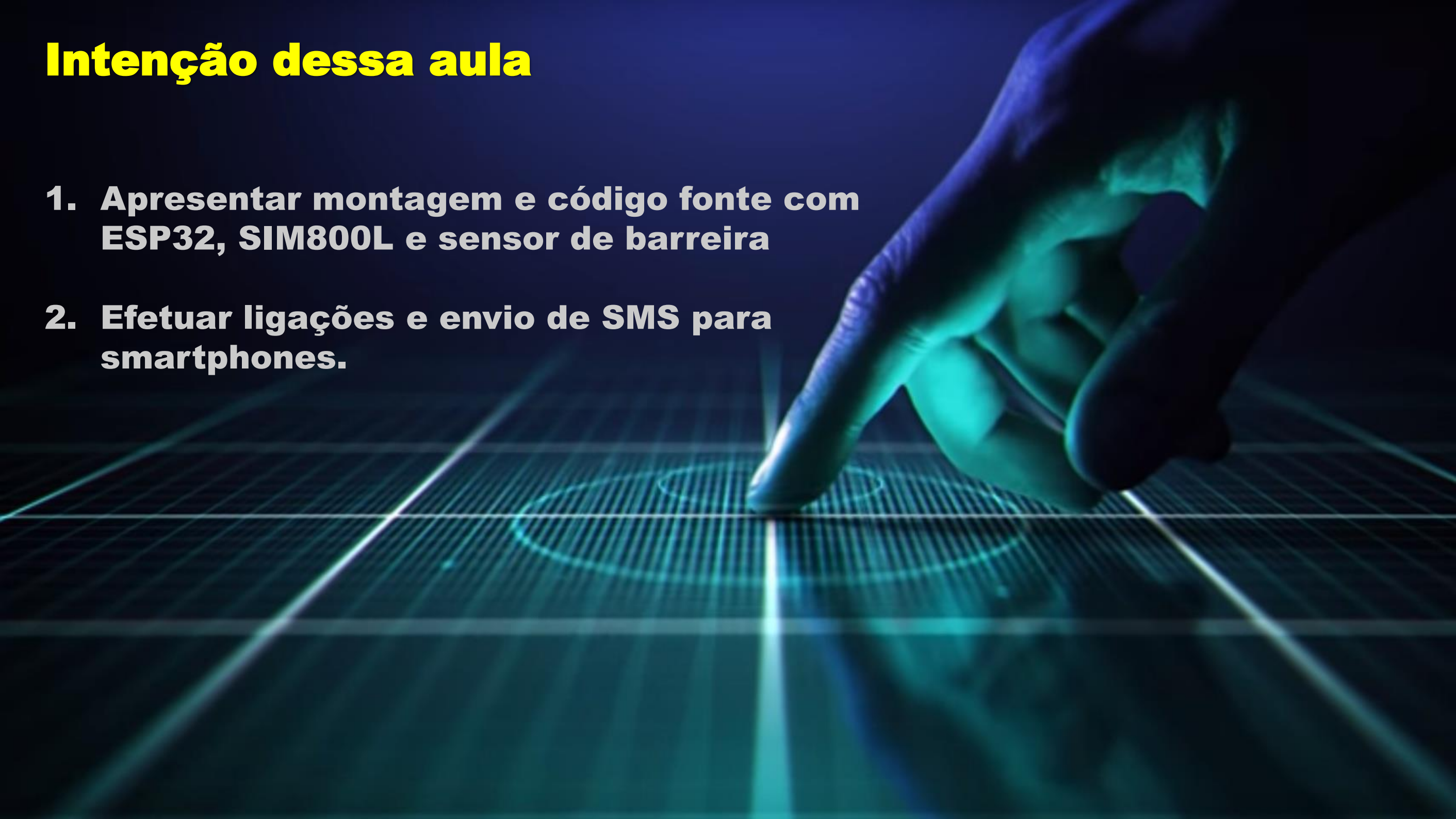
## Automação com Sensor de Barreira



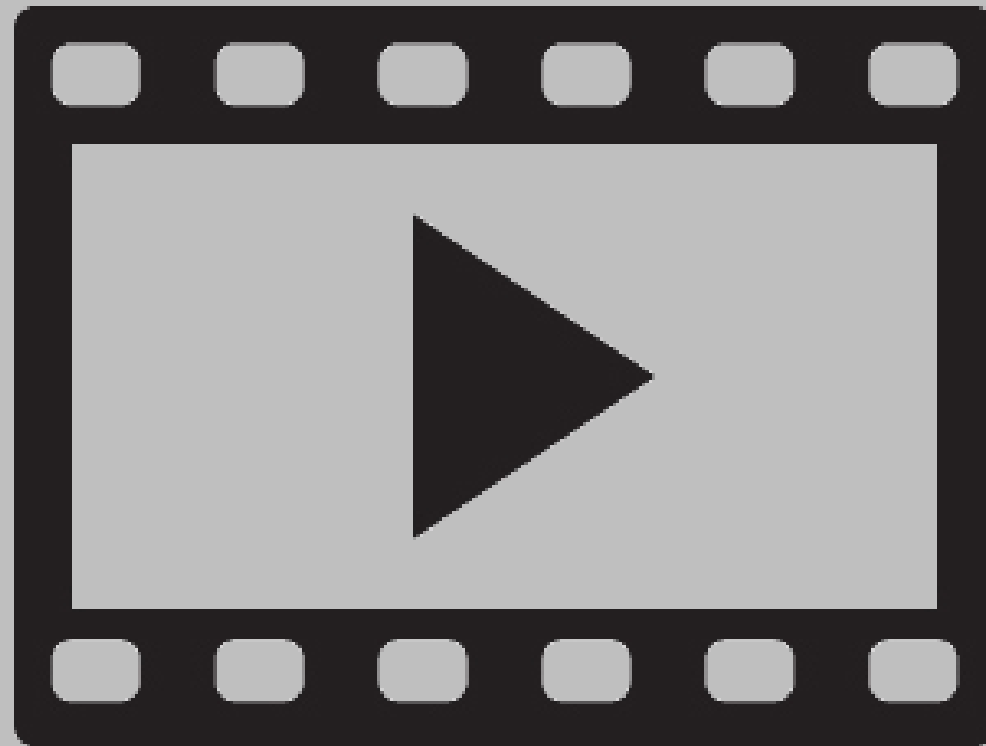
Por Fernando Koyanagi

# **Intenção dessa aula**

- 1. Apresentar montagem e código fonte com ESP32, SIM800L e sensor de barreira**
- 2. Efetuar ligações e envio de SMS para smartphones.**



# Demonstração







Em [www.fernandok.com](http://www.fernandok.com)

Seu e-mail



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



## 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

- ☐ 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

[www.fernandok.com](http://www.fernandok.com) [/fernandokoyanagi](#) [/fernandokoyanagi](#) [/fernandok\\_oficial](#) [/fernandok\\_oficial](#)



Links rápidos







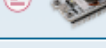

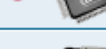



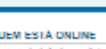

fernandokoyanagi

Bem-vindo: 05/Out/2018, 11:16

A sua última visita foi em 10/Set/2018, 15:47

Assinalar todos os fóruns como lidos

SUPPORT: FÓRUM FERNANDOK	TÓPICOS	MENSAGENS	ÚLTIMA MENSAGEM
 <b>Feedback</b> Dúvidas, críticas ou sugestões sobre o Fórum FernandoK. Para demais questões utilize o fórum correto.	6	11	<b>Re: O russo voltou</b> por Iperito  01/Out/2018, 08:25

FERNANDO K	TÓPICOS	MENSAGENS	ÚLTIMA MENSAGEM
 <b>Arduino</b> Projetos de arduino	31	79	<b>skard y txgji</b> por Soresorcem  05/Out/2018, 10:55
 <b>ESP32</b> Projetos de ESP32	29	62	<b>Duvidas sobre como instalar a...</b> por Marcelo Jorge  04/Out/2018, 15:52
 <b>ESP8266</b> O ESP8266 é um microcontrolador do fabricante chinês Espressif que inclui capacidade de comunicação por Wi-Fi.	24	51	<b>Re: NodeMCU não conecta em qu...</b> por ivansilva  04/Out/2018, 14:39
 <b>LoRa</b> Projetos com LoRa	11	31	<b>Projeto de irrigação de jardim</b> por marlonc  04/Out/2018, 21:30
 <b>STM32</b> Projetos com STM32	3	8	<b>Re: Imprecisão de tempo de de...</b> por biaroto  12/Set/2018, 09:15
 <b>Motor</b> Projetos com motor	5	11	<b>Re: impressora 3d com motor dc</b> por Magetron  24/Set/2018, 19:06
 <b>Display</b> Projetos com Display	4	11	<b>Re: Alguem conhece o VIRTUINO...</b> por Joel Luz  21/Set/2018, 11:39

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

Í...

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



# Instagram

fernandok\_oficial

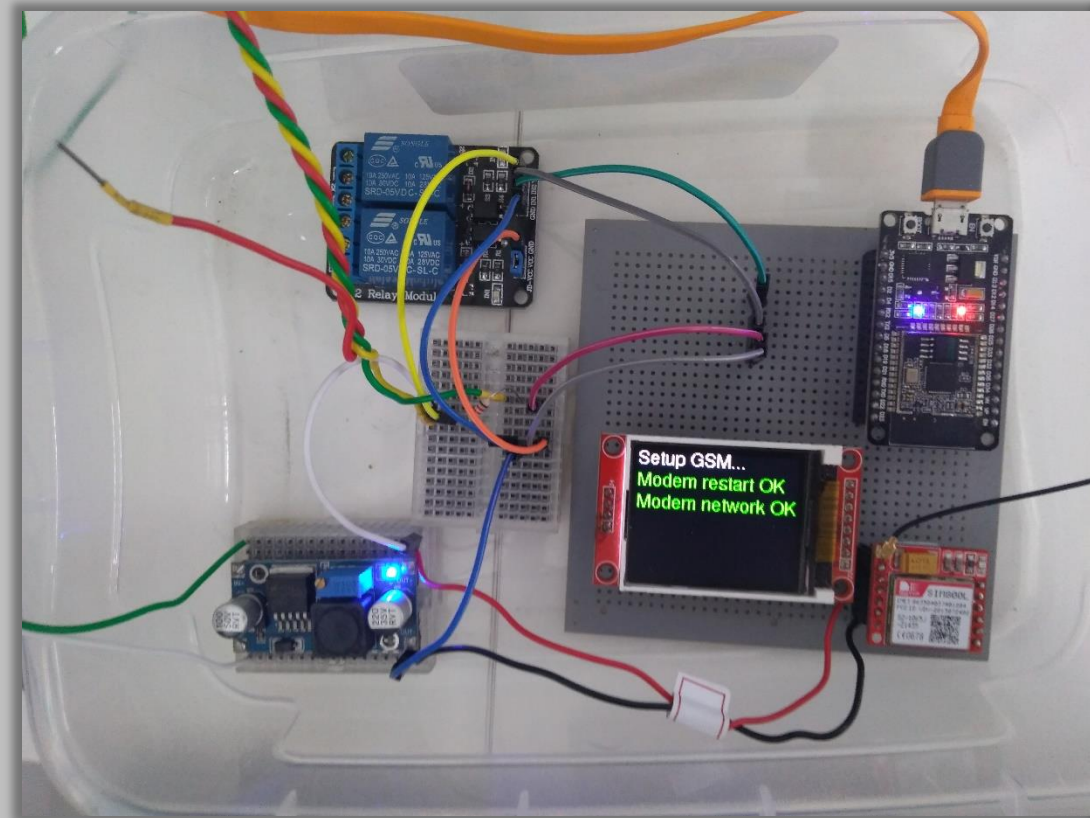


# Telegram

fernandok\_oficial



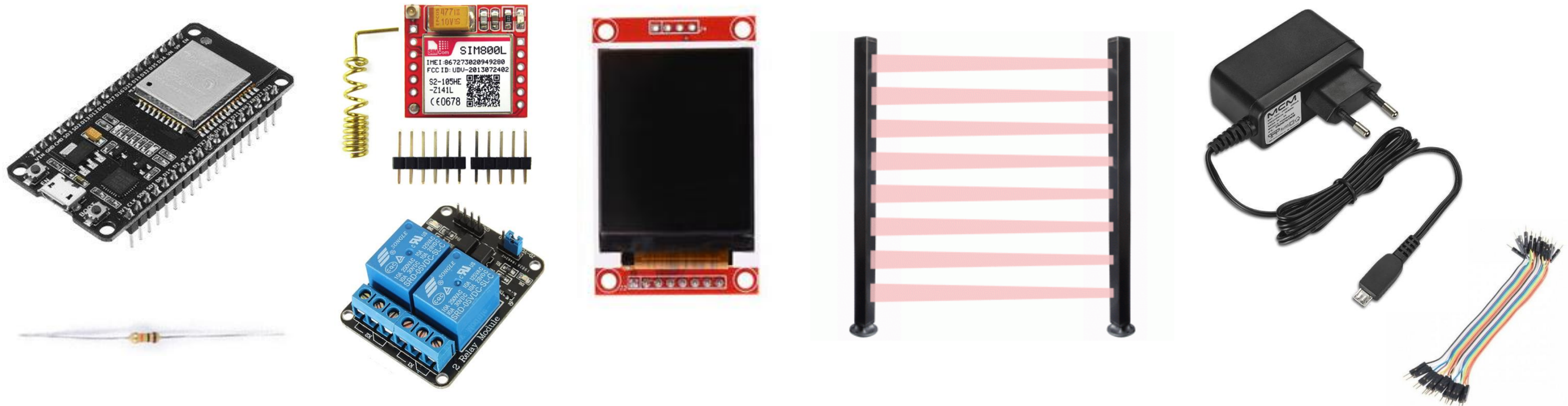
# Montagem





# Recursos usados

- ESP32 – WROOM
- SIM800L
- Display TFT 1.8"
- Sensor de barreira
- Módulo de 2 Reles
- Resistor 10k ohm
- Fonte 4.1V e 5V
- Jumpers
- 1x Cartão SIM com plano SMS para o Smartphone
- 1x Cartão SIM com plano SMS e ligação para o SIM800L
- Smartphone





# Pinout ESP32

## ESP32 PINOUT

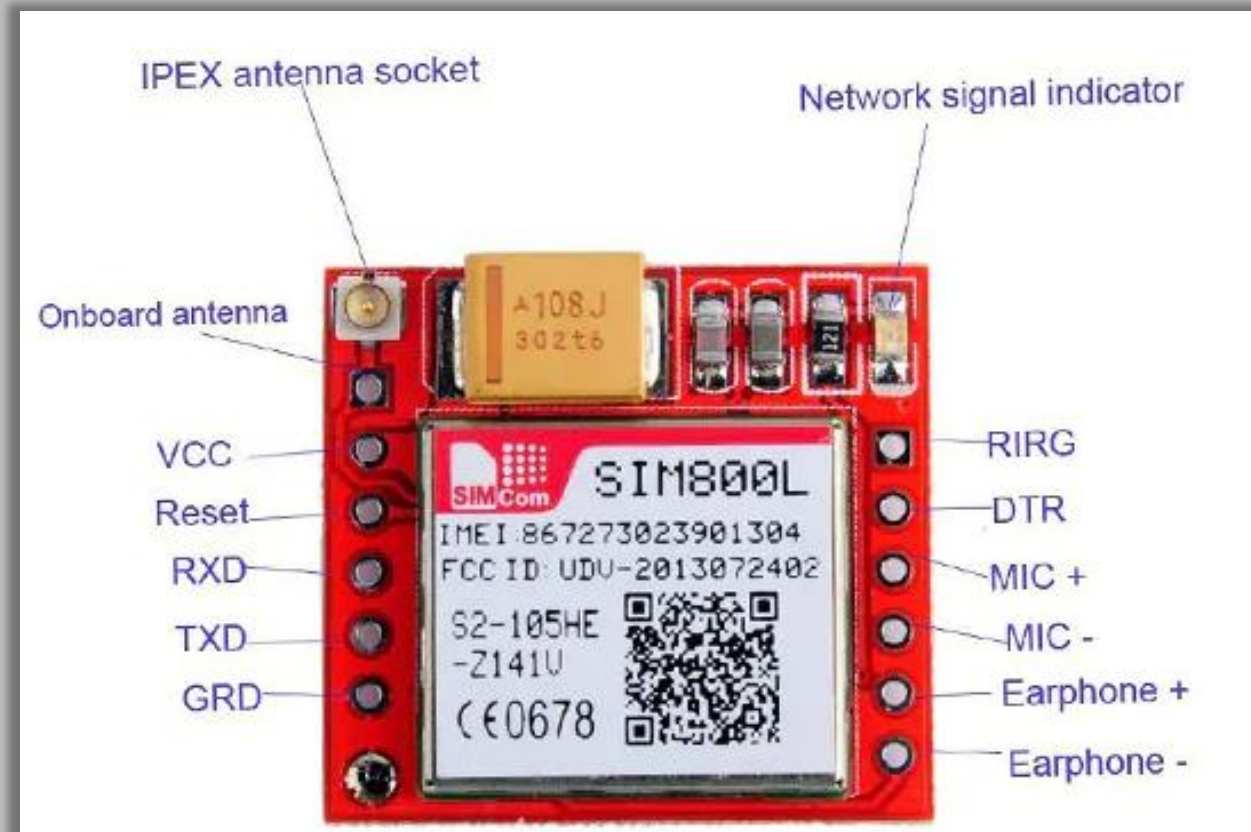
### ESP32 DEVELOPMENT BOARD DUAL CORE ESP-32 & ESP-32S BOARD



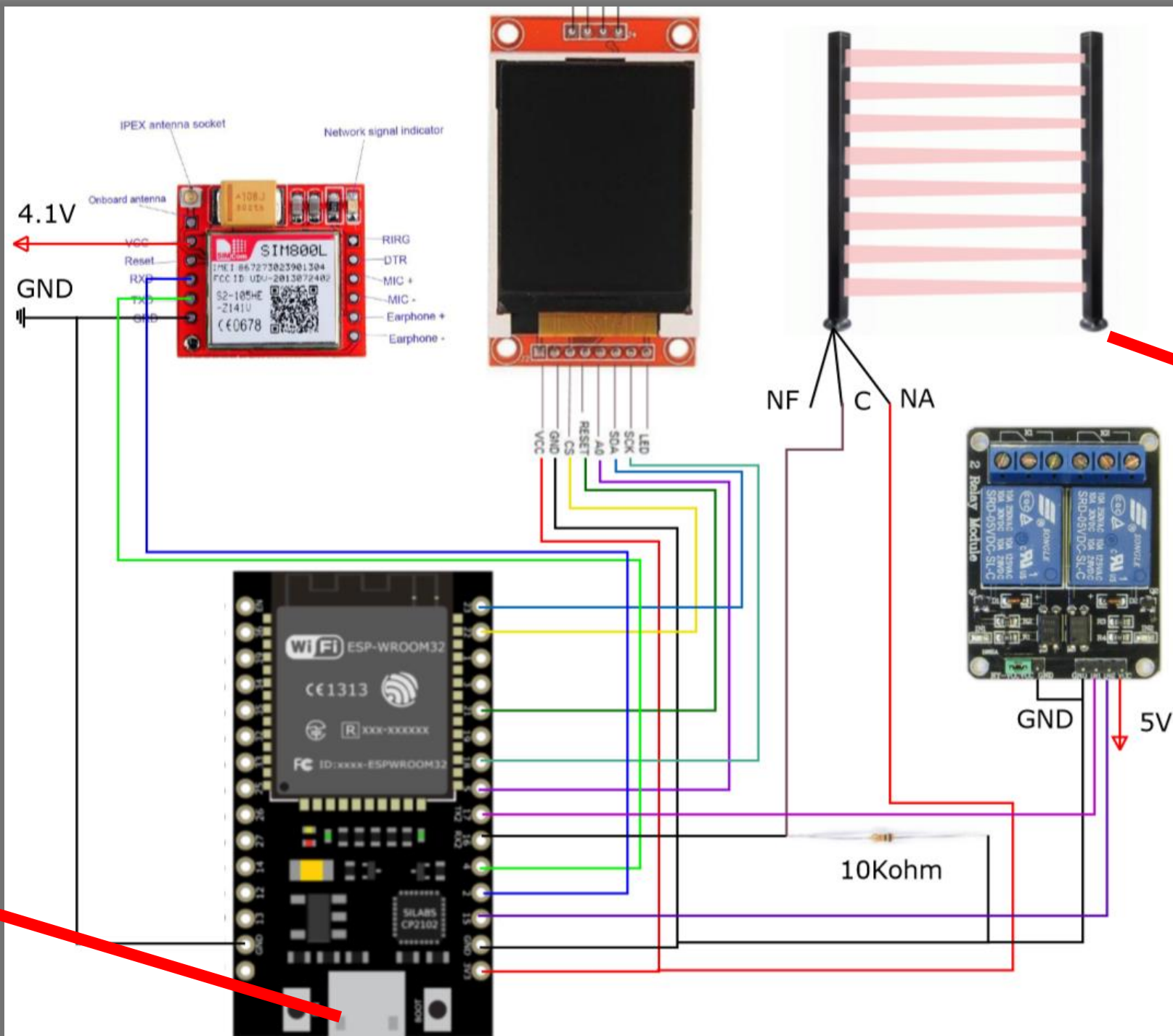
Freely adapted by <https://MJRoBot.org>



# Pinout SIM800L



# Montagem



Alimentação 12V

\*Deixe os GND em comum



# Montagem - Tabela

ESP32	SIM800L
GPIO 4(RX)	TX
GPIO 2(TX)	RX
GND	GND

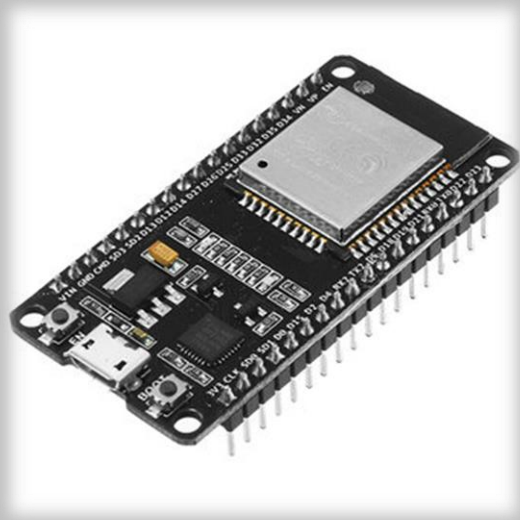
ESP32	Módulo reles
GPIO 17	IN1
GPIO 15	IN2

ESP32	Display
GPIO 22	CS
GPIO 21	RST
GPIO 5	DC
GPIO 23	MOSI
GPIO 18	CLK
3V3	VCC
GND	GND

ESP32	Sensor de barreira
3V3	NA
GPIO 16	C

\*ligar um resistor de **10kohm** entre GPIO16 e GND do ESP32 conforme slide anterior

Alimentação	
Fonte 4V1	SIM800L (VCC e GND)
Fonte 5V	Módulo reles (VCC e GND)
Fonte 12V	Sensor de barreira



# Código

# **Instalação de bibliotecas**

<https://github.com/adafruit/Adafruit-GFX-Library>

<https://github.com/adafruit/Adafruit-ST7735-Library>

<https://github.com/vshymanskyy/TinyGSM/blob/master/src/TinyGsmClient.h>



# Código **ESP32** Declarações e variáveis

```
#include <Arduino.h> //biblioteca arduino (opcional)
#include <Adafruit_GFX.h> //biblioteca do display gráfico
#include <Fonts/FreeSans9pt7b.h> //fonte usada no display
#include <Adafruit_ST7735.h> // biblioteca de hardware do display
#include <SPI.h> // biblioteca de comunicação SPI

#define TINY_GSM_MODEM_SIM800 // definição do modem usado (SIM800L)
#include <TinyGsmClient.h> // biblioteca com comandos GSM

// pinos display
#define TFT_CS 22 // CS
#define TFT_RST 21 // RESET
#define TFT_DC 5 // A0
#define TFT_MOSI 23 // SDA
#define TFT_CLK 18 // SCK
// objeto do display
Adafruit_ST7735 display = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_CLK, TFT_RST);
// tamanho da fonte do display
int fontHeight = 12;
// objeto de comunicação serial do SIM800L
HardwareSerial SerialGSM(1);
// objeto da biblioteca com as funções GSM
TinyGsm modemGSM(SerialGSM);
// velocidade da serial tanto do SIM800L quanto do monitor serial
const int BAUD_RATE = 9600;
```

# Código **ESP32** Declarações e variáveis (continuação)

```
// variáveis usadas para contar o tempo sem travar a função loop
// millis de referencia
long int millisRefCon, millisUserResp;
// flag que indica a contagem de tempo (usadas pela função 'timeout')
bool flagCon = false, flagUserResp = false;
// pinos aonde os reles serão ligados e RX / TX aonde o SIM800L será ligado
const int relayPin1 = 17, relayPin2 = 15, sensorPin = 16, RX_PIN = 4, TX_PIN = 2;
// Access point name da vivo
const char *APN = "zap.vivo.com.br";
// Usuario, se não existir deixe em vazio
const char *USER = "";
// Password, se não existir deixe em vazio
const char *PASSWORD = "";
// as variáveis abaixo usadas pela função loop
// flag que indica se, após a ligação feita pelo SIM800L, um usuario respondeu com um SMS em até 1min
bool userResponseSMS = false;
// flag que indica se o sensor está ativo
bool sensorActivated = false;
// index do vetor de numeros de celular, usado para percorrer o vetor
int i = 0;
// quantidade de celulares que receberão mensagens e ligações e poderão enviar comandos SMS
const int numbersTL = 2;
// número de celulares, a ordem de chamada pelo programa é da esquerda para a direita
const String numbers[numbersTL] = {"+5518999999999", "+5518999999999"};
```

# Código **ESP32** Setup

Setup

```
void setup()
{
  Serial.begin(BAUD_RATE);
  Serial.println("Starting...");
  // seta pinos do sensor como entrada
  pinMode(sensorPin, INPUT);
  // seta pinos dos reles como saída
  pinMode(relayPin1, OUTPUT);
  pinMode(relayPin2, OUTPUT);
  // os reles trabalham com lógica inversa, setamos como HIGH para desligá-los de início
  digitalWrite(relayPin1, HIGH);
  digitalWrite(relayPin2, HIGH);

  // atribui para as variáveis de contagem de tempo o tempo atual antes de entrar no loop
  millisRefCon = millisUserResp = millis();

  display.initR(INITR_BLACKTAB);
  resetDisplay();

  // inicia e configura o SIM800L
  setupGSM();
  resetDisplay();
  display.println("GPRS: Connected");
}
```



# Código **ESP32** *setupGSM*

Funções  
do Setup

```
// inicializa GSM
void setupGSM()
{
  display.println("Setup GSM...");
  display.setTextColor(ST7735_GREEN);
  // inicia serial SIM800L
  SerialGSM.begin(BAUD_RATE, SERIAL_8N1, RX_PIN, TX_PIN, false);
  delay(3000);

  // exibe info do modem no monitor serial
  Serial.println(modemGSM.getModemInfo());

  // inicia o modem
  if (!modemGSM.restart())
  {
    display.setTextColor(ST7735_RED);
    display.println("Restarting GSM\nModem failed");
    delay(10000);
    ESP.restart();
    return;
  }

  display.println("Modem restart OK");
}
```

```
display.println("Modem network OK");
if(!modemGSM.gprsConnect(APN,USER,PASSWORD)) // conecta na rede (tecnologia GPRS)
{
    display.setTextColor(ST7735_RED);
    display.println("GPRS Connection\nFailed");
    delay(10000);
    ESP.restart();
    return;
}
display.println("GPRS Connect OK");
if(sendAT("AT+CMGF=1").indexOf("OK") < 0) //Define modo SMS para texto (0 = PDU mode, 1 = Text mode)
{
    display.setTextColor(ST7735_RED);
    display.println("SMS Txt mode Error");
    delay(10000);
    ESP.restart();
    return;
}
display.println("SMS Txt mode OK");
//Exclui todos SMS armazenados
sendAT("AT + CMGD=1,4");
resetDisplay();
display.setTextColor(ST7735_WHITE);
}
```

# Código **ESP32** *sendAT e resetDisplay*

Funções  
do Setup

```
//Envia comando AT e aguarda até que uma resposta seja obtida
String sendAT(String command)
{
    String response = "";
    SerialGSM.println(command);
    // aguardamos até que haja resposta do SIM800L
    while(!SerialGSM.available());

    response = SerialGSM.readString();
    return response;
}

// limpa e configura display
void resetDisplay()
{
    display.setRotation(1);
    display.setFont(&FreeSans9pt7b);
    display.fillScreen(ST77XX_BLACK);
    display.setTextColor(ST7735_WHITE);
    display.setCursor(0,fontHeight);
}
```



# Código **ESP32** *loop*

Loop

```
void loop()
{
    String msg, number;
    // de 5 em 5 segundos, verifica se o SIM800L está desconectado, se sim, tenta reconectar
    if(timeout(5000, &millisRefCon, &flagCon))
        verifyGPRSConnection();

    if(modemGSM.isGprsConnected()) // se o SIM800L está conectado
    {
        if(isItToCall()) // função que verifica se deve-se efetuar a chamada ou não
        {
            sensorActivated = true; // sinaliza que o sensor foi ativado
            millisUserResp = millis(); // atribui à var de referencia de contagem de tempo o millis atual

            Serial.println("Sensor activated!");
            display.println("Sensor activated!");
            Serial.println("Calling to number "+String(i+1));
            display.println("Calling to number "+String(i+1));

            call(numbers[i++]); // efetua a ligação para um dos nºs do vetor, iniciando com 0
            // após a chamada soma-se 1 ao i

            if(i>=numbersTL) // se chegou ao fim do vetor, retorna ao início (0)
                i = 0;
        }
    }
}
```

# Código **ESP32** *loop (continuação)*

Loop

```
// verifica se foi recebido um SMS
if(SMSMessageRecv(&msg, &number))
{
    // exibe mensagem no display e monitor serial
    resetDisplay();
    display.println("SMS Msg Received");
    Serial.println("SMS Msg Received");
    delay(2500);

    // validamos o SMS e executamos uma ação
    executeCommand(number, msg);
}
else // exibe na serial que o modem está desconectado
    Serial.println("Disconnected");

// aguardamos 10ms
delay(10);
}
```

```
// Função que compara se o tempo foi atingido, sem que 'congele' a execução do loop
bool timeout(const int DELAY, long *previousMillis, bool *flag)
{
    if(*flag)
    {
        *previousMillis = millis();
        *flag = false;
    }

    if((*previousMillis + DELAY) < millis())
    {
        *flag = true;
        return true;
    }

    return false;
}
```

```
// verifica se o SIM800L se desconectou, se sim tenta reconectar
void verifyGPRSConnection()
{
    resetDisplay();
    display.print("GPRS: ");

    if(modemGSM.isGprsConnected())
        display.println("Connected");
    else
    {
        display.println("Disconnect");
        display.println("Reconnecting...");
        if(!modemGSM.waitForNetwork())
        {
            display.setTextColor(ST7735_RED);
            display.println("GPRS Con. Failed");

            delay(5000);
            display.setTextColor(ST7735_WHITE);
        }
    }
}
```

# Código **ESP32** *verifiGPRSConnection (continuação)*

Funções  
do Loop

```
else
{
  if(!modemGSM.gprsConnect(APN,USER,PASSWORD))
  {
    display.setTextColor(ST7735_RED);
    display.println("GPRS Con. Failed");
    delay(5000);
    display.setTextColor(ST7735_WHITE);
  }
  else
  {
    display.setTextColor(ST7735_GREEN);
    display.println("GPRS Con. OK");
  }
}
}
```



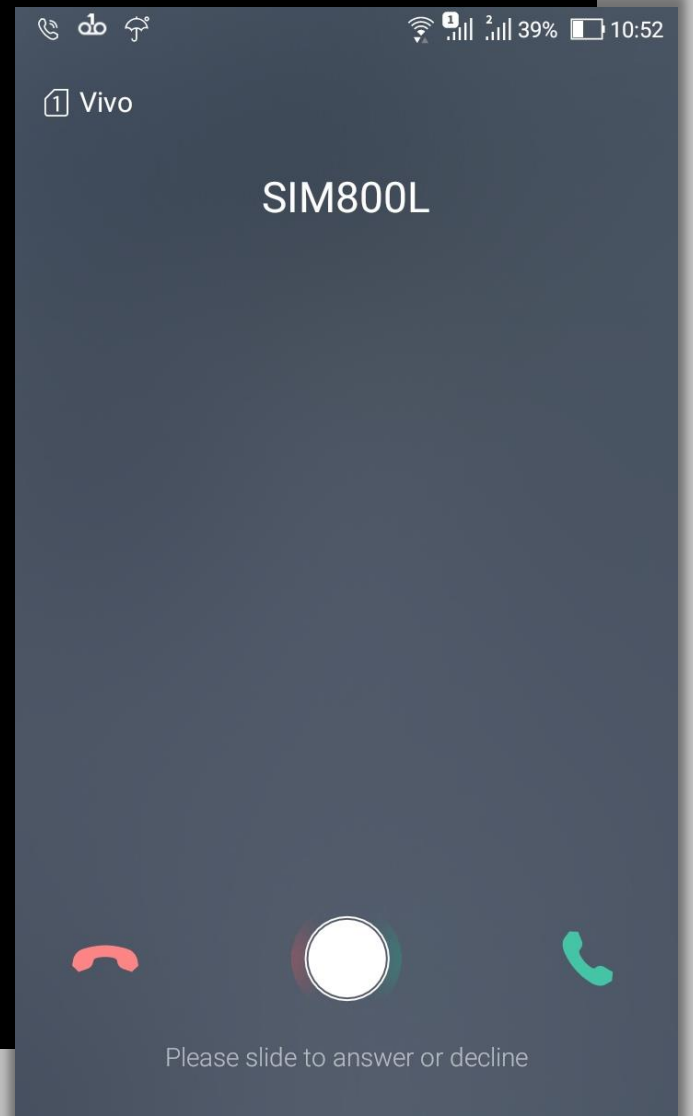
# Código **ESP32** *isItToCall e call*

Funções  
do Loop

```
bool isItToCall()
{
    // se o sensor de barreira está ativo ou foi uma vez ativado e não foi recebido um SMS
    // em 1 min, retornamos true indicando que deve-se ligar parao próx número
    return digitalRead(sensorPin) == HIGH || (sensorActivated && !userResponseSMS &&
    timeout(60000, &millisUserResp, &flagUserResp));
}

// executa chamada
void call(String number)
{
    display.print("Calling...");
    Serial.println(number);
    // tenta executar chamada
    bool res = modemGSM.callNumber(number);

    // se obteve sucesso exibe OK, se não, fail
    if(res)
        display.println(" OK");
    else
        display.println(" fail");
}
```



```
if (res)
{
    // assim que a chamada for feita é finalizada
    res = modemGSM.callHangup();
    // exibe se foi possível finalizar ou não
    display.print("Hang up: ");
    if(res)
        display.println("OK");
    else
        display.println("fail");
}
}

// verifica se um sms é recebido e obtém o número de quem o enviou
bool SMSMessageRecv(String *msg, String *number)
{
    // comando AT que lista todos os SMS armazenados
    *msg = sendAT("AT+CMGL=\"ALL\"");

    // se o SIM800L responder com SM, significa que um novo SMS acaba de chegar
    // então pedimos novamente que o SIM nos liste os SMS armazenados
    if((*msg).indexOf("+CMTI: \"SM\"")>=0)
        *msg = sendAT("AT+CMGL=\"ALL\"");
}
```

# Código **ESP32** *SMSSMessageRecv (continuação)*

Funções  
do Loop

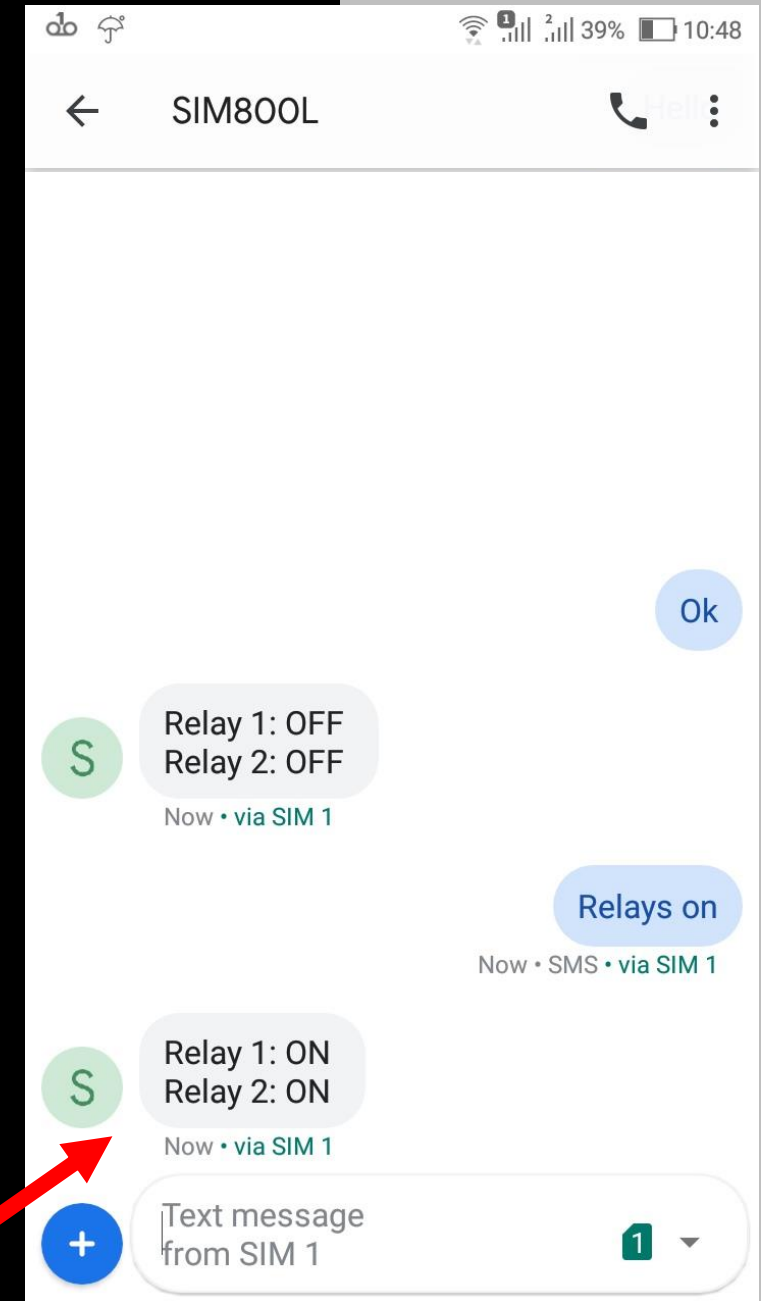
```
// se a mensagem possui um OK e possui mais que 10 caracteres
(existe pelo menos um SMS)
if((*msg).indexOf("OK")>=0 && (*msg).length()>10)
{
    // exibe a mensagem na serial
    Serial.println(*msg);

    // obtém o número que nos enviou o SMS e exibe se obteve sucesso
    if(getSMSNumber(*msg, *&number))
    {
        Serial.println("numero obtido: "+*number);
        return true;
    }
    else
    {
        Serial.println("Erro ao obter numero");
        return false;
    }
}
// exibe na serial um ponto (debug)
Serial.print(".");
return false;
}
```

# Código **ESP32** *executeCommand*

Funções  
do Loop

```
void executeCommand(String number, String msg)
{
    // se o número não é válido, exibe mensagem e não faz nada
    if(!numberIsValid(number))
    {
        display.println("Number is not valid");
        Serial.println("Number is not valid");
        Serial.println(number);
        delay(2500);
    }
    else// se o número é válido
    {
        getTextSMS(&msg); // obtem o texto do SMS recebido
        Serial.println(msg);
        if(commandOK(number, msg)) // executa comando conforme SMS
        {
            // sinaliza que o usuario respondeu com um SMS válido
            userResponseSMS = true;
            // retorna para false a flag que indica se sensor está ativo
            sensorActivated = false;
            display.println("Sending status");
            Serial.println("Sending status");
            sendResponse(number); // Envia o estado dos dois relés por SMS
        }
    }
}
```



# Código **ESP32** *executeCommand (continuação)*

Funções  
do Loop

```
else
{
    // se o comando é inválido, exibe no display
    // sinaliza que o usuário respondeu com um SMS inválido
    userResponseSMS = false;

    display.println("Cmd is not valid");
    Serial.println("Cmd is not valid");
}
}
// exclui todos os SMS armazenados
sendAT("AT + CMGD=1,4");
}
```



# Código **ESP32** *getTextSMS*

Funções  
do Loop

```
// obtém texto da mensagem e o retorna por parâmetro
void getTextSMS(String *msg)
{
    String aux;
    /*
        Exemplo de mensagem:
        [
        +CMGL: 1,"REC UNREAD","+5518999999999","","18/11/30,11:36:14-08"
        Hello

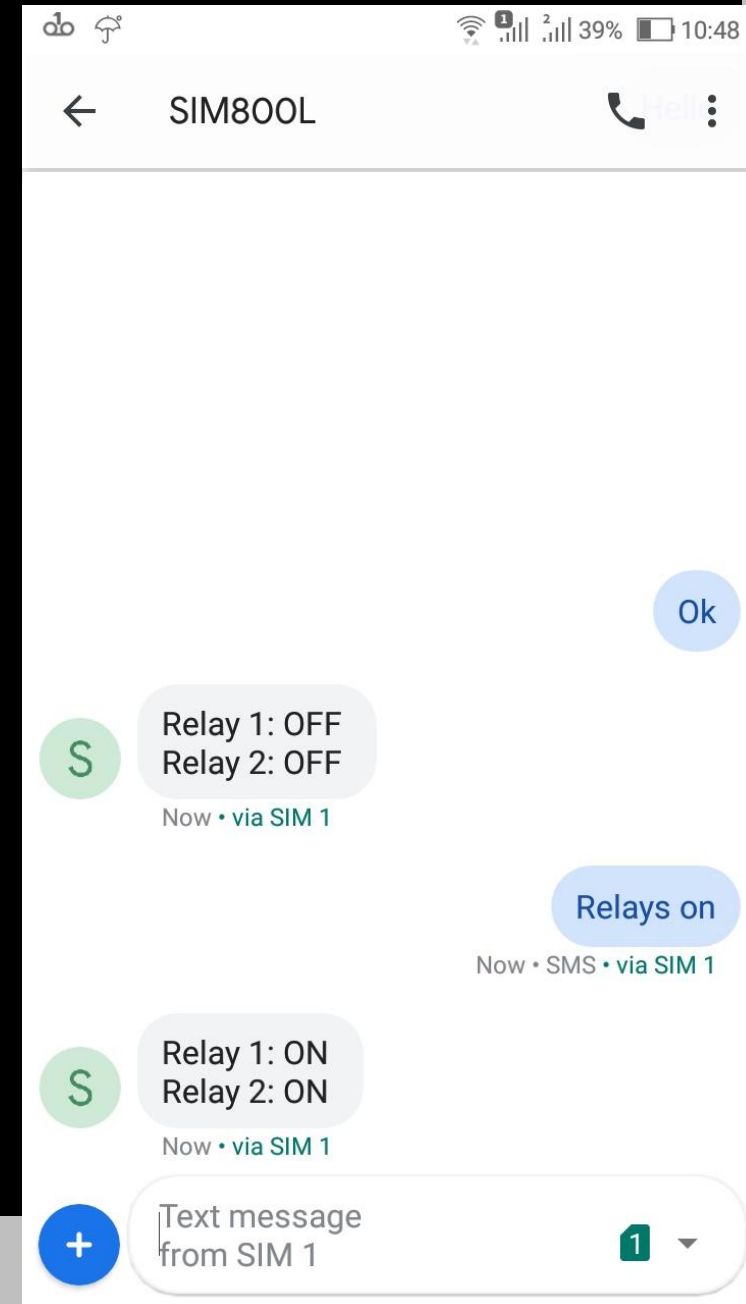
        OK
        ]
    */
    //pula primeiro \n
    *msg = (*msg).substring((*msg).indexOf( "\n" )+1);
    //pula primeira linha: 'Cmd: +CMGL: 1,"REC UNREAD","+5518999999999","","18/11/30,11:04:30-08" '
    *msg = (*msg).substring((*msg).indexOf( "\n" )+1);
    aux = *msg;
    if(aux.length() <= 8)
        return;
    *msg = "";
    // retira a substring "\r\n\r\nOK\r\n" (8 caracteres)
    for(int i=0; i<aux.length()-8; i++)
        *msg += aux.charAt(i);
}
```

# Código **ESP32** *commandOK*

## Funções do Loop

```
// executa comando de acordo com a variavel 'msg'
bool commandOK(String number, String msg)
{
    // flag que indica se entrou em algum if (indicando comando válido)
    bool smsOK = false;

    // verifica se a msg corresponde a algum dos comandos existentes e seta
    os pinos correspondentes
    // os reles possuem lógica inversa
    if(msg.equalsIgnoreCase("relay 1 on"))
    {
        digitalWrite(relayPin1, LOW);
        smsOK = true;
    }
    else
    if(msg.equalsIgnoreCase("relay 1 off"))
    {
        digitalWrite(relayPin1, HIGH);
        smsOK = true;
    }
}
```



# Código **ESP32** *commandOK (continuação)*

Funções  
do Loop

```
else
if(msg.equalsIgnoreCase("relay 2 on"))
{
    digitalWrite(relayPin2, LOW);
    smsOK = true;
}
else
if(msg.equalsIgnoreCase("relay 2 off"))
{
    digitalWrite(relayPin2, HIGH);
    smsOK = true;
}
else
if(msg.equalsIgnoreCase("relays off"))
{
    digitalWrite(relayPin1, HIGH);
    digitalWrite(relayPin2, HIGH); smsOK = true;
}
else
if(msg.equalsIgnoreCase("relays on"))
{
    digitalWrite(relayPin1, LOW);
    digitalWrite(relayPin2, LOW); smsOK = true;
}
```

# Código **ESP32** *commandOK (continuação)*

Funções  
do Loop

```
else
// comando OK, usado para sinalizar alarme falso
if(msg.equalsIgnoreCase("ok"))
    smsOK = true;
else
// comando hello, usado para verificar o funcionamento
if(msg.equalsIgnoreCase("hello"))
{
    modemGSM.sendSMS(number, "Hello!");
    smsOK = true;
}
else
// comando status que obtém os estados dos pinos
if(msg.equalsIgnoreCase("status"))
    smsOK = true;

return smsOK;
}
```

Em [www.fernandok.com](http://www.fernandok.com)

Download arquivos PDF e **INO** do código fonte

