



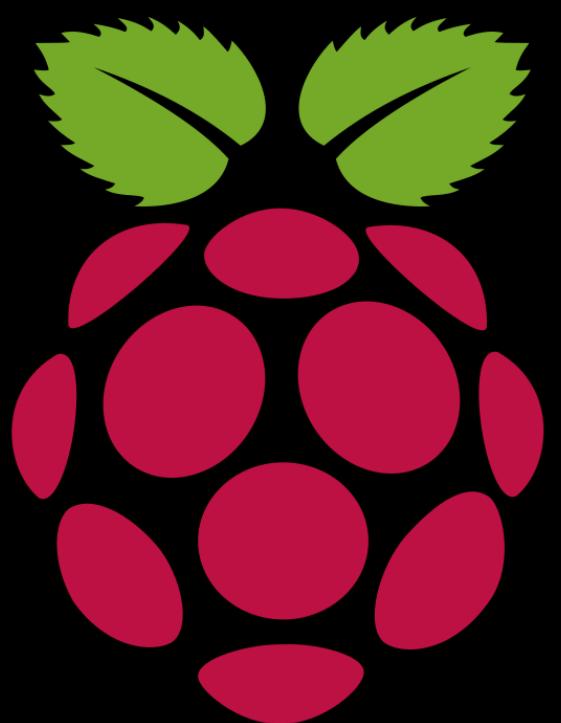
# Projeto 01

## Coisas Configuráveis – Teoria

Jan K. S. – [janks@puc-rio.br](mailto:janks@puc-rio.br)

ENG4051 – Projeto Internet das Coisas

Coisas



RaspberryPi



ESP32



Microcontrolador Usado no Curso



ESP32



ESP32-CAM



ESP32-S3 N32R8



ESP32-S3-BOX-3B



ESP32-S3-CAM

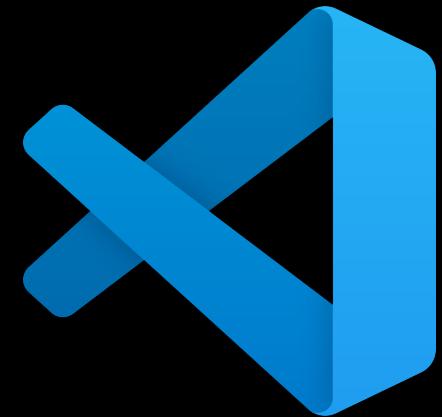


LILYGO ESP32 T-Display

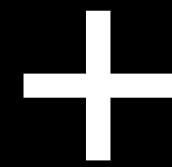
Exemplos de Modelos ESP32 (DevKits)

	Raspberry Pi 5	ESP32-S3-Cam	Arduino Mega (clone)
Processador	4 núcleos, 2.4GHz	2 núcleos, 240MHz	1 núcleo, 16MHz
RAM	4GB ou 8GB	512KB SRAM + 8MB PSRAM	8KB SRAM
Armazenamento	SD (até 2TB)	16MB Flash + SD (até 32GB)	256KB Flash
Pinos Digitais	28 pinos	34 pinos	54 pinos
Pinos Analógicos	não tem	20 pinos (12 bits)	16 pinos (10 bits)
Pinos PWM	1 pino	8 pinos	15 pinos
Tensão nos Pinos	3.3V	3.3V	5.0V
Wi-Fi	2.4GHz e 5GHz	2.4GHz	não tem
Bluetooth	5.0	5.0	não tem
Ethernet	Gigabit	não tem	não tem
Consumo Ativo	~1A - 2A	~160mA - 260mA	~70mA - 100mA
Consumo Repouso	~600mA - 800mA	~20µA - 2.5mA (deep sleep)	~50mA
Preço (2024)	R\$ 650 - R\$ 1000	R\$ 130	R\$ 90

Comparativo entre Raspberry Pi, ESP32 e Arduino Mega



# VS Code



# PlatformIO

The screenshot shows the Visual Studio Code interface with the following details:

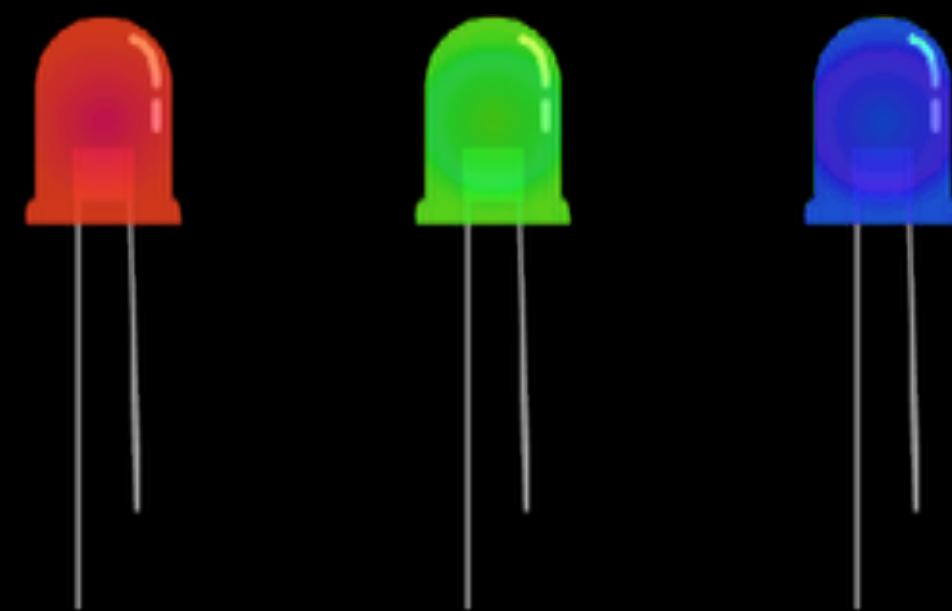
- EXPLORADOR:** Shows the project structure:
  - PROJETO01
    - .pio
    - .vscode
    - data
    - pdfs
    - src
      - 01a\_testes\_iniciais.cpp
      - 01b\_prototipo.cpp
      - 01c\_integracao.cpp
      - 01d\_desafio.cpp
      - certificados.h
    - .gitignore
    - partitions.csv
    - platformio.ini
  - ESTRUTURA DO CÓDIGO
  - LINHA DO TEMPO
- CODE EDITOR:** Displays the file `01a_testes_iniciais.cpp` with the following code:

```
1 void setup() {  
2 }  
3 }  
4  
5 void loop() {  
6 }  
7 }  
8 }
```
- STATUS BAR:** Shows icons for file status (0 errors, 0 warnings), home, search, replace, and file operations, followed by "Default (Projeto01)", "Auto", and "PlatformIO".

A programação é a  
mesma do Arduino,  
só que numa IDE  
mais rápida.



Programação via Visual Studio Code + PlatformIO (Extensão)



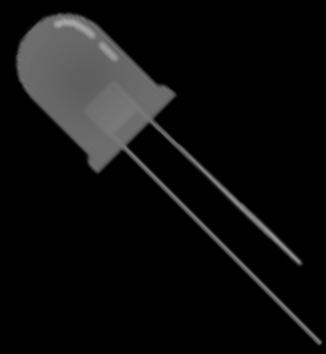
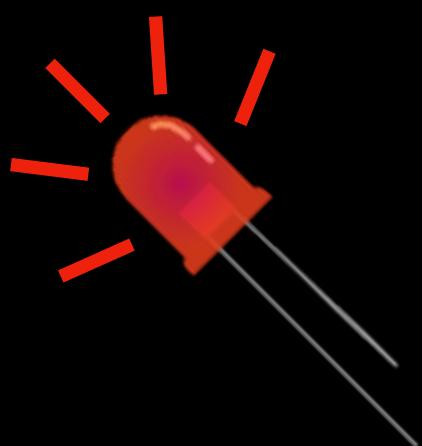
LED

```
int pinoDoLED = 40;

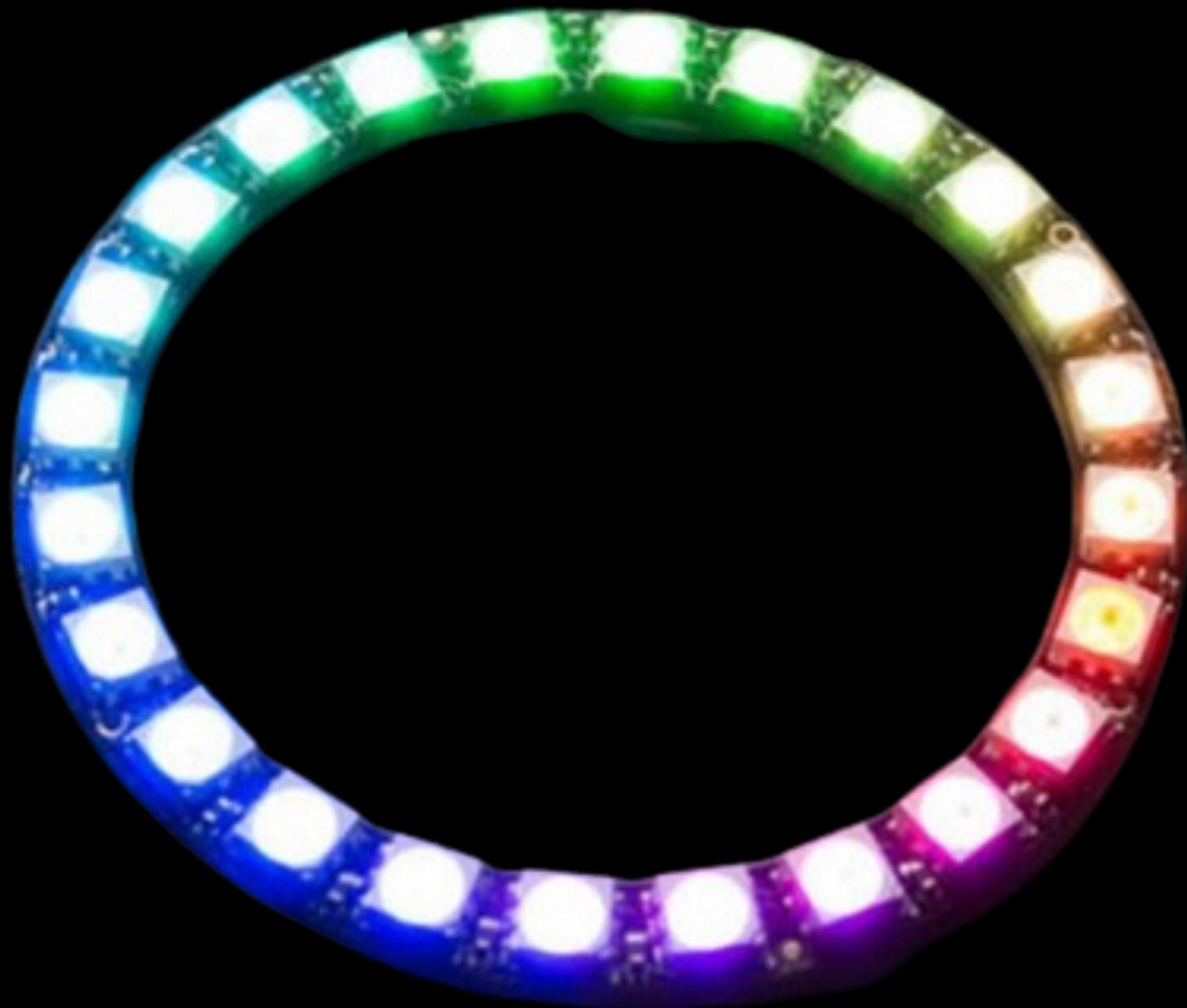
void setup () {
    pinMode(pinoDoLED, OUTPUT);
    digitalWrite(pinoDoLED, HIGH); // começa apagado
}

// em alguma função do código...
digitalWrite(pinoDoLED, LOW);

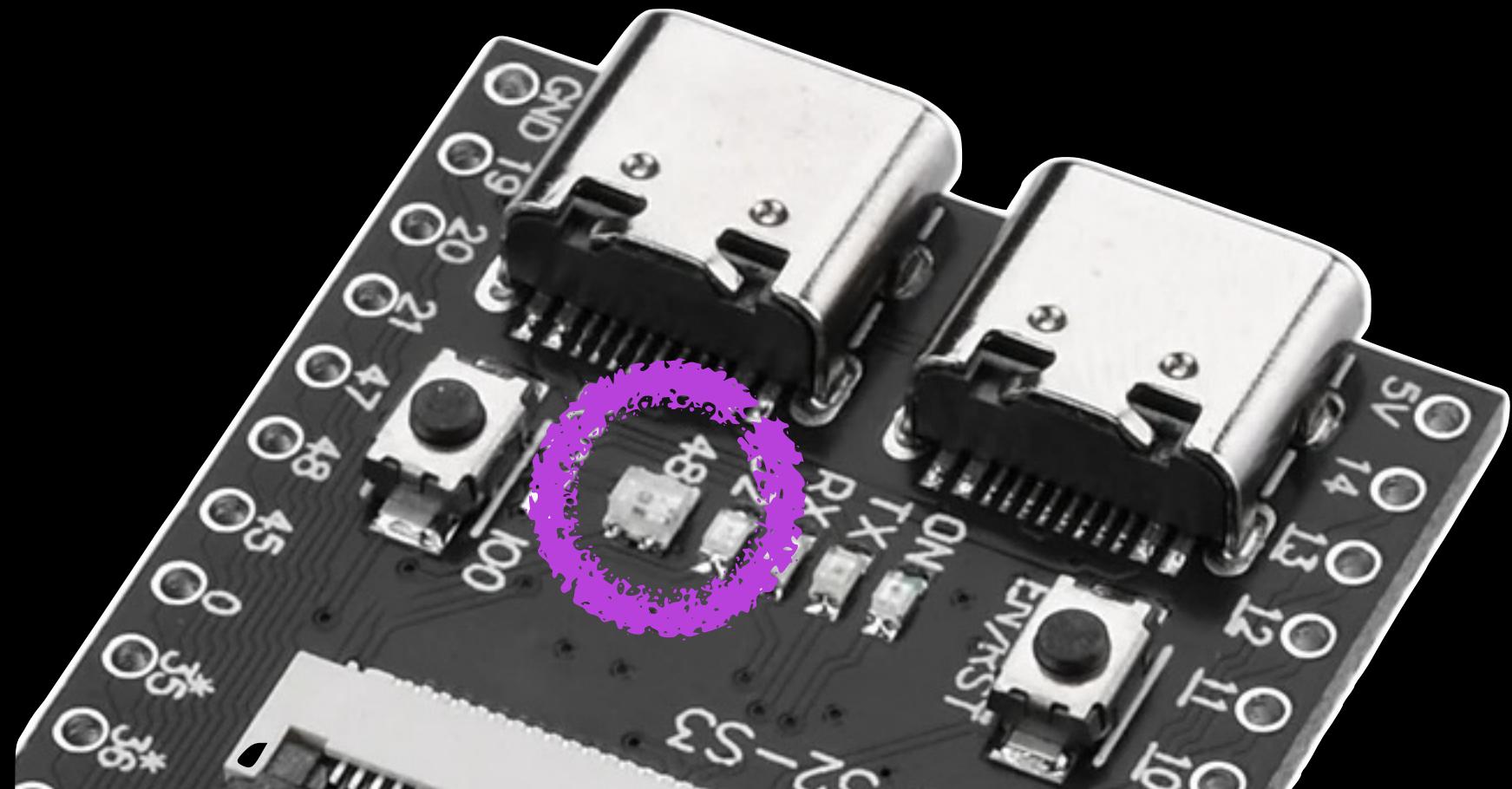
digitalWrite(pinoDoLED, HIGH);
```



Código para Acender e Apagar LEDs



LED RGB Neopixel



O ESP32 S3 vêm com um pequeno LED RGB embutido controlado apenas por 1 pino (48).



bit 0



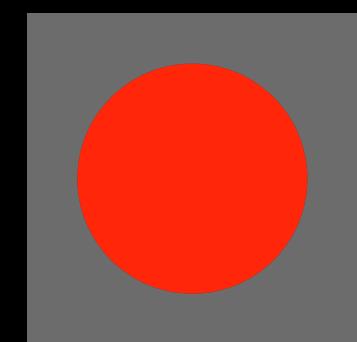
ESP32

G

R

B

00000000 11111111 00000000



bit 1



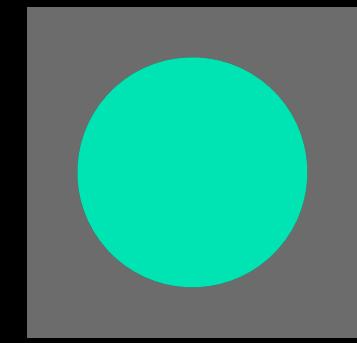
ESP32

G

R

B

11100000 00000000 10110110



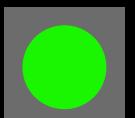
Codificação para a Cor do LED

```
rgbLedWrite(RGB_BUILTIN, vermelho, verde, azul);
```

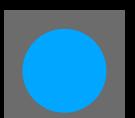
```
rgbLedWrite(RGB_BUILTIN, 255, 0, 0);
```



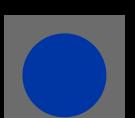
```
rgbLedWrite(RGB_BUILTIN, 0, 255, 0);
```



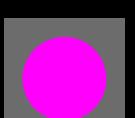
```
rgbLedWrite(RGB_BUILTIN, 0, 0, 255);
```



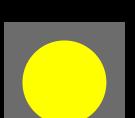
```
rgbLedWrite(RGB_BUILTIN, 0, 0, 80);
```



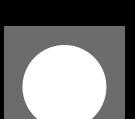
```
rgbLedWrite(RGB_BUILTIN, 255, 0, 255);
```



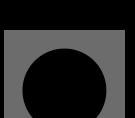
```
rgbLedWrite(RGB_BUILTIN, 255, 255, 0);
```

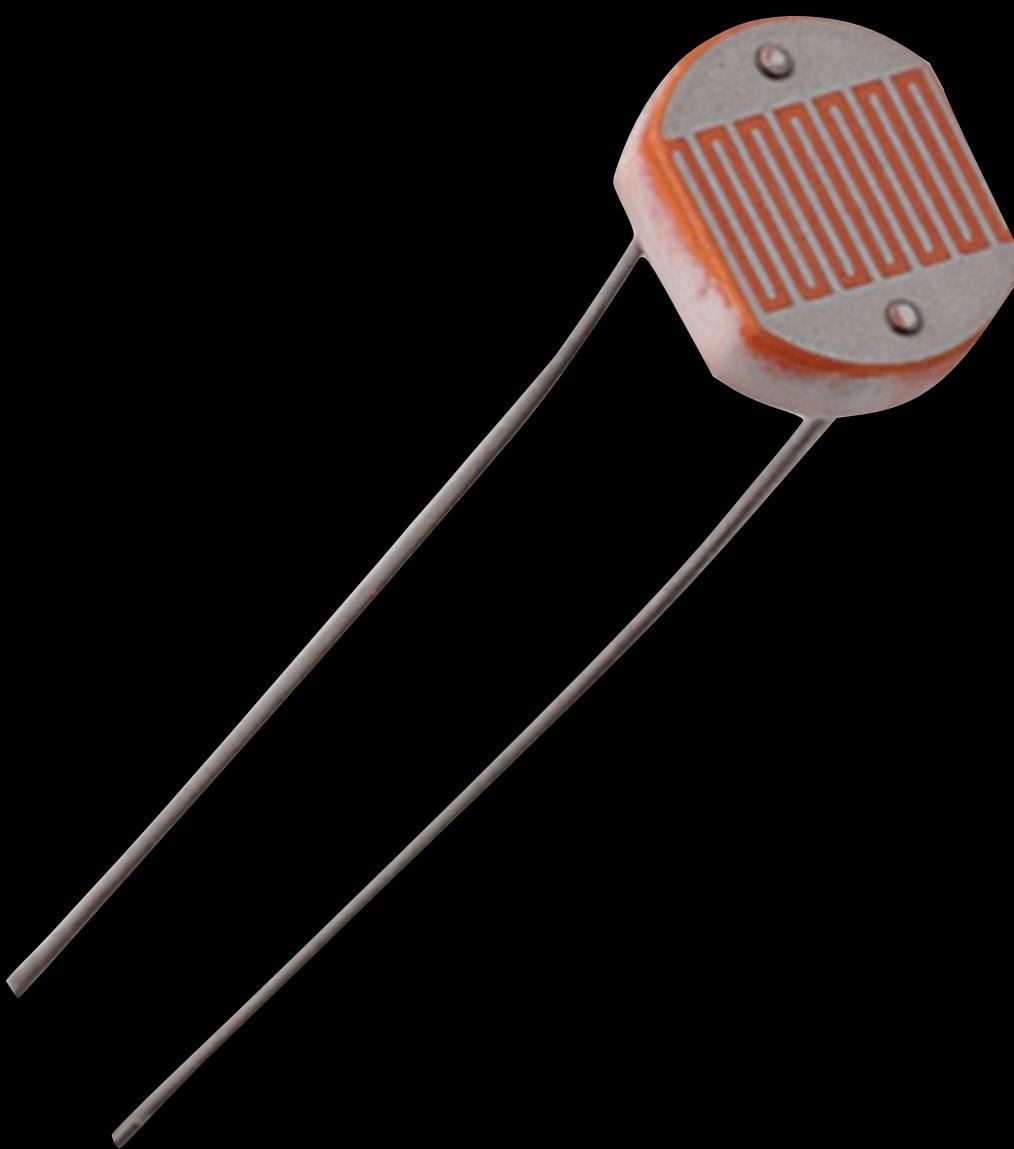


```
rgbLedWrite(RGB_BUILTIN, 255, 255, 255);
```

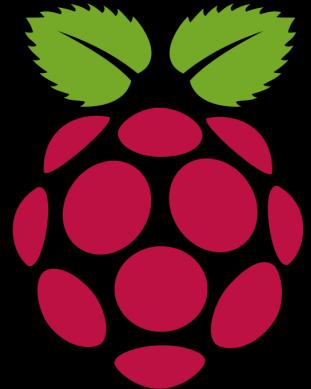


```
rgbLedWrite(RGB_BUILTIN, 0, 0, 0);
```

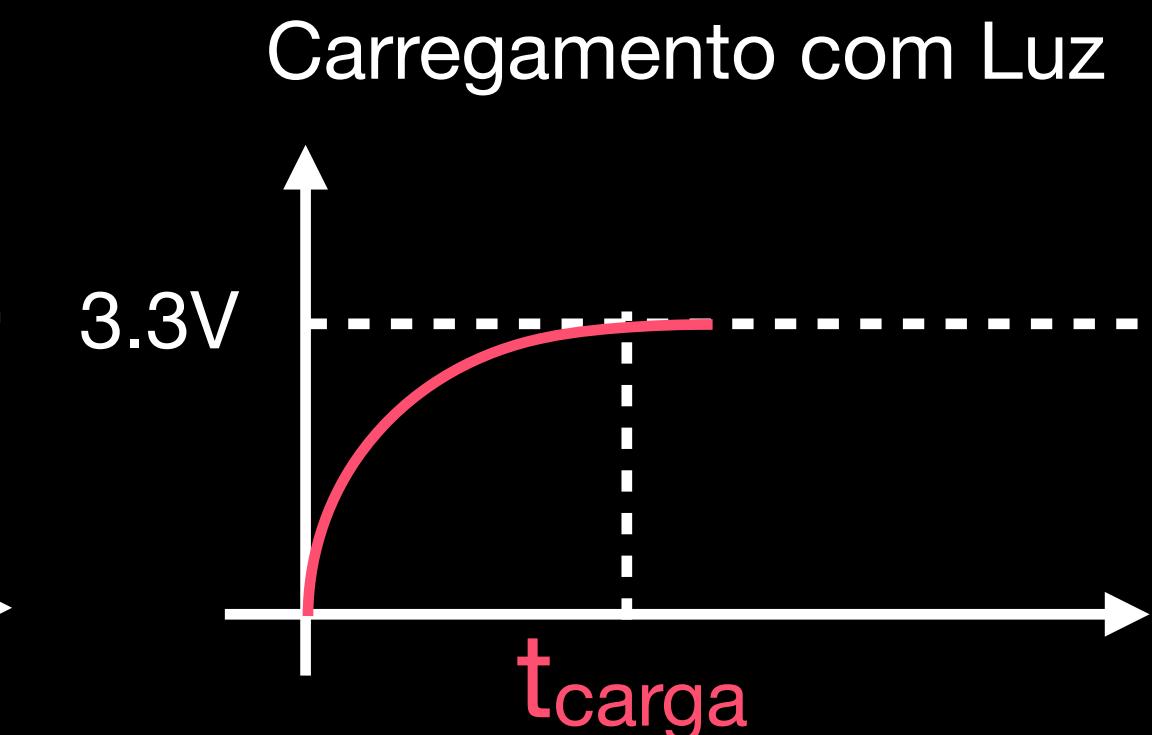
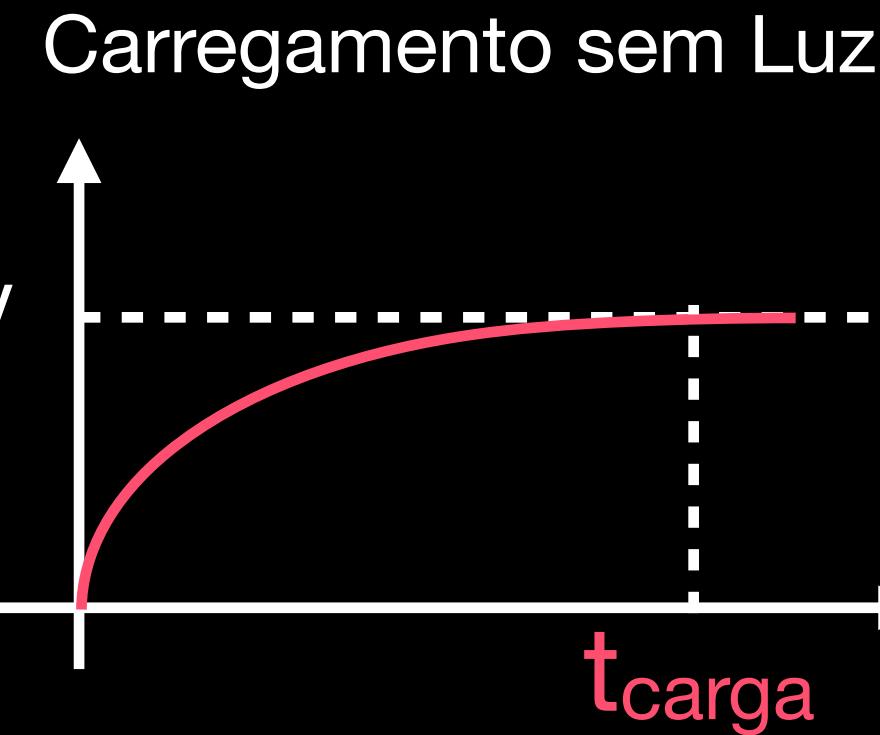
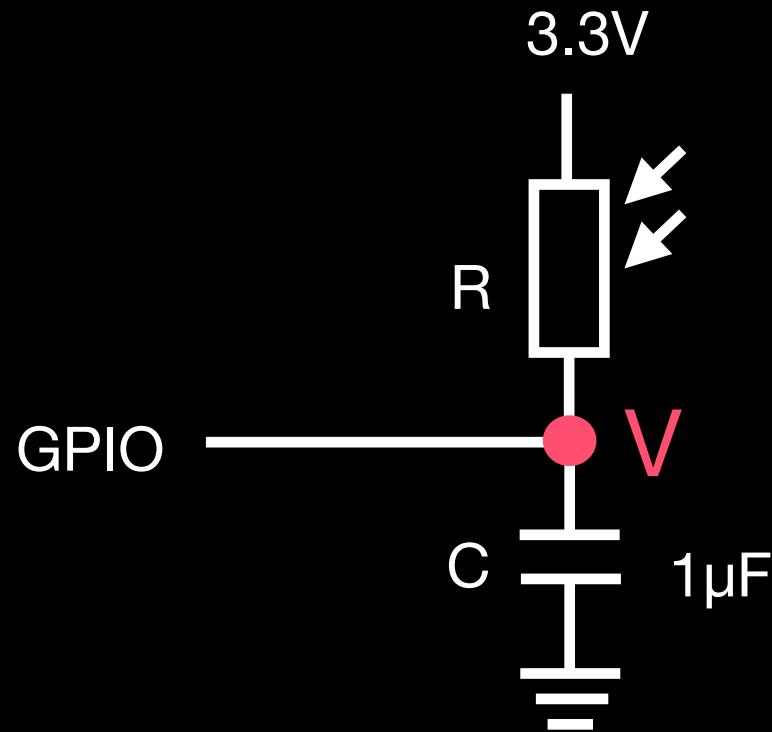




Sensor de Luz (LDR)



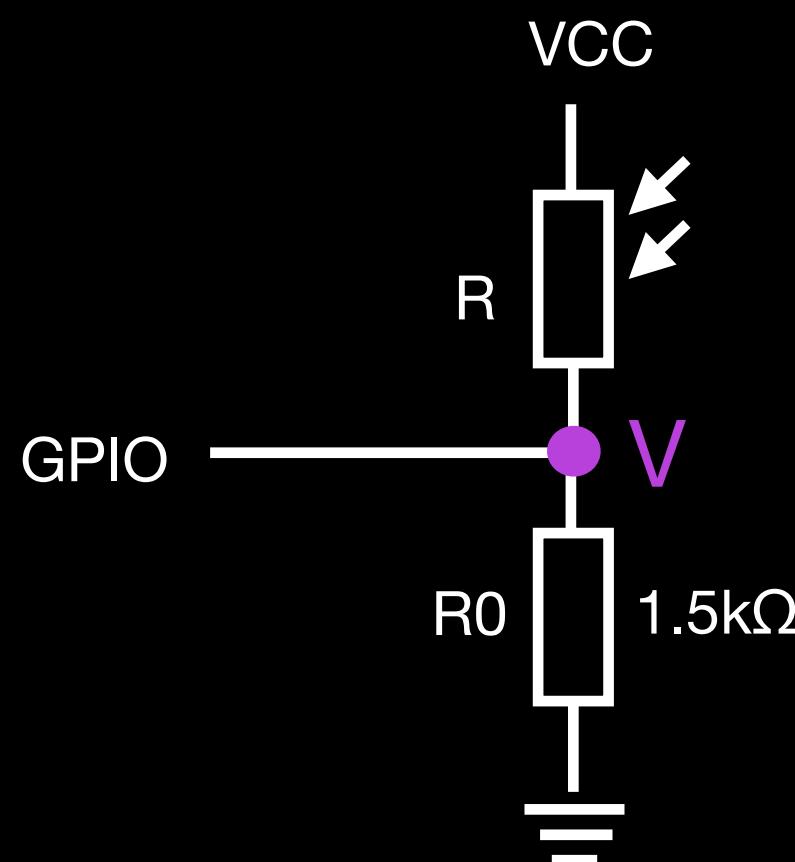
RaspberryPi



ARDUINO



ESP32



$$V = V_0 \frac{R_0}{R + R_0}$$

Leitura do Sensor de Luz no Raspberry Pi x Arduino / ESP32



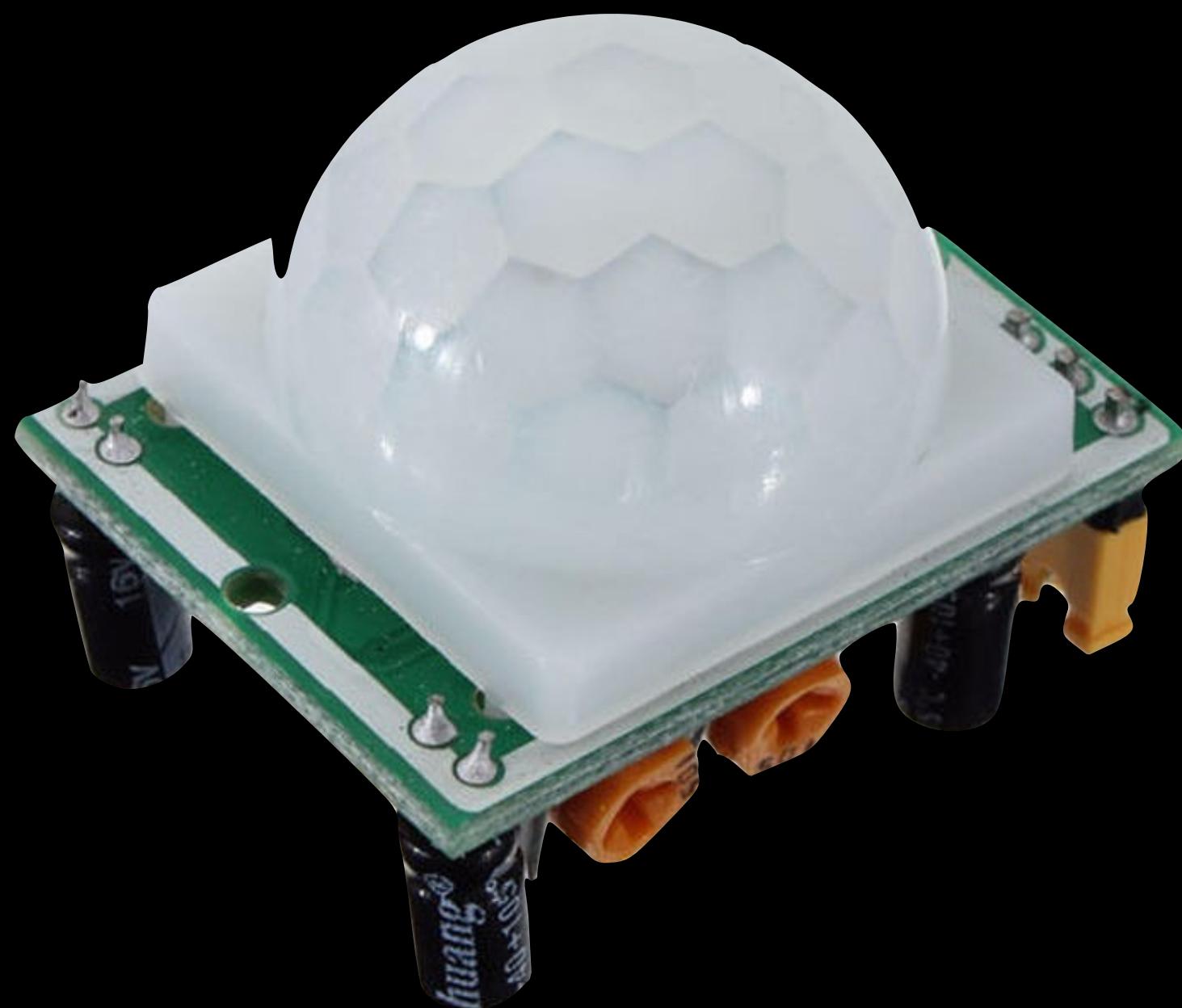
Conversor Analógico-Digital  
de 10 bits

```
int leituraAnalogica = analogRead(pino); // entre 0 e 1023  
int porcentagemLuz = map(leituraAnalogica, 0, 1023, 0, 100);
```

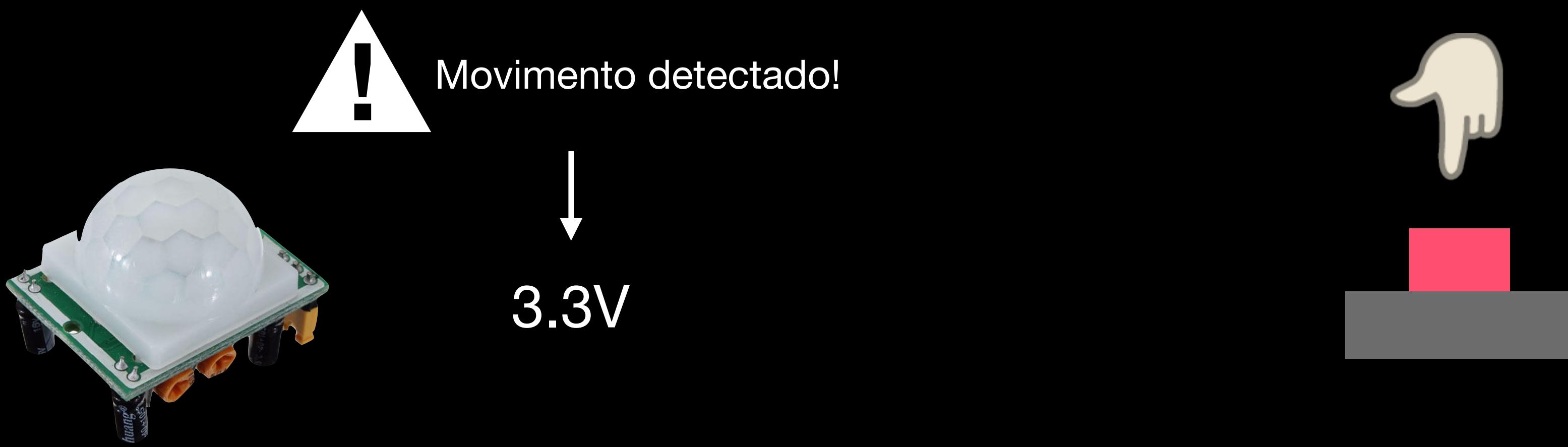


Conversor Analógico-Digital  
de 12 bits

```
int leituraAnalogica = analogRead(pino); // entre 0 e 4095  
int porcentagemLuz = map(leituraAnalogica, 0, 4095, 0, 100);
```



Sensor de Movimento (HC-SR501)



Equivalência dos Eventos do Sensor de Movimento com os de um Botão

```
#include <GButton.h>

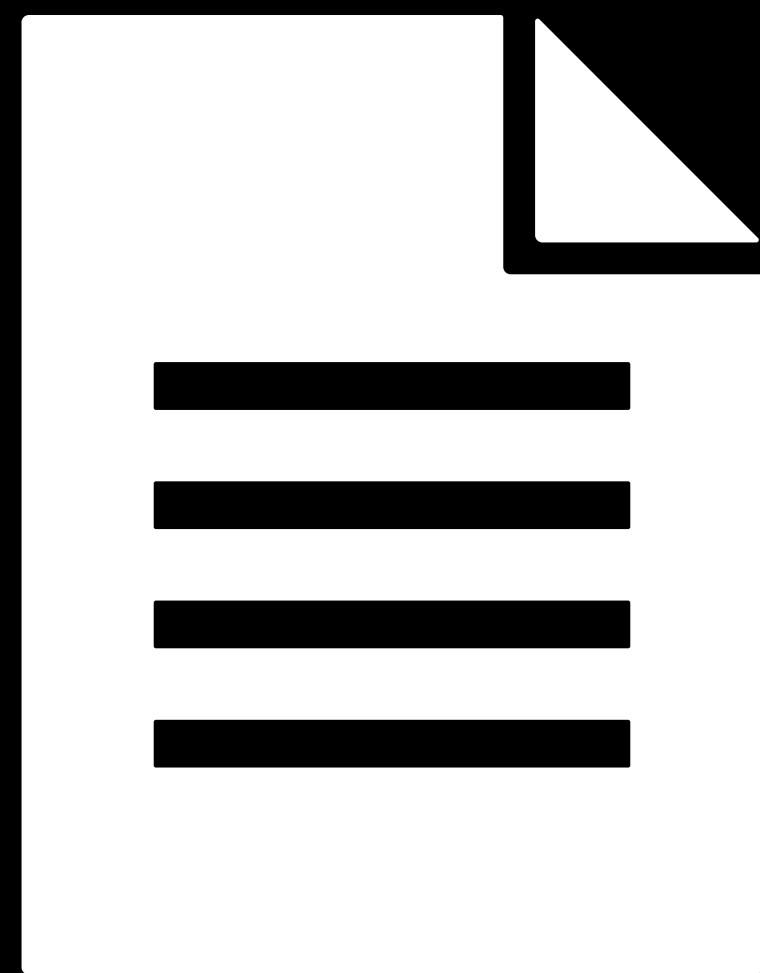
GButton sensorDeMovimento(21);

void movimentoDetectado (GButton& sensor) {
    Serial.println("Detectei movimento!");
}

void inerciaDetectada (GButton& sensor) {
    Serial.println("Detectei inércia!");
}

void setup () {
    Serial.begin(115200);
    sensorDeMovimento.setReleaseHandler(movimentoDetectado);
    sensorDeMovimento.setPressHandler(inerziaDetectada);
}

void loop () {
    sensorDeMovimento.process();
}
```



Arquivos

# memória Flash

nvs (dados não voláteis)

ota (atualização remota)

app0 (aplicação)

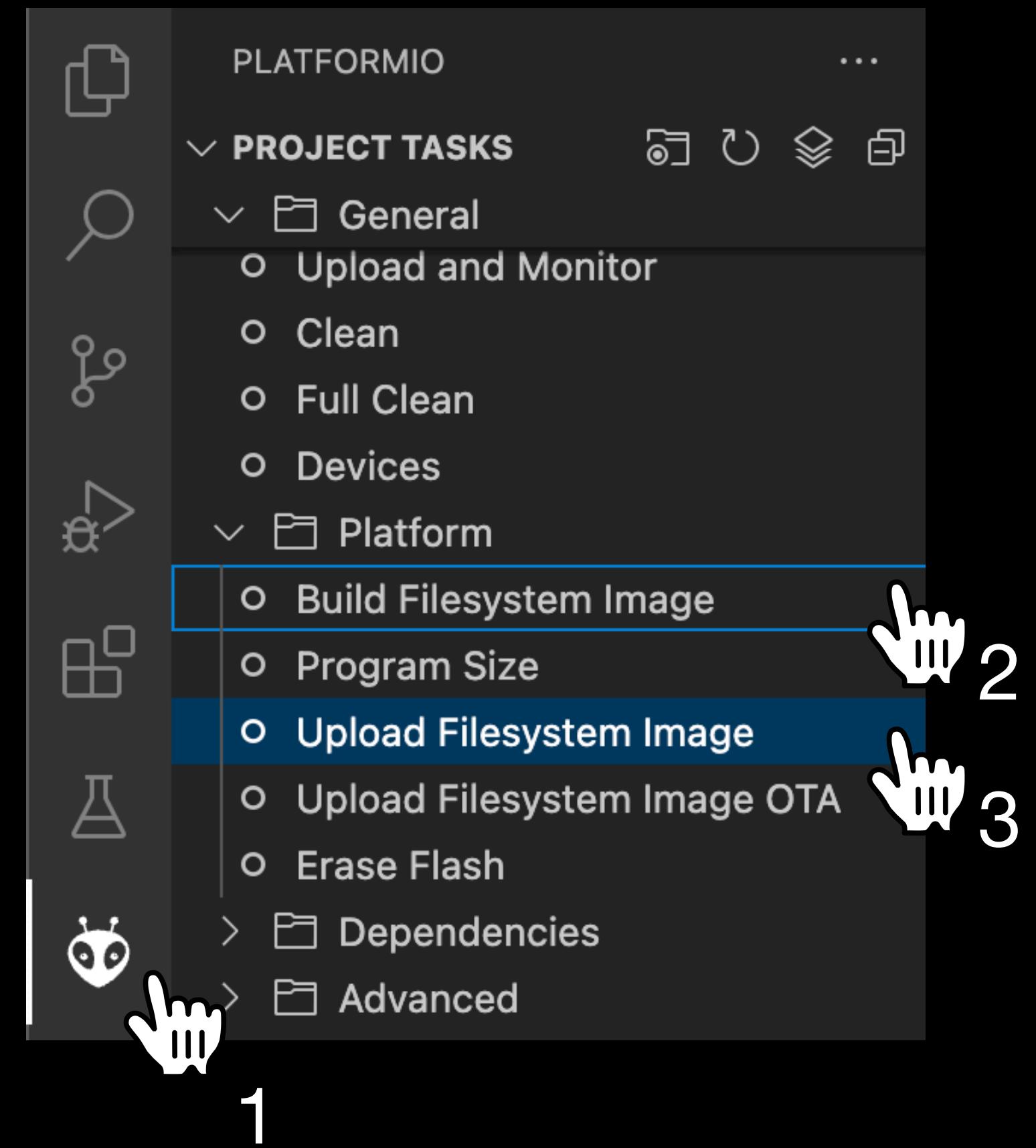
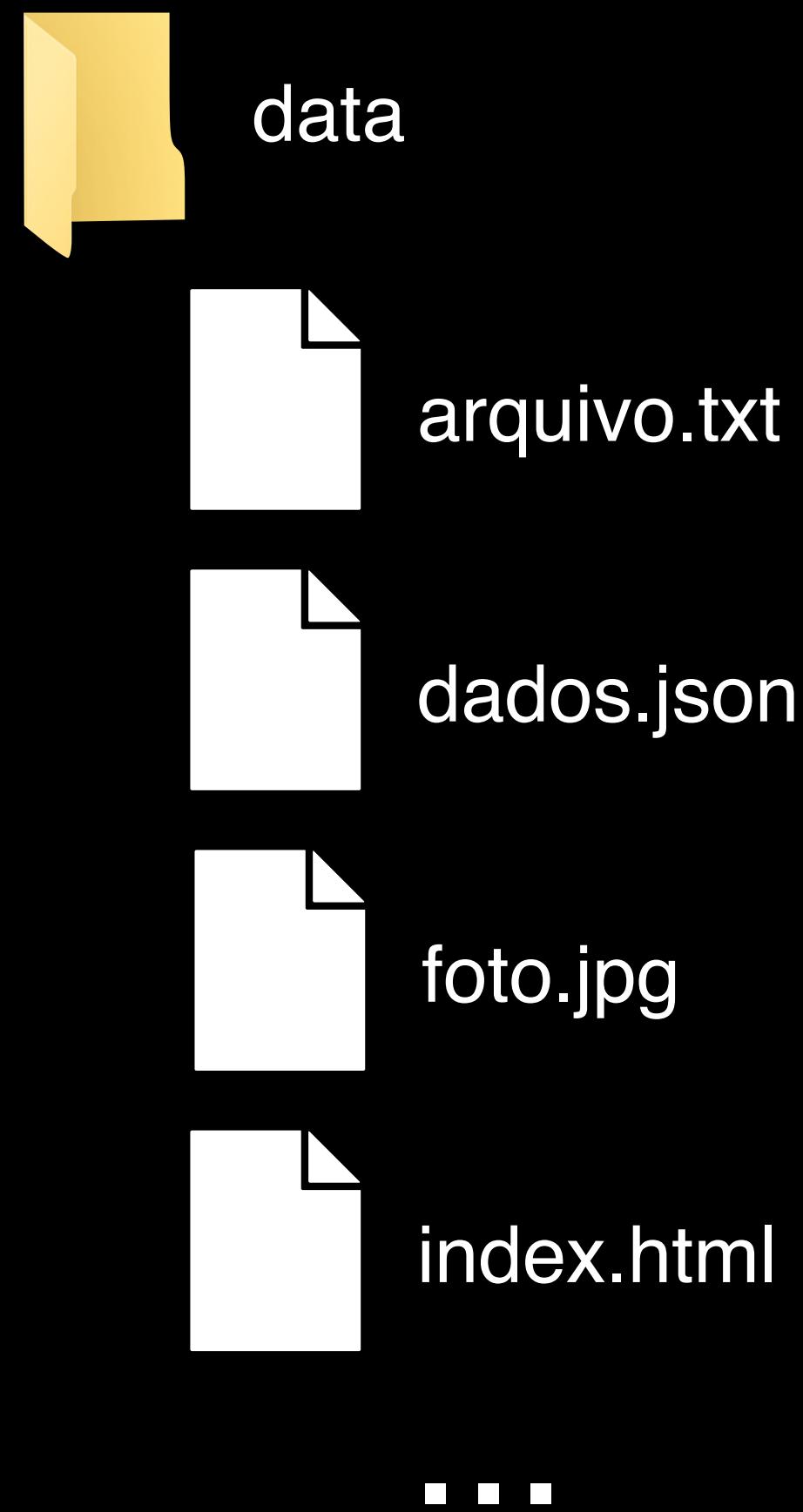
app1 (outra aplicação)

spiffs (arquivos)

coredump (debug)

## partitions.csv

#	Name,	Type,	SubType,	Offset,	Size,	Flags
	nvs,	data,	nvs,	0x9000,	0x5000,	
	otadata,	data,	ota,	0xe000,	0x2000,	
	app0,	app,	ota_0,	0x10000,	0x640000,	
	app1,	app,	ota_1,	0x650000,	0x640000,	
	spiffs,	data,	spiffs,	0xc90000,	0x360000,	
	coredump,	data,	coredump,	0xFF0000,	0x10000,	



Transferência de Arquivos para o ESP-32 via PlatformIO no Visual Studio Code

```
#include <LittleFS.h>

void setup() {
    Serial.begin(115200);
    delay(500);

    if (!LittleFS.begin()) {
        Serial.println("Falha ao montar o sistema de arquivos!");
        while (true) {} // trava programa aqui em caso de erro
    }
}

// em alguma função do código...

File arquivo = LittleFS.open("/arquivo.txt", "r"); // "read"
if (!arquivo) {
    Serial.println("Erro ao abrir o arquivo!");
    while (true) {} // trava programa aqui
}
String conteudo = arquivo.readString();
arquivo.close();

Serial.println(conteudo);
```

```
#include <LittleFS.h>

void setup() {
    Serial.begin(115200);
    delay(500);

    if (!LittleFS.begin()) {
        Serial.println("Falha ao montar o sistema de arquivos!");
        while (true) {} // trava programa aqui em caso de erro
    }
}

// em alguma função do código...

File arquivo = LittleFS.open("/arquivo2.txt", "w"); // "write"
arquivo.println("IoT");
arquivo.print("PUC-Rio ");
arquivo.println(2020 + 4);
arquivo.close();
```



```
void setup() {
    // inicia o LittleFS, que nem antes...
}

        // em alguma função do código...

File arquivo = LittleFS.open("/dados.json", "r"); // "read"
if (!arquivo) {
    Serial.println("Erro ao abrir o arquivo JSON!");
    while (true) {} // trava programa aqui em caso de erro
}
JsonDocument dados;
deserializeJson(dados, arquivo);
arquivo.close();

        // em alguma função do código...

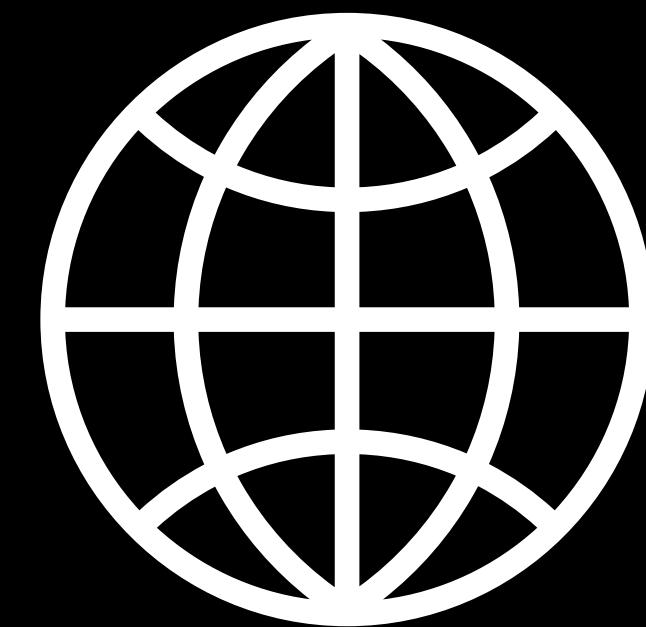
JsonDocument dados2;
dados2["nome"] = "Jan K. S.";
dados2["status"] = "cansado";

File arquivo = LittleFS.open("/dados2.json", "w"); // "write"
serializeJson(dados2, arquivo);
arquivo.close();
```

# Internet



Servidor Web no ESP32



1. Rede WiFi

2. Servidor

2 Etapas para Criar um Servidor

Vamos sempre chamar essa função  
na setup e no loop!



```
#include <WiFi.h>

void reconnectarWiFi() {
    if (WiFi.status() != WL_CONNECTED) {
        WiFi.begin("NOME DA REDE WIFI", "SENHA DA REDE WIFI");

        Serial.print("Conectando ao WiFi...");
        while (WiFi.status() != WL_CONNECTED) {
            Serial.print(".");
            delay(1000);
        }

        Serial.print("conectado!\nEndereço IP: ");
        Serial.println(WiFi.localIP());
    }
}
```

Conectado ao WiFi..... conectado!  
Endereço IP: 192.168.0.XXX

```
#include <WebServer.h>
```

Conectado ao WiFi..... conectado!  
Endereço IP: 192.168.0.XXX

```
WebServer servidor(80); // porta 80 (padrão do HTTP)
```

```
void paginaInicial () {  
    servidor.send(200, "text/html", "Bem-vindo!");  
}
```



```
void setup () {  
    Serial.begin(115200);  
    reconnectWiFi(); // função mencionada anteriormente
```

Enviamos o código HTTP 200 ao navegador para indicar que deu tudo certo na página.

```
    servidor.on("/inicio", HTTP_GET, paginaInicial);  
    servidor.begin();  
}
```

```
void loop () {  
    reconnectWiFi();  
    servidor.handleClient();  
}
```

192.168.0.XXX/inicio

Bem-vindo!

Exemplo de Rota Simples

```
#include <WebServer.h>
#include <uri/UriBraces.h>
```

Conectado ao WiFi..... conectado!  
Endereço IP: 192.168.0.XXX

```
WebServer servidor(80); // porta 80 (padrão do HTTP)
```

```
void paginaComParametros () {
    int x = servidor.pathArg(0).toInt();
    int y = servidor.pathArg(1).toInt();
    servidor.send(200, "text/html", String(x+y));
}
```

```
void setup () {
    Serial.begin(115200);
    reconnectWiFi(); // função mencionada anteriormente
    servidor.on(UriBraces("/soma/{}/{}"), HTTP_GET, paginaComParametros);
    servidor.begin();
}
```

192.168.0.XXX/soma/3/5

```
void loop () {
    reconnectWiFi();
    servidor.handleClient();
}
```

8

Exemplo de Rota com Parâmetros



```
#include <LittleFS.h>

void paginaComArquivoHTML () {
    File arquivo = LittleFS.open("/pagina.html", "r");
    if (!arquivo) {
        servidor.send(500, "text/html", "Erro ao abrir HTML");
        return;
    }
    String html = arquivo.readString();
    arquivo.close();

    html.replace("{{nome}}", "Jan K. S.");
    int pontos = 10000;
    html.replace("{{pontos}}", String(pontos));

    servidor.send(200, "text/html", html);
}

void setup () {
    // inicia Serial, WiFi, LittleFS...

    servidor.on("/pontuacao", HTTP_GET, paginaComArquivoHTML);
    servidor.begin();
}
```

```
<h1>{{nome}}</h1>
<p>{{pontos}} XP</p>
```

192.168.0.XXX/pontuacao

**Jan K. S.**  
10000 XP

```
#include <LittleFS.h>

void paginaComArquivoHTML () {
    // mesma lógica de arquivo HTML anterior
}

void tratarDadosSubmetidos () {
    String usuario = servidor.arg("usuario");
    String senha = servidor.arg("senha");
    // faz alguma coisa com esses dados
    // redireciona para uma outra página na rota "/"
    servidor.sendHeader("Location", "/");
    servidor.send(303);
}

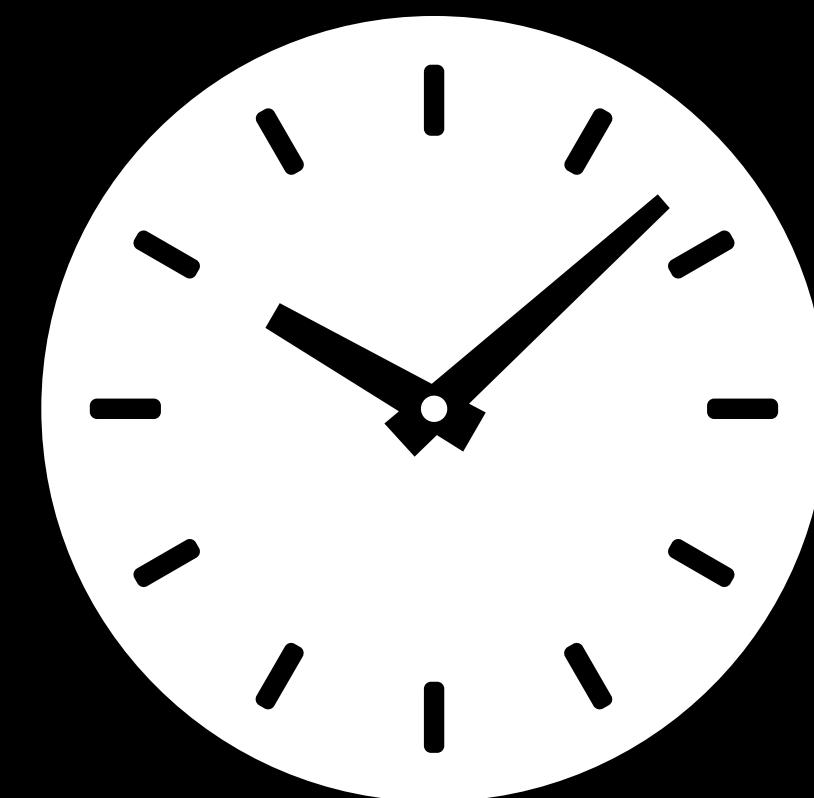
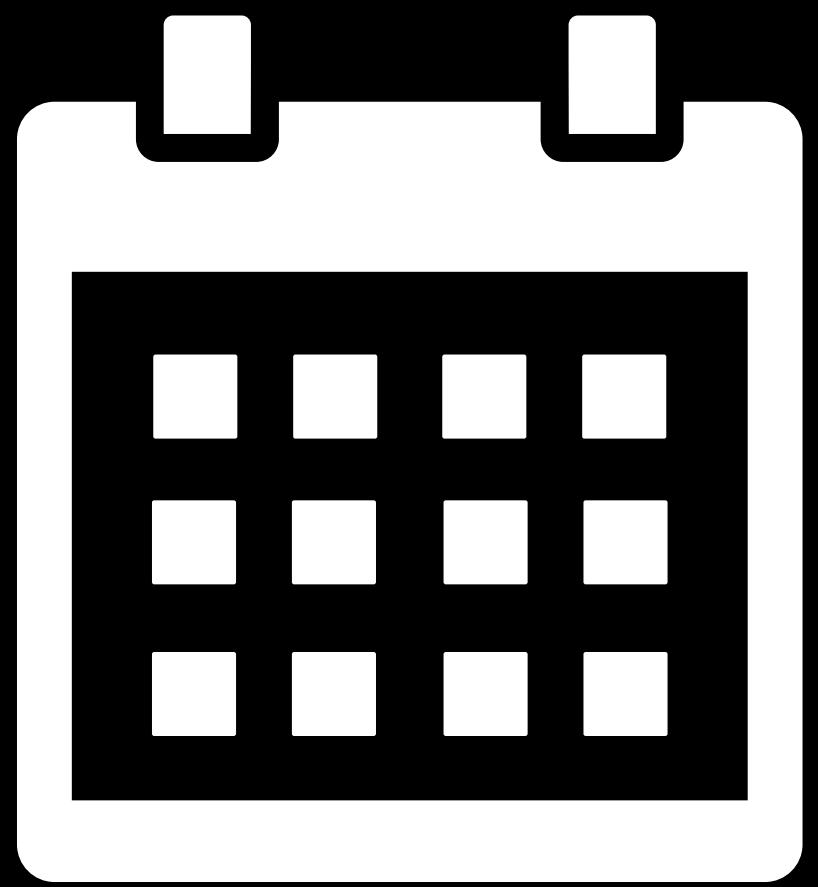
void setup () {
    // inicia WiFi, LittleFS...

    servidor.on("/formulario", HTTP_GET, paginaComArquivoHTML);
    servidor.on("/formulario", HTTP_POST, tratarDadosSubmetidos);
    servidor.begin();
}
```

Usuário

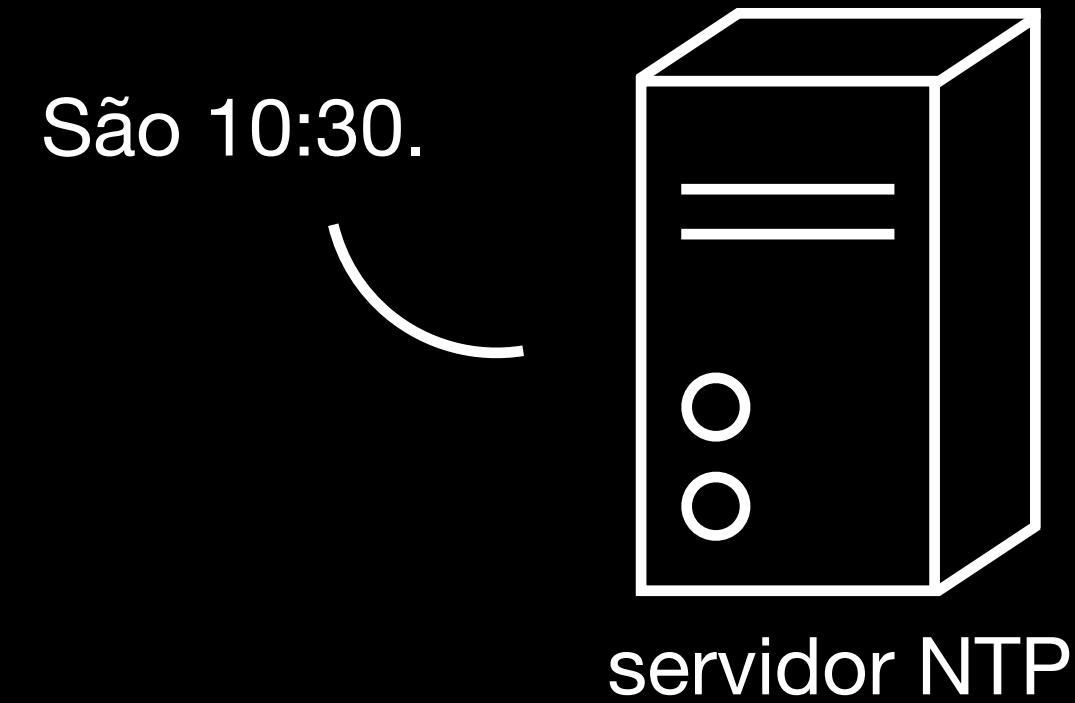
Senha

Entrar



Data / Hora no ESP32

ao começar o programa...



um tempo depois...



NTP (Network Time Protocol) para Sincronizar Data e Hora

```
#include <WiFi.h>
#include <time.h>

void setup() {
    Serial.begin(115200); delay(500);

    reconnectWiFi();

    // fuso horário do Brasil (GMT-3), servidor, servidor alternativo
    configTzTime("<-03>3", "a.ntp.br", "pool.ntp.org");
}

        // em alguma função do código...

struct tm tempo;
getLocalTime(&tempo);

Serial.println(&tempo, "%d/%m/%Y %H:%M:%S");
23/08/2024 16:30:01

char buffer[100];
strftime(buffer, sizeof(buffer), "%d/%m/%Y %H:%M:%S", &tempo);
String tempoString = String(buffer);
```

```
struct tm tempo;
getLocalTime(&tempo);

int hora = tempo.tm_hour;
int minuto = tempo.tm_min;
int segundo = tempo.tm_sec;
int dia = tempo.tm_mday;
int mes = tempo.tm_mon + 1;
int ano = tempo.tm_year + 1900;
int diaDaSemana = tempo.tm_wday; // 0 = domingo, 1 = segunda...
```



**https://  
(TLS)**

Protocolo HTTPS (HTTP + TLS)

Pai, preciso falar contigo...



Diga lá, filha!



Acho que tem um Deus nórdico  
bisbilhotando a gente...

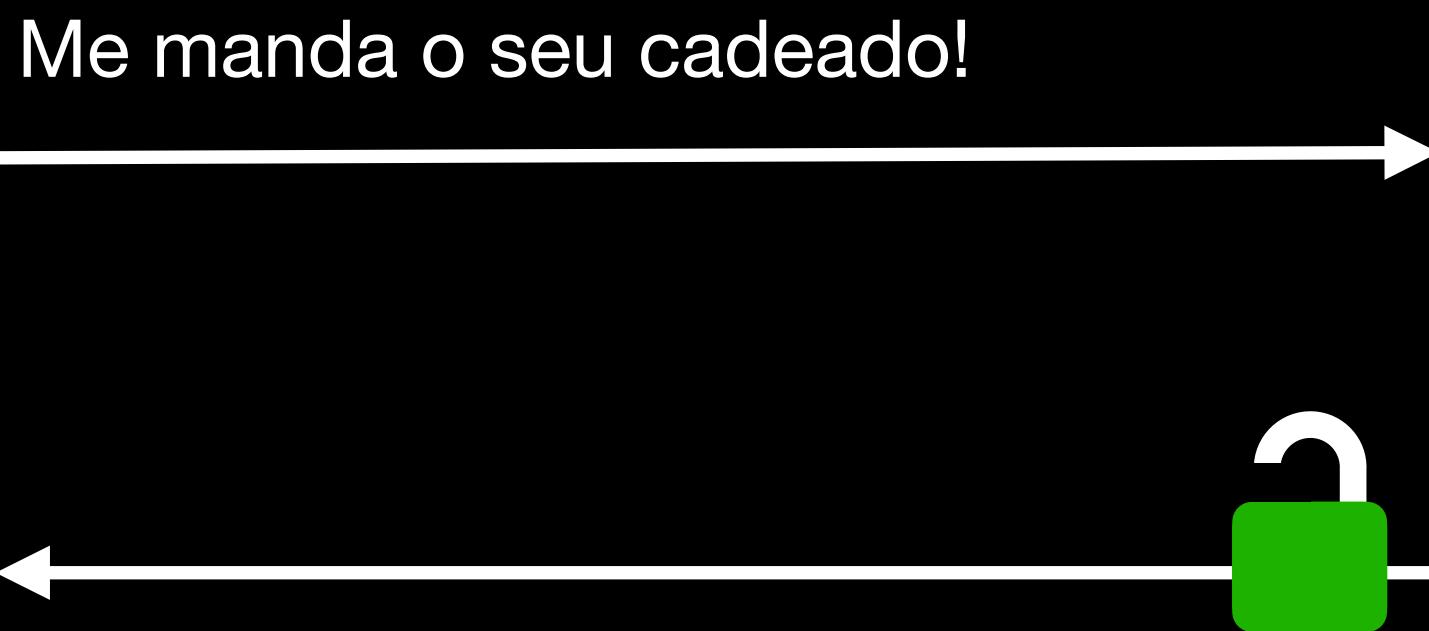


Confiança é para  
crianças e cães.

Comunicação entre Alice e Bob com Loki Bisbilhoteiro



**chave simétrica**  
(tranca e destranca dados)



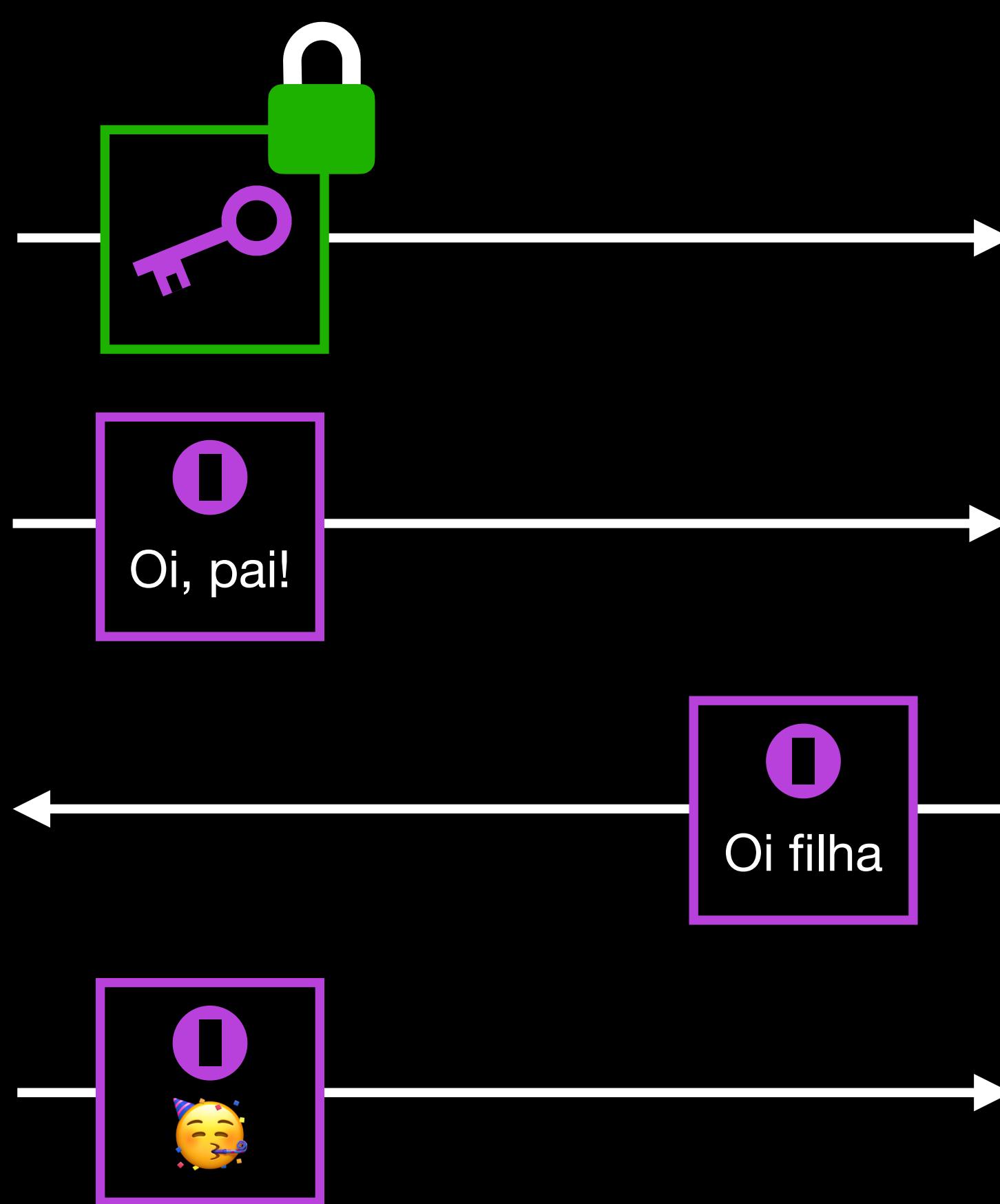
**chave pública**  
(tranca dados)



**chave privada**  
(destranca dados)



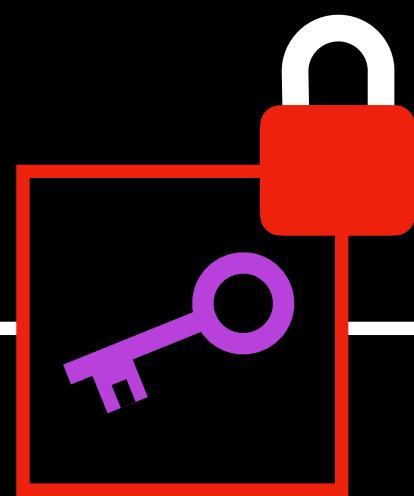
(cópia da chave da Alice)



Criptografia com Chaves Assimétricas e Simétricas



Me manda o  
seu cadeado!



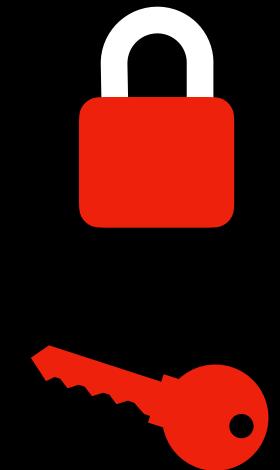
Oi, pai!



Senha?



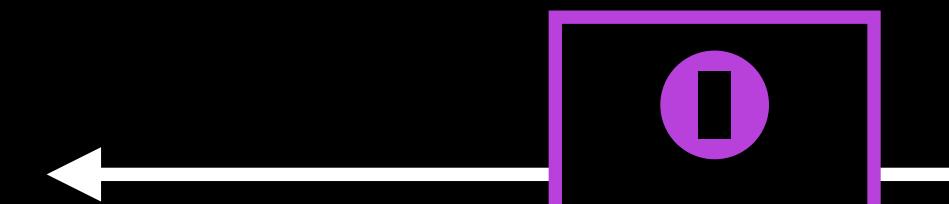
Kronos



Me manda o  
seu cadeado!



Pix?  
\$\$\$

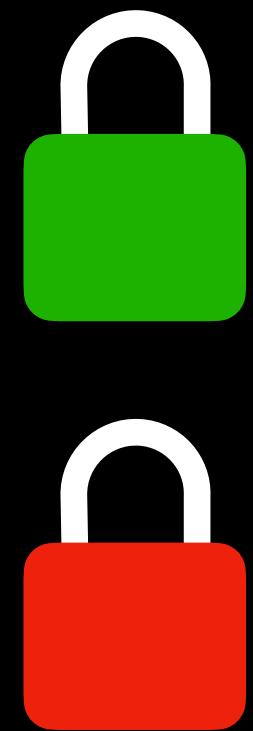


Valeu!



Ataque Intermediário (Man in the Middle)

Como eu sei se o  
cadeado veio do Bob  
ou de outra pessoa?



Vou te mandar  
um certificado de  
autenticidade!



Problema de Confiança na Chave Pública

Não sei se essa é a assinatura mesmo do Bob ou não...



Não confio em político.



Em Deus eu confio e não temerei!



Autoridades de Certificação Raiz

*Jesus  
Alá  
Zeus*

Oi, Alice!

Só para certificar: sou eu mesmo mandando esse cadeado.

Rio de Janeiro, 23 de agosto de 2024  
Bob

**Certifico que a assinatura "Bob" é válida e confiável.**

**Brasília, 1º de janeiro de 2023**  
**Presidente**

*Certifico que a assinatura "Presidente" é válida e confiável.*

*Céu, 1º de janeiro de 2002*  
**Jesus**

Cadeia de Certificados

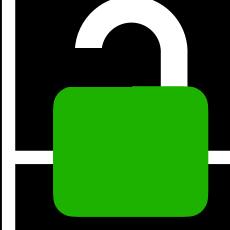


É o Bob mesmo,  
posso continuar!



Me manda o seu cadeado!

Cadeado ok!  
Bob  
Presidente  
Jesus



Tem boi na linha.  
Deu ruim. Parei!

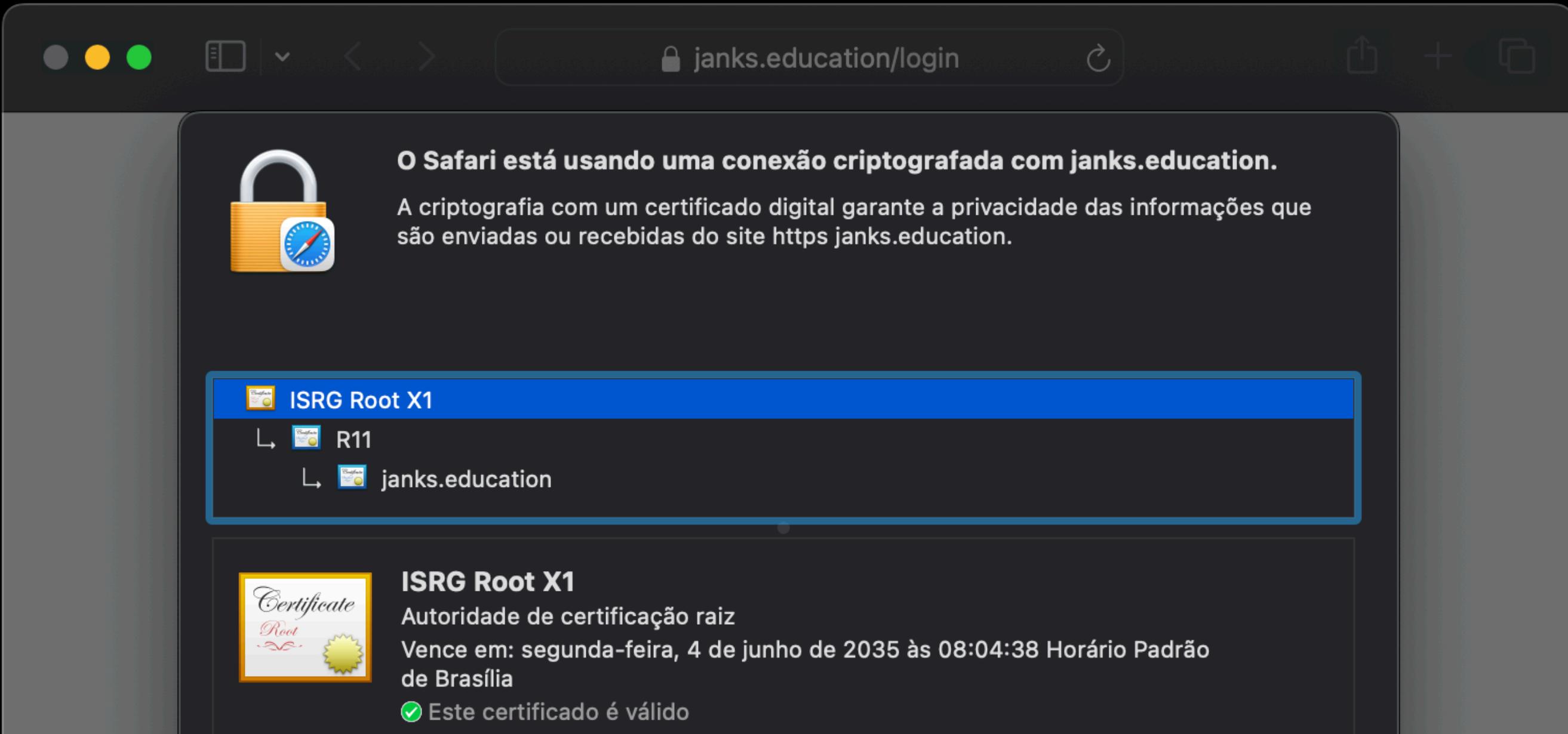


Me manda o seu cadeado!

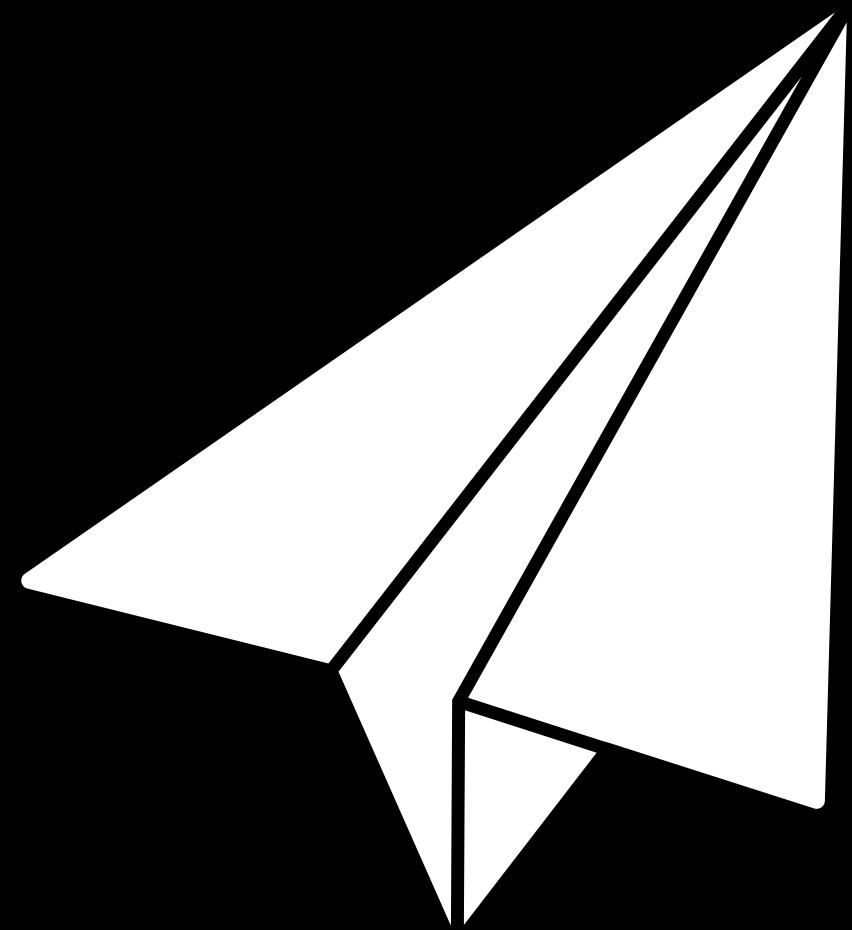
Cadeado ok!  
Bob  
É verdade  
esse bileté.



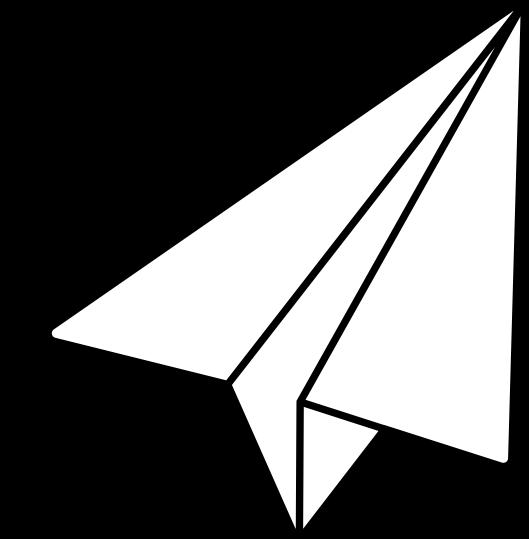
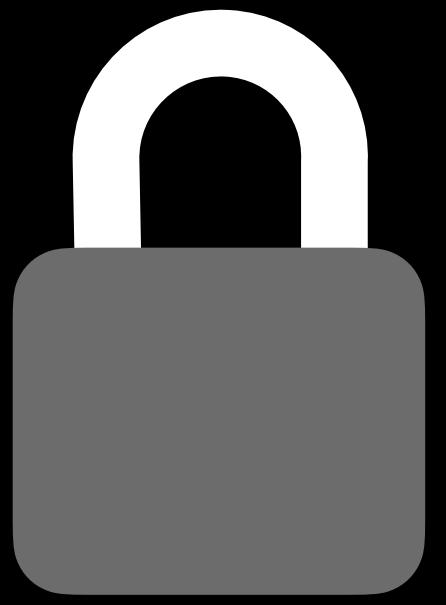
Envio da Chave Pública com Certificado



Exemplos de Certificados HTTPS nos Navegadores



Requisição Web no ESP32



1. WiFi

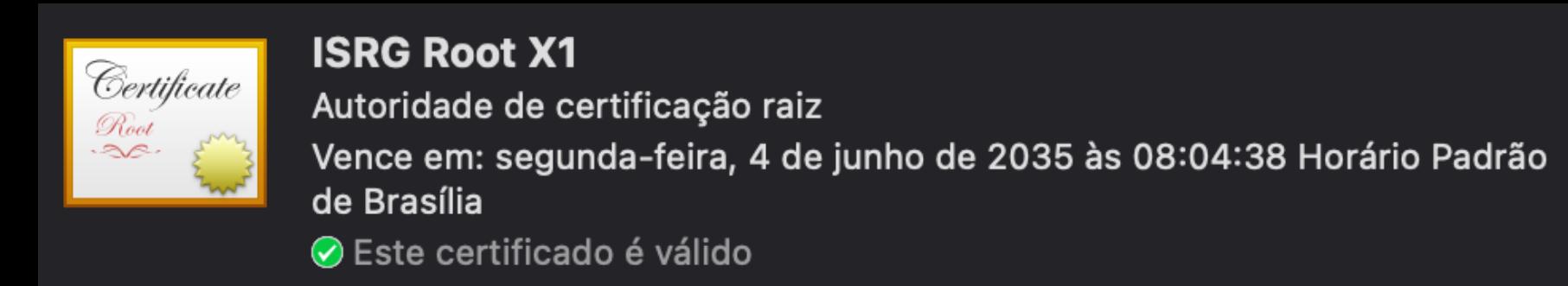
2. Certificado

3. Requisição

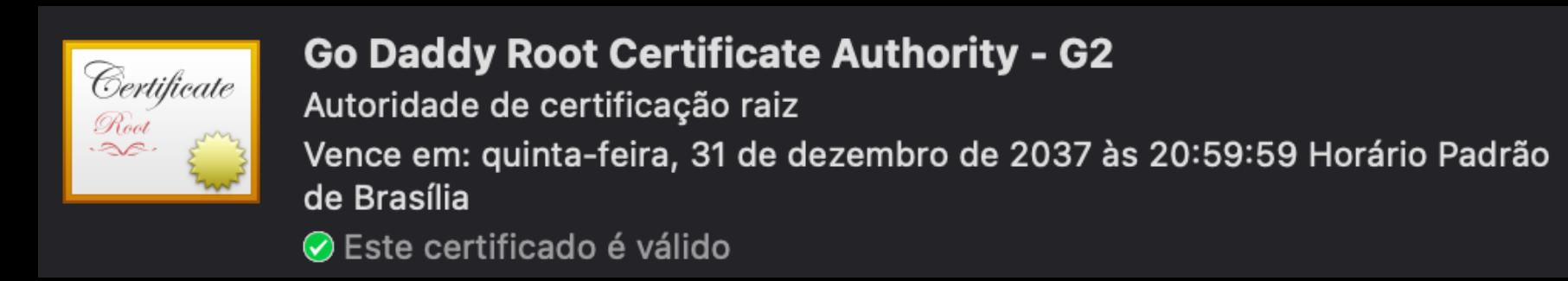
3 Etapas para uma Requisição Web

# certificados.h

```
const char certificado1[] = R"=EOF=(  
-----BEGIN CERTIFICATE-----  
MIIFazCCA10gAwIBAgIRAIQz7DSQ0NZRGPgu20CiwAwDQYJKoZIhvcNAQELBQA  
TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmlFNlY3VyaXR5IFJlc2Vh  
cmNoIEdyb3VwMRUwEwYDVQQDEwJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4  
WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu  
...  
-----END CERTIFICATE-----  
)=EOF=";
```



```
const char certificado2[] = R"=EOF=(  
-----BEGIN CERTIFICATE-----  
MIIDxTCCAq2gAwIBAgIBADANBgkqhkiG9w0BAQsFADCBgzELMAkGA1UEBhMCVVMx  
EDA0BgNVBAgTB0FyaXpvbmExEzARBgNVBAcTC1Njb3R0c2RhGUxGjAYBgNVBAoT  
EUdvRGFkZHkuY29tLCBJbmMuMTEwLwYDVQQDEyhHbyBEYWRkeSBSb290IENlcnRp  
...  
4uJEvlz36hz1  
-----END CERTIFICATE-----  
)=EOF=";
```



```
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include "certificados.h"

String chaveTelegram = "COLOQUE A SUA CHAVE DO BOTFATHER AQUI!";
String idDoChat = "COLOQUE O ID DA SUA CONVERSA AQUI!";
String enderecoBase = "https://api.telegram.org/bot" + chaveTelegram;

WiFiClientSecure conexaoSegura;

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

    conexaoSegura.setCACert(certificado1);
    conexaoSegura.setCACert(certificado2);
}

// código continua a seguir...
```

```
// continuação do código anterior...

void enviarMensagemTelegram(String mensagem) {
    JsonDocument dados;
    dados["chat_id"] = idDoChat;
    dados["text"] = mensagem;
    String dadosString;
    serializeJson(dados, dadosString);

    String enderecoMensagemTexto = enderecoBase + "/sendMessage";
    HTTPClient requisicao;
    requisicao.begin(conexaoSegura, enderecoMensagemTexto);
    requisicao.addHeader("Content-Type", "application/json");
    int codigoDoResultado = requisicao.POST(dadosString);
    String resposta = requisicao.getString();
    Serial.println(resposta);

    if (codigoDoResultado != 200) {
        Serial.println("Erro ao enviar mensagem!");
    }
}
```

Requisição de Envio de Mensagem para o Telegram