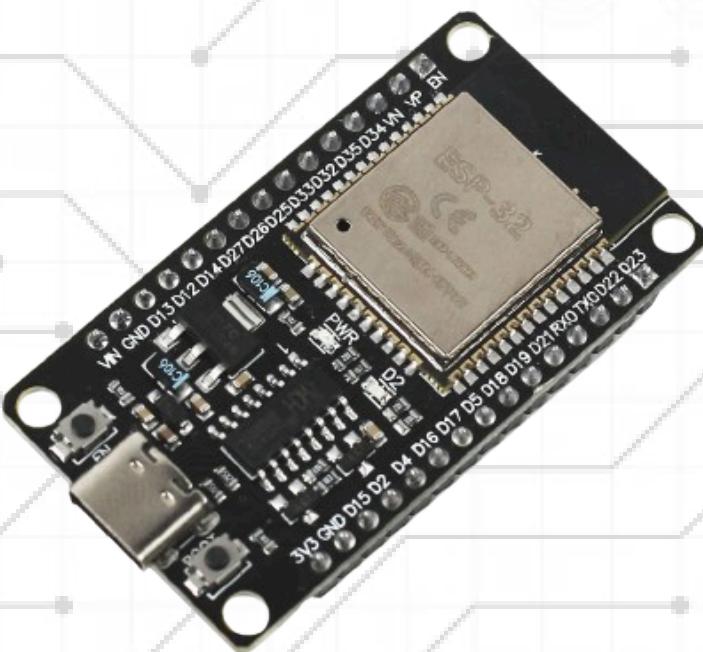
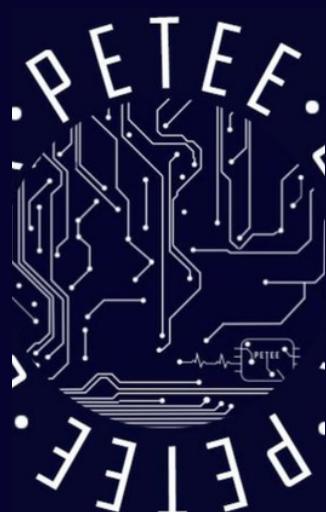


# CONTROLE NA PALMA DA MÃO: OFICINA DE CONTROLE DE DISPOSITIVOS ELETRÔNICOS VIA SMARTPHONE

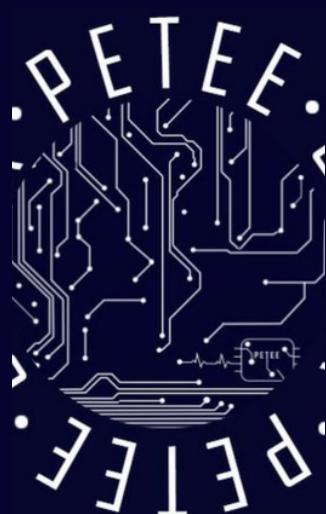


**MIT**  
APP INVENTOR

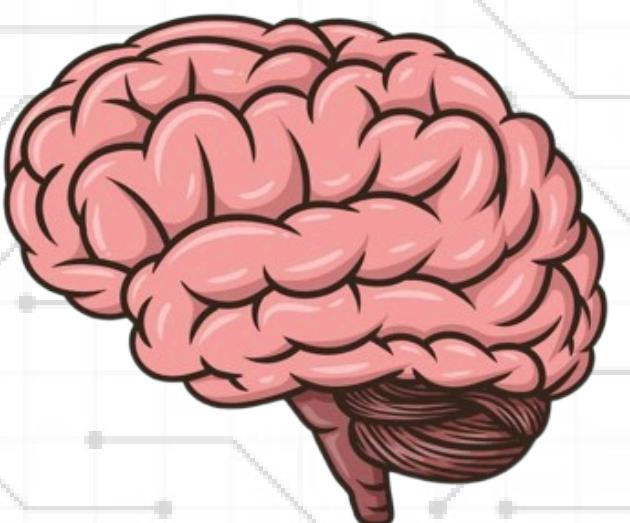


# NESSA AULA VAMOS SER CAPAZES DE:

- CRIAR SEU PRIMEIRO  
APLICATIVO ANDROID
- ACENDER UMA “LÂMPADA”  
PELO CELULAR



# ESP32 É COMO SE FOSSE O CÉREBRO DE TUDO



**MICROCONTROLADOR**

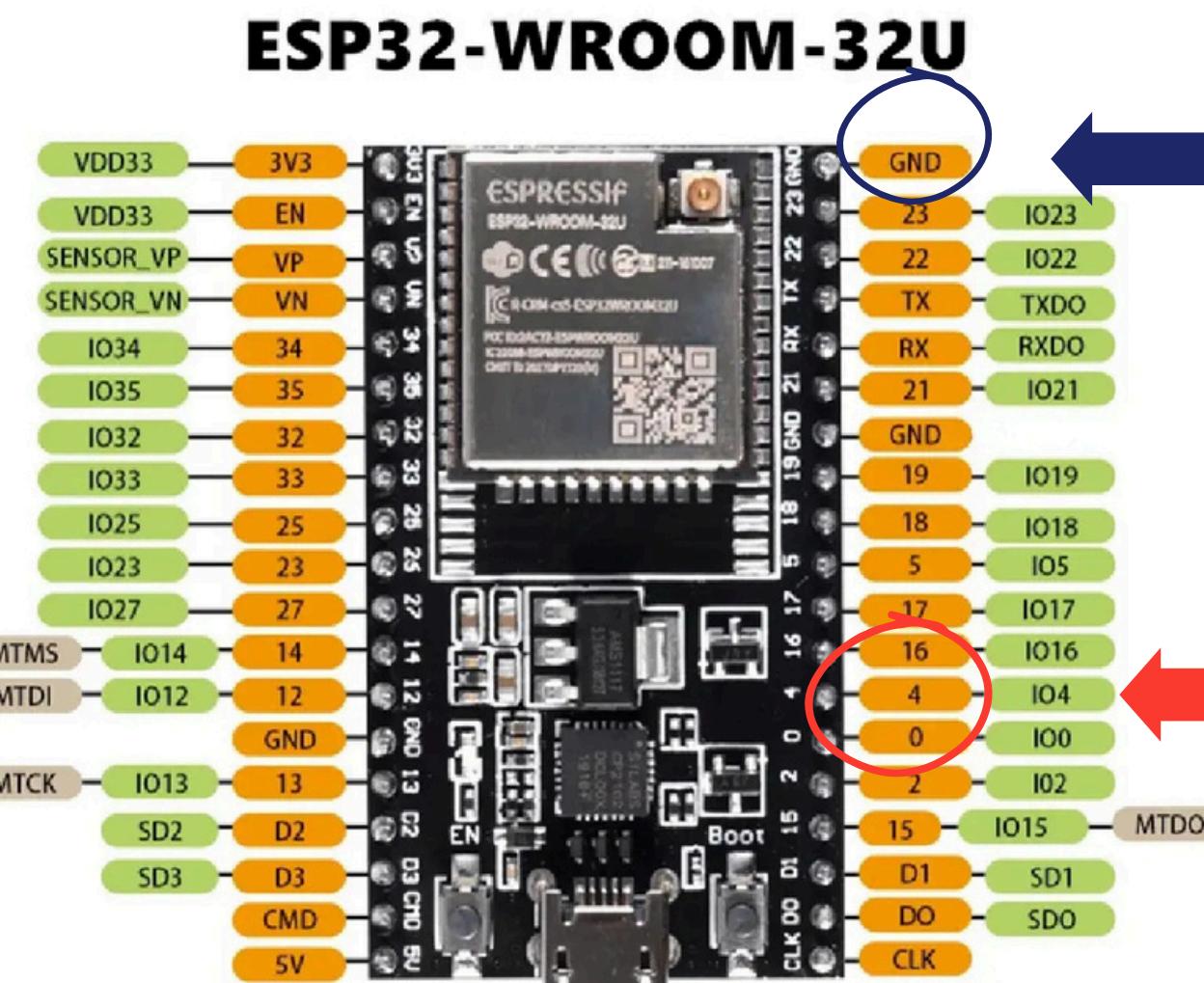
- RECEBE E REALIZA AS AÇÕES
- PODEMOS USAR O TEMPO E  
LAÇOS DE REPETIÇÃO

# QUAIS OS PINOS QUE DEVO CONECTAR



CUIDADO COM OS  
PINOS:

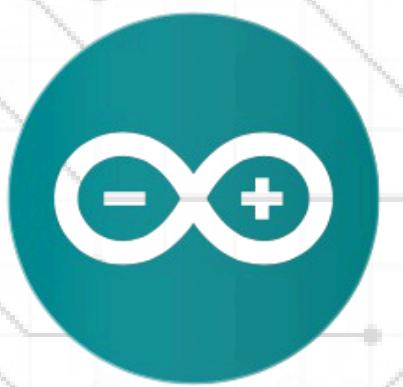
- VCC
- 5V
- 3.3V



GND → UM FIO  
DOS CANTOS DO  
BOTÃO

P4 → FIO DO  
MEIO DO BOTÃO

# ONDE PROGRAMAMOS O CÉREBRO



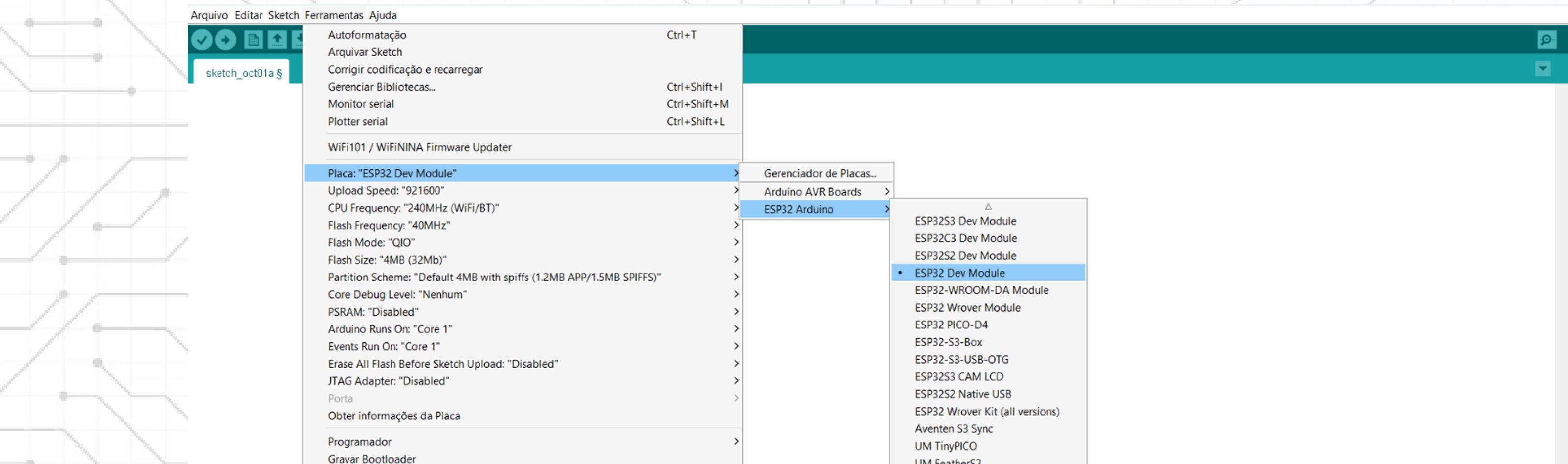
Arduino IDE

- **INSERIMOS OS CÓDIGOS**
- **CONECTAMOS A ESP32**
- **VISUALIZAÇÃO DOS DADOS**





# Arduino IDE



The screenshot shows the Arduino IDE interface. The top menu bar includes Arquivo, Editar, Sketch, Ferramentas, and Ajuda. A toolbar with icons for file operations is visible. The main window displays a sketch titled "sketch\_oct01a §". The Tools menu is open, showing various options like Autoformatação, Arquivar Sketch, and Gerenciar Bibliotecas... On the right, a submenu for "Placa: 'ESP32 Dev Module'" is expanded, listing details such as Upload Speed, CPU Frequency, Flash Frequency, and Partition Scheme. The "ESP32 Dev Module" option is also highlighted. A secondary dropdown menu for "Gerenciador de Placas..." is open under "ESP32 Arduino", listing numerous ESP32-based boards, with "ESP32 Dev Module" again being the selected item.

Arquivo Editar Sketch Ferramentas Ajuda

sketch\_oct01a §

Autoformatação  
Arquivar Sketch  
Corrigir codificação e recarregar  
Gerenciar Bibliotecas...  
Monitor serial  
Plotter serial  
WiFi101 / WiFiINA Firmware Updater

Placa: "ESP32 Dev Module"

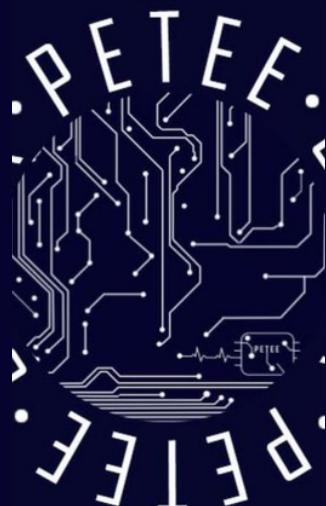
Upload Speed: "921600"  
CPU Frequency: "240MHz (WiFi/BT)"  
Flash Frequency: "40MHz"  
Flash Mode: "QIO"  
Flash Size: "4MB (32Mb)"  
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"  
Core Debug Level: "Nenhum"  
PSRAM: "Disabled"  
Arduino Runs On: "Core 1"  
Events Run On: "Core 1"  
Erase All Flash Before Sketch Upload: "Disabled"  
JTAG Adapter: "Disabled"  
Porta  
Obter informações da Placa

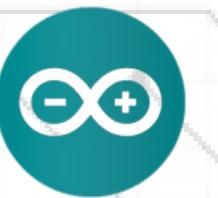
Programador  
Gravar Bootloader

Ctrl+T  
Ctrl+Shift+I  
Ctrl+Shift+M  
Ctrl+Shift+L

Gerenciador de Placas...  
Arduino AVR Boards  
ESP32 Arduino

ESP32S3 Dev Module  
ESP32C3 Dev Module  
ESP32S2 Dev Module  
• ESP32 Dev Module  
ESP32-WROOM-DA Module  
ESP32 Wrover Module  
ESP32 PICO-D4  
ESP32-S3-Box  
ESP32-S3-USB-OTG  
ESP32S3 CAM LCD  
ESP32S2 Native USB  
ESP32 Wrover Kit (all versions)  
Aventen S3 Sync  
UM TinyPICO  
UM FeatherS2  
UM FeatherS2 Neo  
UM TinyS2  
UM RMP  
UM NanoS3  
UM TinyS3  
UM PROS3  
UM FeatherS3  
S.ODI Ultra v1  
LilyGo T-Display  
LilyGo T-Display-S3  
microS2  
MagicBit





Arduino IDE

Arquivo Editar Sketch Ferramentas Ajuda



sketch\_oct01a §

Autoformatação  
Arquivar Sketch  
Corrigir codificação e recarregar  
Gerenciar Bibliotecas...  
Monitor serial  
Plotter serial  
WiFi101 / WiFiNINA Firmware Updater  
  
Placa: "ESP32 Dev Module"  
Upload Speed: "921600"  
CPU Frequency: "240MHz (WiFi/BT)"  
Flash Frequency: "40MHz"  
Flash Mode: "QIO"  
Flash Size: "4MB (32Mb)"  
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"  
Core Debug Level: "Nenhum"  
PSRAM: "Disabled"  
Arduino Runs On: "Core 1"  
Events Run On: "Core 1"  
Erase All Flash Before Sketch Upload: "Disabled"  
JTAG Adapter: "Disabled"  
**Porta**  
Obter informações da Placa  
  
Programador  
Gravar Bootloader

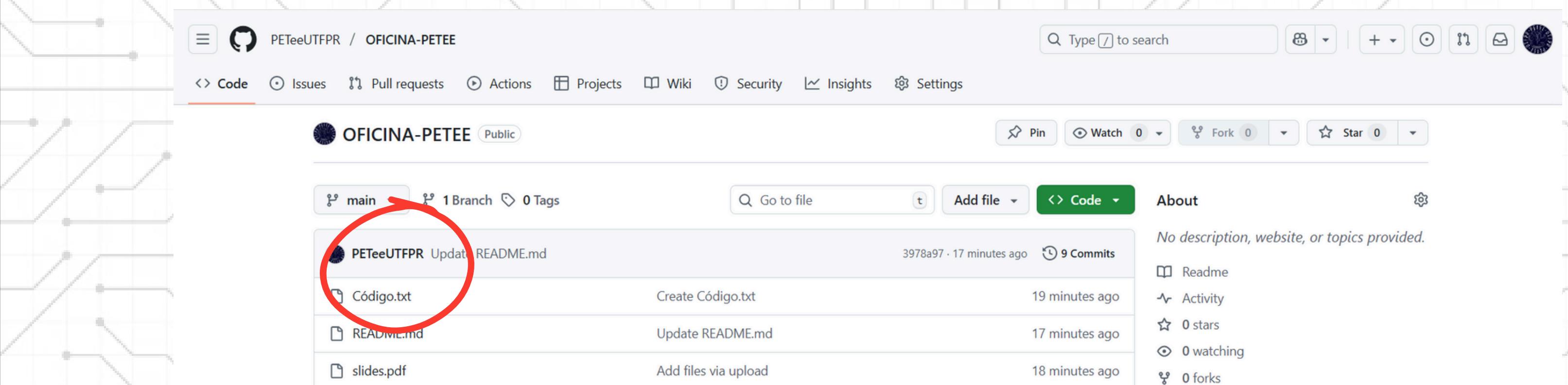
Ctrl+T

Ctrl+Shift+I  
Ctrl+Shift+M  
Ctrl+Shift+L

Portas seriais  
COM7

- PORTA PODE VARIAR DE NOME , EXEMPLO: COM7, COM13, COM8, ENTRE OUTRAS

# **ABRA A PASTA CÓDIGO.TXT E COPIE O CÓDIGO DENTRO PARA O ARDUINO**



# PODE USAR O BOTÃO “COPY RAW FILE” OU COPIAR TUDO COM O MOUSE

OFICINA-PETEE / Código.txt

PETEEUTPFR Create Código.txt

Code Blame 93 lines (71 loc) · 2.46 KB

```
1 #include "BluetoothSerial.h"
2
3 // Verifica se o Bluetooth está habilitado nas configurações do projeto
4 #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
5 #error Bluetooth ta desligado, provavelmente desabilitado na placa, ou no celular.
6 #endif
7
8 BluetoothSerial SerialBT;
9
10 // --- Mapeamento de Pinos ---
11 const int pinoLed = 2;      // Pino do LED embutido da ESP32
12 const int pinoSwitch = 4;    // Pino onde o switch será conectado
13
14 // --- Variáveis de Controle ---
15 String dadosRecebidos = "";
16
17 int estadoSwitch;
18 int ultimoEstadoSwitch = HIGH; // Pull-up mantém HIGH por padrão
```

1d170dd · 21 minutes ago History

Copy raw file

Raw



# MUDE “COLOCARNOME” POR OUTRO NOME QUALQUER POR EXEMPLO “BATMAN”

```
digitalWrite(pinoLed, 1);
delay(500);
digitalWrite(pinoLed, 0);
delay(500);
}

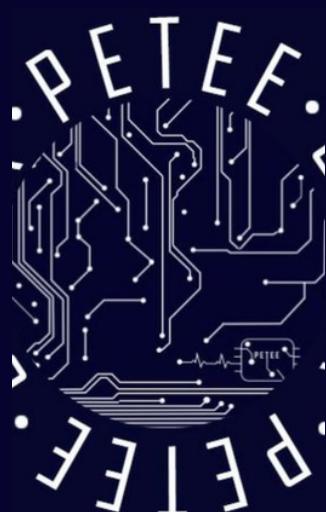
// --- Configuração Inicial ---
void setup() {
Serial.begin(15200);
SerialBT.begin("COLOCARNOME");

pinMode(pinoLed, OUTPUT);
pinMode(pinoSwitch, INPUT_PULLUP); // Usa pull-up interno para evitar flutuação

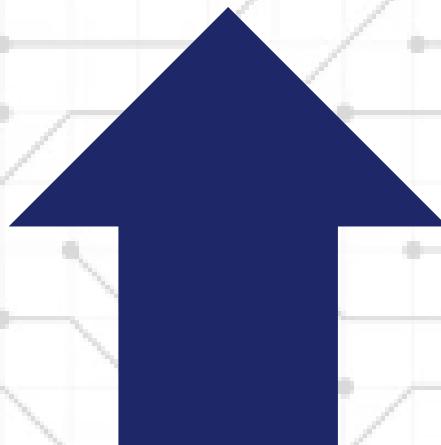
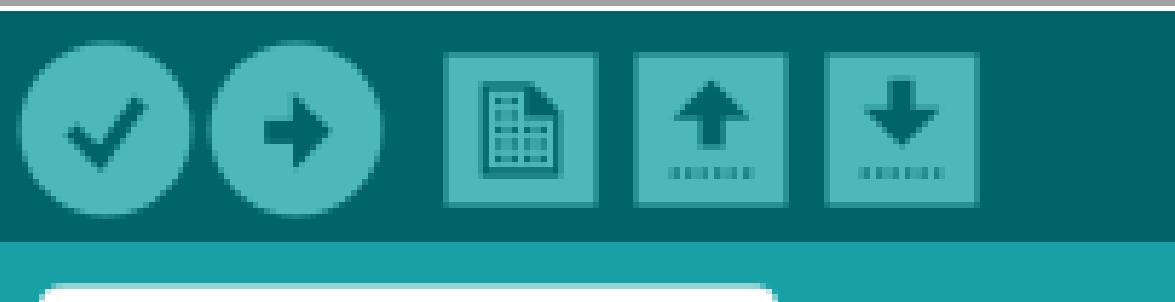
digitalWrite(pinoLed, LOW);

Serial.println("ESP32 pronto. Conecte ao Bluetooth 'ESP32_Botao_LED'.");
}

// --- Loop Principal ---
void loop() {
```



# COMPILAR E CARREGAR PARA A PLACA



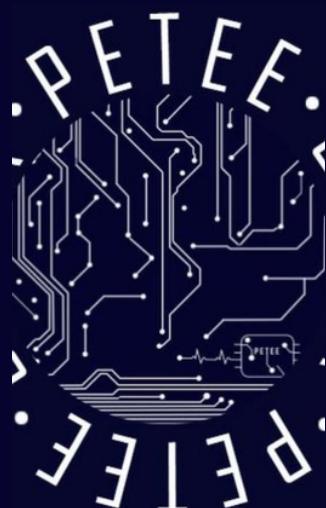
## COMPILAR

Carregado.

```
Writing at 0x000d65a2... (76 %)
Writing at 0x000dc02d... (79 %)
Writing at 0x000e19c4... (81 %)
Writing at 0x000e7ca2... (83 %)
Writing at 0x000f00e9... (86 %)
Writing at 0x000f87f4... (88 %)
Writing at 0x00101d60... (90 %)
Writing at 0x001071bb... (93 %)
Writing at 0x0010c93b... (95 %)
Writing at 0x00112487... (97 %)
Writing at 0x00117b21... (100 %)
Wrote 1094896 bytes (698202 compressed) at 0x00010000 in 9.7 seconds
Hash of data verified.
```

Leaving...

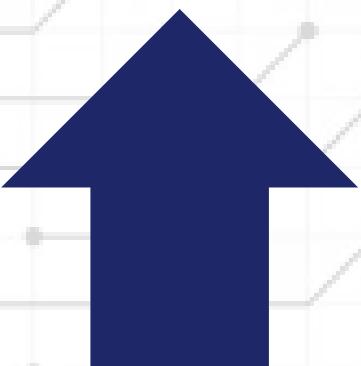
Hard resetting via RTS pin...



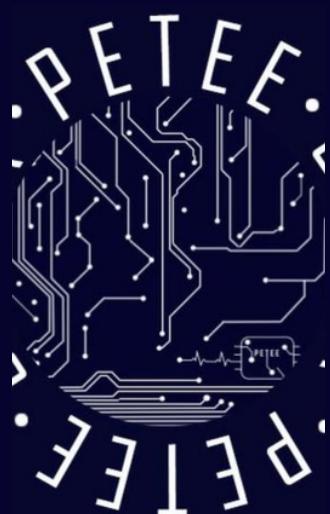
Carregado.

```
Writing at 0x000d65a2... (76 %)
Writing at 0x000dc02d... (79 %)
Writing at 0x000e19c4... (81 %)
Writing at 0x000e7ca2... (83 %)
Writing at 0x000f00e9... (86 %)
Writing at 0x000f87f4... (88 %)
Writing at 0x00101d60... (90 %)
Writing at 0x001071bb... (93 %)
Writing at 0x0010c93b... (95 %)
Writing at 0x00112487... (97 %)
Writing at 0x00117b21... (100 %)
Wrote 1094896 bytes (698202 compressed) at 0x00010000 in 9.7 seconds (effective 902.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```



**ESSA MENSAGEM INDICA QUE TIVEMOS ÉXITO  
PARA ENVIAR PRA PLACA**



```
#include "BluetoothSerial.h"

// Verifica se o Bluetooth está habilitado nas configurações do projeto
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth ta desligado, provavelmente desabilitado na placa, ou no celular.
#endif

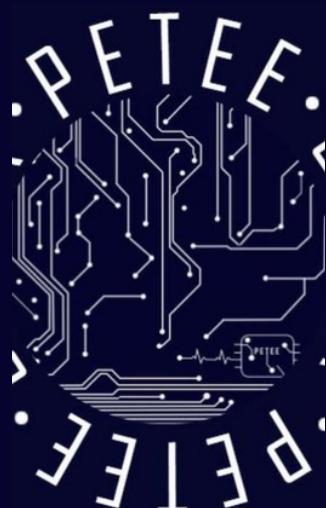
BluetoothSerial SerialBT;

// --- Mapeamento de Pinos ---
const int pinoLed = 2;          // Pino do LED embutido da ESP32
const int pinoSwitch = 4;        // Pino onde o switch será conectado

// --- Variáveis de Controle ---
String dadosRecebidos = "";

int estadoSwitch;
int ultimoEstadoSwitch = HIGH; // Pull-up mantém HIGH por padrão
unsigned long ultimoTempoDebounce = 0;
unsigned long delayDebounce = 50; // 50ms para considerar leitura estável

void piscaLED() {
    for(int i = 0; i< 10; i++){
        digitalWrite(pinoLed, 1);
        delay(500);
        digitalWrite(pinoLed, 0);
        delay(500);
    }
}
```



```
// --- Configuração Inicial ---
void setup() {
    Serial.begin(115200);
    SerialBT.begin("ESPDOPOTEE");

    pinMode(pinoLed, OUTPUT);
    pinMode(pinoSwitch, INPUT_PULLUP); // Usa pull-up interno para evitar flutuação

    digitalWrite(pinoLed, LOW);

    Serial.println("ESP32 pronto. Conecte ao Bluetooth 'ESP32_Botao_LED'.");
}

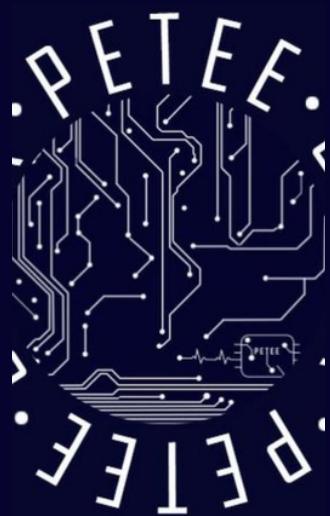
// --- Loop Principal ---
void loop() {

    // --- Leitura do switch ---
    int leituraAtual = digitalRead(pinoSwitch);

    // Debounce
    if (leituraAtual != ultimoEstadoSwitch) {
        ultimoTempoDebounce = millis();
    }

    if ((millis() - ultimoTempoDebounce) > delayDebounce) {
        if (leituraAtual != estadoSwitch) {
            estadoSwitch = leituraAtual;

            // Apenas reage quando o switch estiver na posição "ativa" (conectado a GND → LOW)
            if (estadoSwitch == LOW) {
                Serial.println("Switch na posição ativa!");
                //piscarLed();
                SerialBT.println("BOTAO_PRESSIONADO"); // envia pro App Inventor
            }
            else {
                Serial.println("Switch na posição inativa.");
                SerialBT.println("BOTAO_SOLTO"); // envia estado inativo
            }
        }
    }
}
```



```
// ---- Verificação de comandos via Bluetooth ----  
  
if (SerialBT.available()) {  
    dadosRecebidos = SerialBT.readStringUntil('\n');  
    dadosRecebidos.trim();  
  
    Serial.print("Comando recebido via Bluetooth: ");  
    Serial.println(dadosRecebidos);  
  
    if (dadosRecebidos == "1") {  
        Serial.println("Acende LED");  
        acenderLED();  
        SerialBT.println("LED aceso");  
    }  
    if (dadosRecebidos == "0") {  
        Serial.println("Apaga LED");  
        apagarLED();  
        SerialBT.println("LED apagado");  
    }  
    delay(20); // Delay do watchdog  
}
```

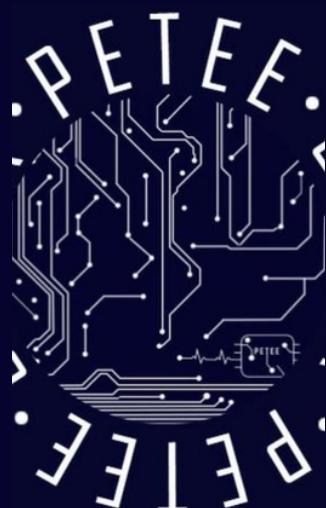
# MIT APP INVENTOR

O MIT APP INVENTOR É UMA PLATAFORMA DE DESENVOLVIMENTO QUE PERMITE CRIAR APLICATIVOS PARA DISPOSITIVOS ANDROID DE FORMA VISUAL E INTUITIVA, UTILIZANDO BLOCOS DE PROGRAMAÇÃO. CRIADO PELO MIT (JURA?), O AMBIENTE É VOLTADO TANTO PARA INICIANTES QUANTO PARA DESENVOLVEDORES EXPERIENTES, FACILITANDO A CRIAÇÃO DE APPS SEM A NECESSIDADE DE CÓDIGO EXTENSIVO. ELE PROMOVE O APRENDIZADO DE PROGRAMAÇÃO E O DESENVOLVIMENTO DE HABILIDADES COMPUTACIONAIS. ALÉM DISSO, É AMPLAMENTE USADO EM PROJETOS EDUCACIONAIS E PROTÓTIPOS RÁPIDOS DE APLICATIVOS.



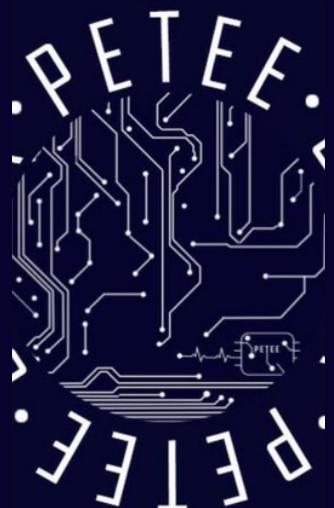
# O que é um aplicativo?

**Um aplicativo é um software destinado a realizar tarefas específicas para o usuário final, como por exemplo edição de textos ou navegação na web. Ele depende de um sistema operacional para funcionar e interage com o hardware de forma indireta. Diferencia-se do software de sistema, que gerencia os recursos do dispositivo e oferece a plataforma para execução dos aplicativos.**

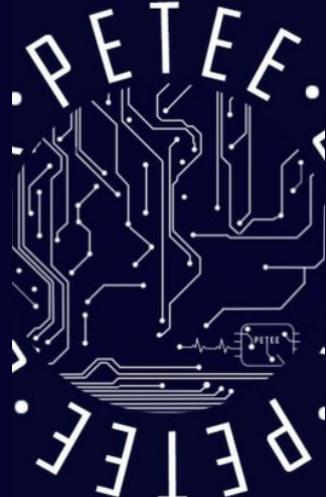


# Componentes básicos de um aplicativo

- Activities
- UI components
- Fragments
- Intents
- Services
- Broadcast Receivers
- Content providers



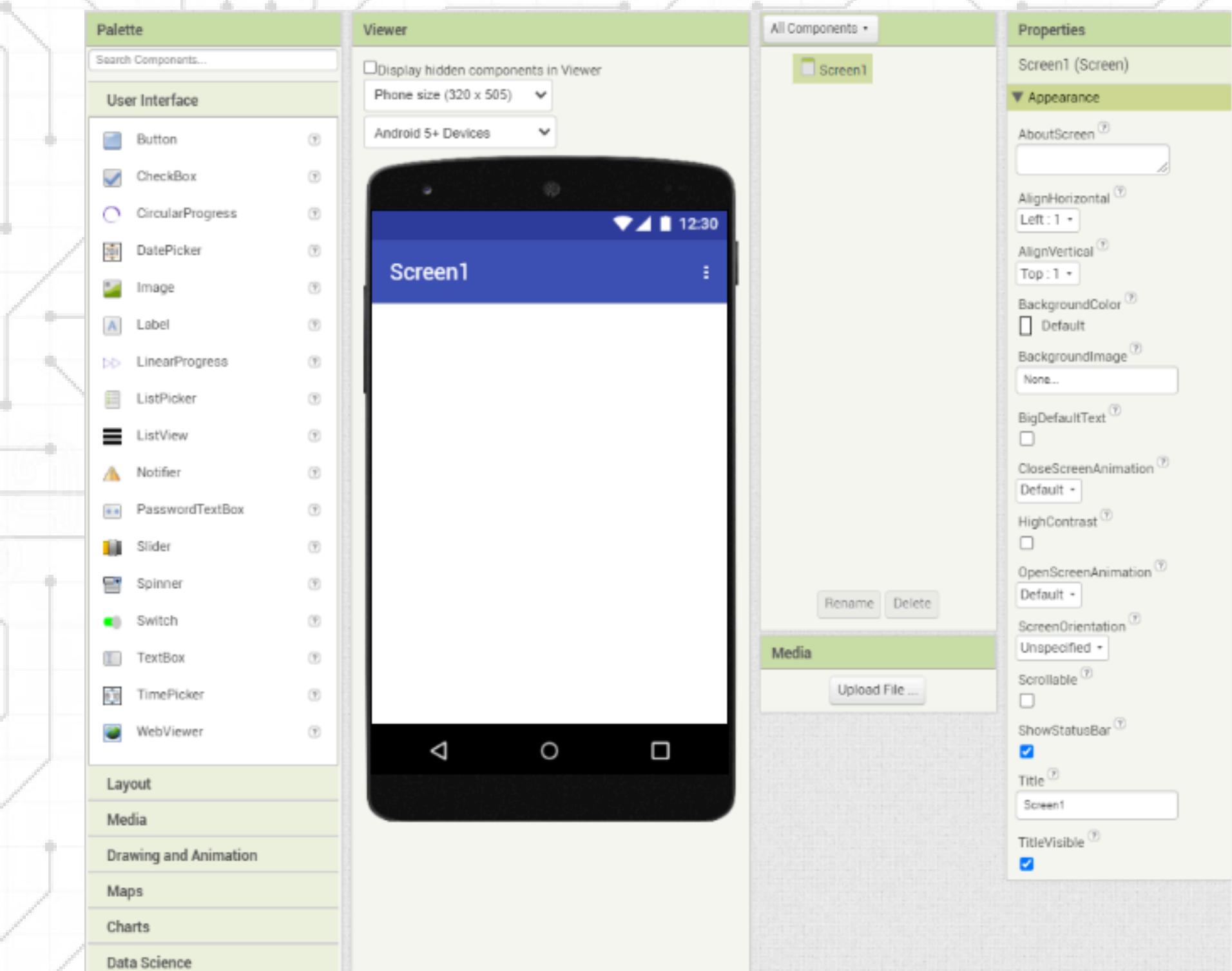
# INTERFACE DE DESENVOLVIMENTO DO M.A.I.



# Designer

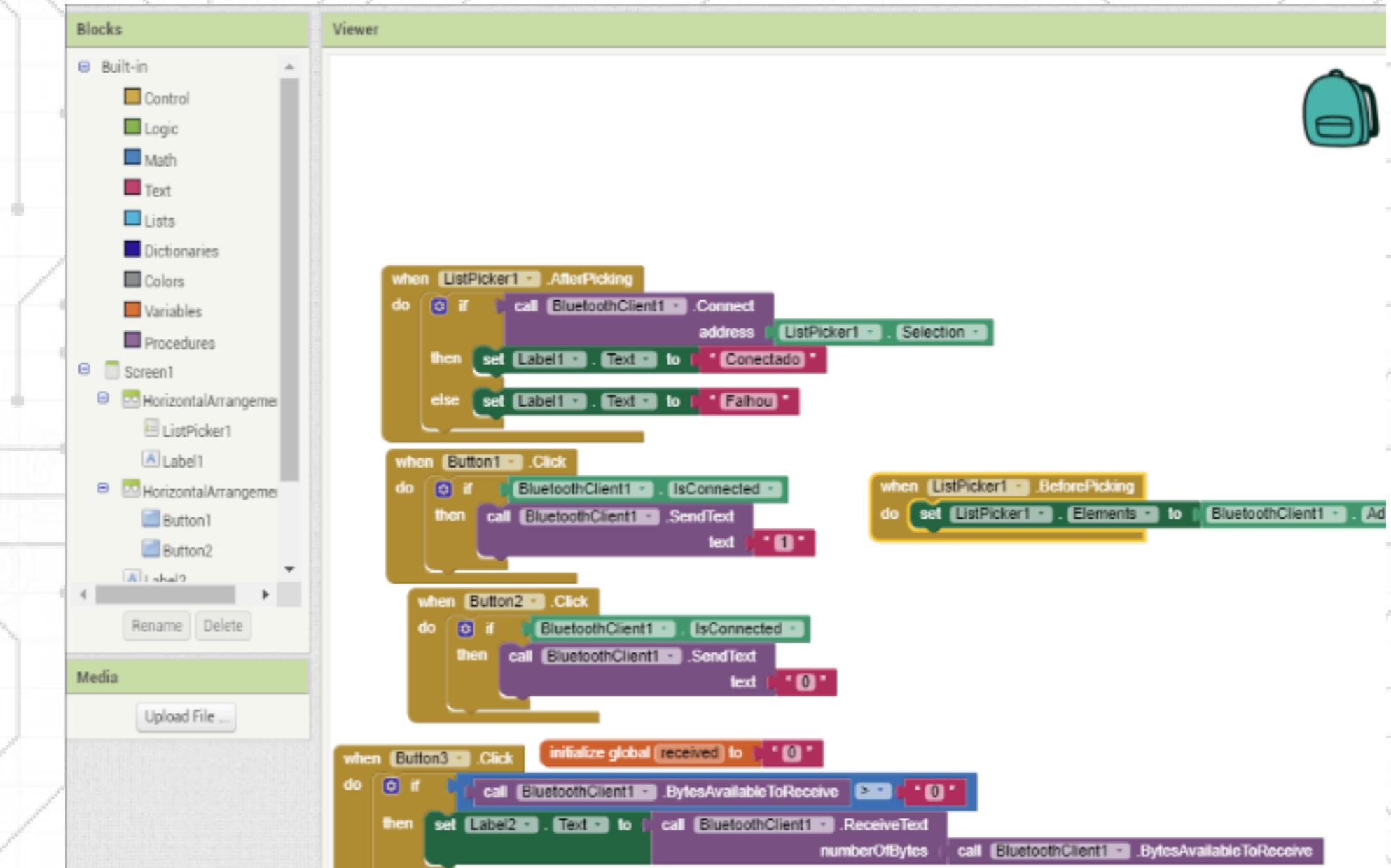
**Essa tela é o espaço onde todos os componentes do aplicativo serão adicionados.**

**Os componentes da interface gráfica são posicionados conforme o layout desejado, enquanto os componentes invisíveis precisam apenas ser arrastados para a tela para que possam ser utilizados posteriormente.**



# Blocks

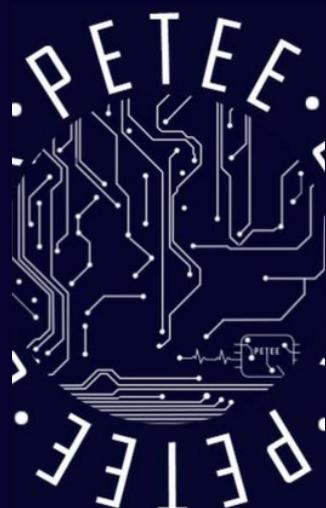
A tela no qual será desenvolvida a lógica por trás das activities. Para se desenvolver isso, arrasta-se blocos prontos de lógica de programação para desenvolver o funcionamento do app.





A dense grid of light gray lines forming a circuit board pattern serves as the background for the entire slide.

**Para o desenvolvimento na  
prática.**



# Interface de Usuário

**Na interface do usuário, primeiro são definidos os arranjos, pois é necessário organizá-los previamente para posicionar os componentes lado a lado.**

**Em seguida, adicionam-se os demais componentes e ajustam-se seus nomes e conteúdos.**

**Por fim, é incluído o Bluetooth Client, que não é parte da interface, mas é incluído na mesma tela**



# Porque o Bluetooth client, e não o servidor?

O **BluetoothClient** é usado quando o seu dispositivo quer se conectar a outro aparelho.

O **BluetoothServer** é usado quando o seu dispositivo vai esperar que outro aparelho se conecte a ele.

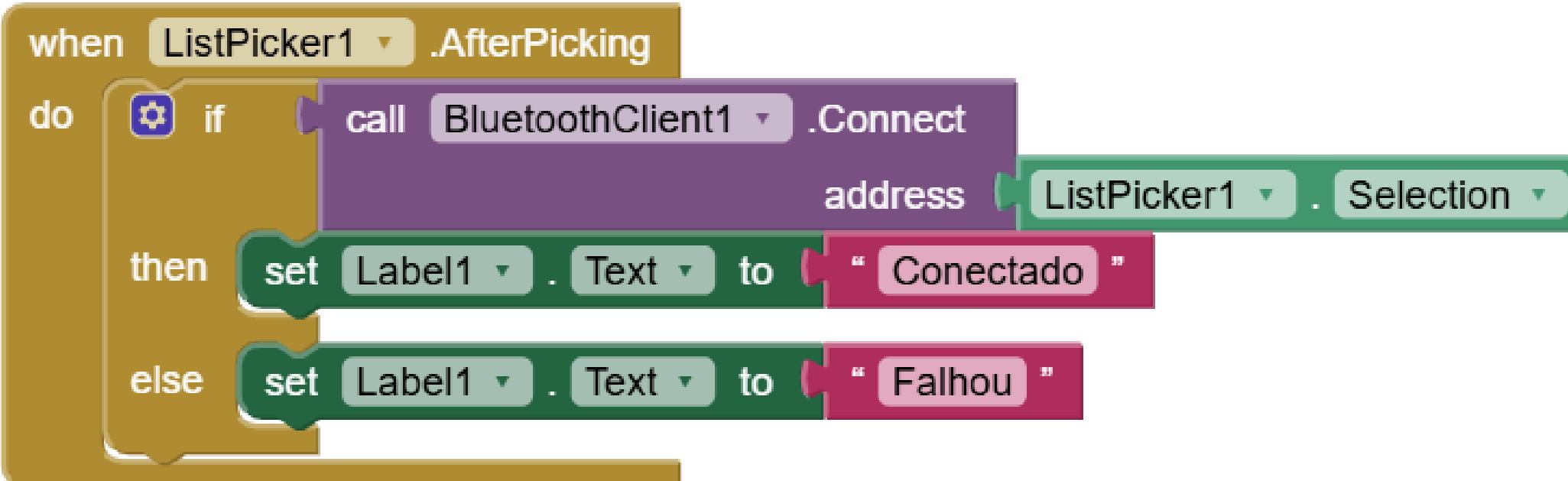
# BLOCOS

```
when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```

**Esse conjunto de blocos declara que, na inicialização do aplicativo, os dados da lista de seleção serão a lista de dispositivos conectados via bluetooth.**

**Passo 1**

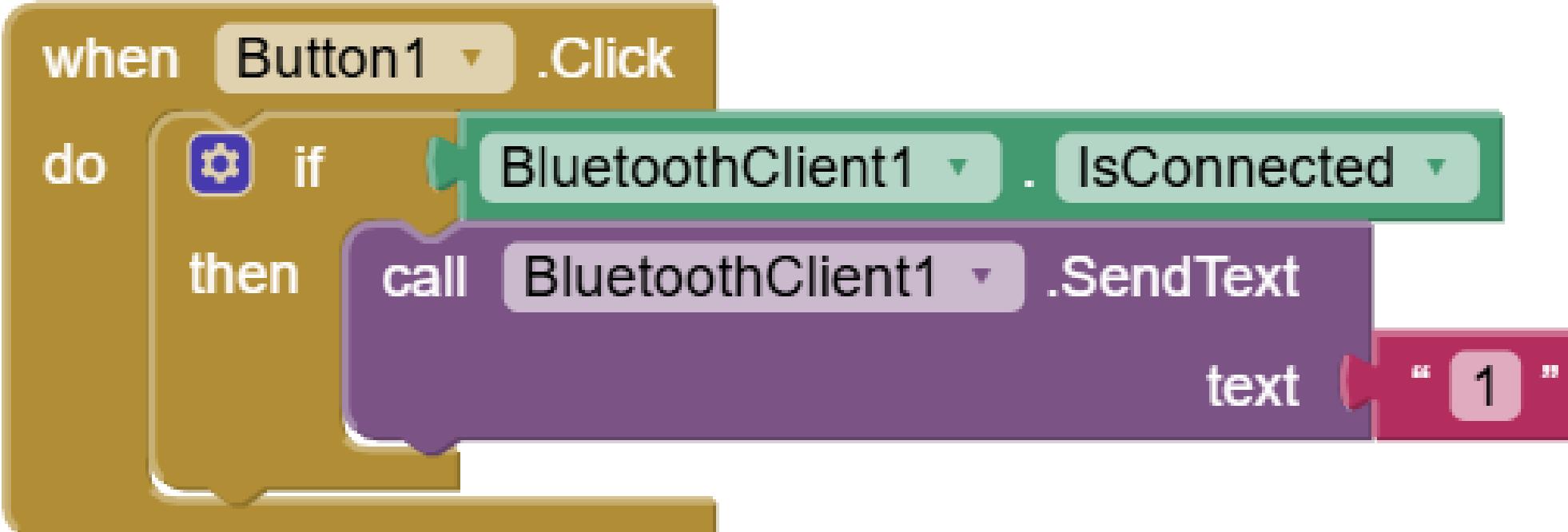
# BLOCOS



**Esse conjunto de blocos declara que ao se selecionar um dos dados da lista de seleção, o aplicativo vai se conectar com o dispositivo que possuí o endereço selecionado, se der certo, a label1 mostrará escrito conectado.**

**Passo 2**

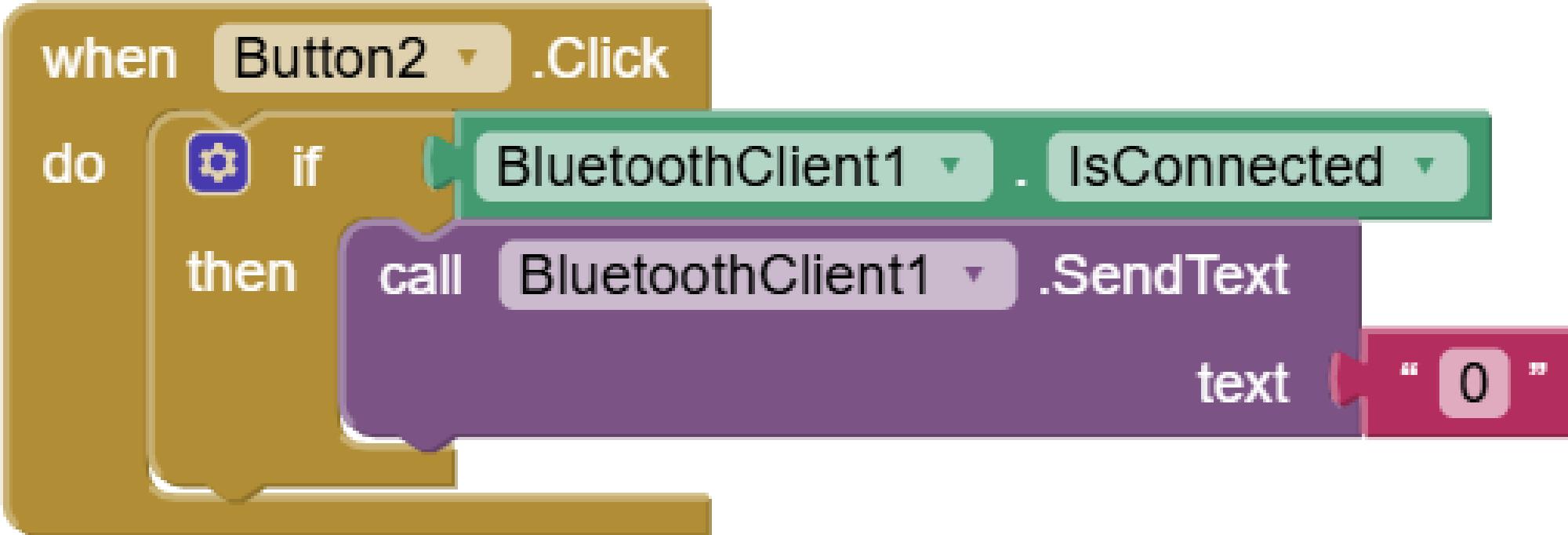
# BLOCOS



**Esse conjunto de blocos declara que ao clicar no botão, se um dispositivo estiver conectado, ele envia o texto 1.**

**Passo 3**

# BLOCOS

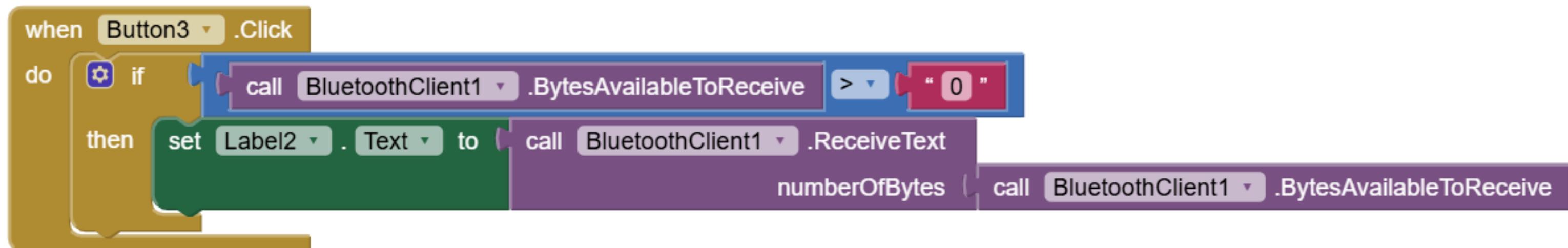


**Esse conjunto de blocos declara que ao clicar no botão, se um dispositivo estiver conectado, ele envia o texto 0.**

**Passo 4**

# BLOCOS

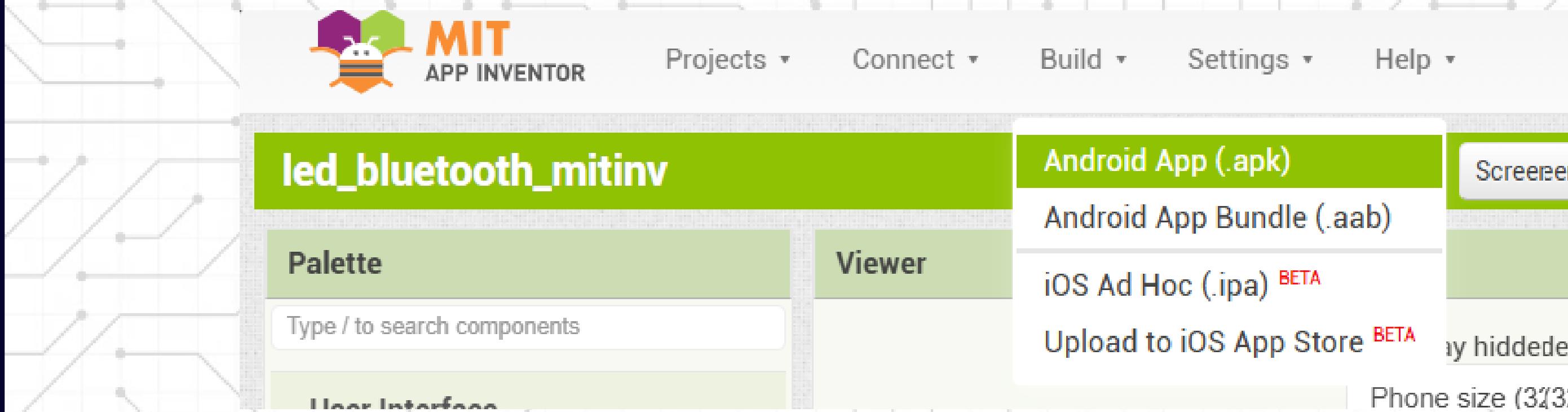
O que pode ser melhorado nesse bloco?



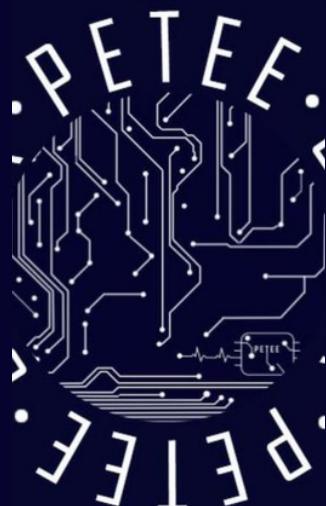
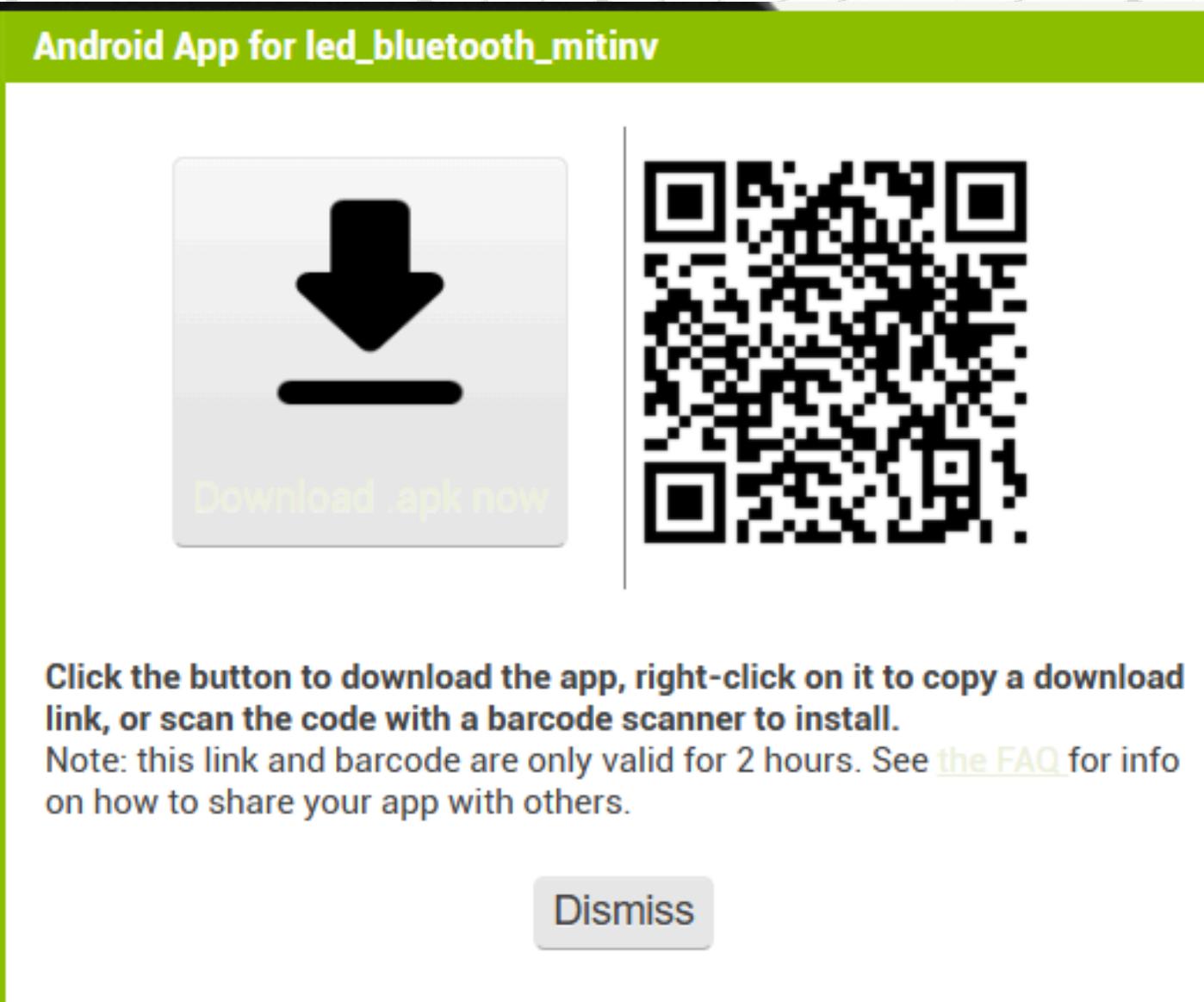
Esse conjunto de blocos declara que ao clicar no botão, se existirem bytes no buffer de recebimento ele lê esses bytes e escreve na label2.

**Passo 5**

# Como gerar o aplicativo

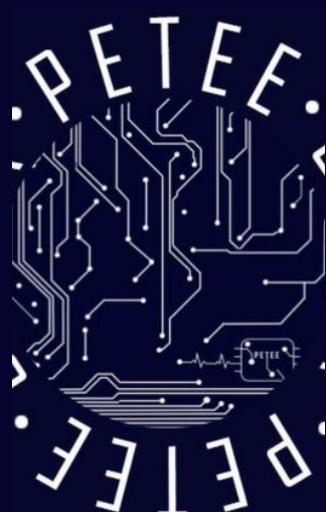


# Como gerar o aplicativo



# Atividade em Grupo

- Implementem mais um botão no aplicativo para enviar mais um dado e façam a função PiscaLED ser chamada com essa mensagem.



# Atividade em Grupo

- Implementem a passagem de mensagens via TextBox no MIT app inventor

