

Documentação Completa - Controle LED via Bluetooth

ESP32 + MIT App Inventor

PARTE 1: CÓDIGO ESP32 (ARDUINO IDE)



Componentes Necessários

- **ESP32** (qualquer modelo)
- **LED** (qualquer cor)
- **Resistor** 220Ω a 330Ω
- **Protoboard**
- **Jumpers**
- **Cabo USB** para programação



Montagem do Circuito

Opção 1: LED Externo

ESP32 GPIO2 → Resistor 220Ω → LED (ânodo +)

LED (cátodo -) → GND do ESP32

Opção 2: LED Interno (mais simples)

- Use apenas o LED interno do ESP32 no GPIO2
- Não precisa de componentes externos



Código ESP32 (Arduino IDE)

```
#include "BluetoothSerial.h"
```

```
// Verificar se o Bluetooth está habilitado
```

```
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
```

```
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
```

```
#endif
```

```
// Criar objeto Bluetooth Serial
```

```
BluetoothSerial SerialBT;
```

```
// Definir pino do LED

const int LED_PIN = 2; // GPIO2 (LED interno do ESP32)


// Variáveis de controle

String receivedMessage = "";

bool ledState = false;


void setup() {

    // Inicializar Serial Monitor

    Serial.begin(115200);


    // Configurar pino do LED como saída

    pinMode(LED_PIN, OUTPUT);

    digitalWrite(LED_PIN, LOW); // LED inicialmente desligado


    // Inicializar Bluetooth Serial

    SerialBT.begin("ESP32_LED_Control"); // Nome do dispositivo Bluetooth

    Serial.println("O dispositivo está pronto para pareamento!");

    Serial.println("Nome do dispositivo: ESP32_LED_Control");

    Serial.println("Comandos disponíveis:");

    Serial.println("'ligar' ou '1' - Liga o LED");

    Serial.println("'desligar' ou '0' - Desliga o LED");

    Serial.println("'status' - Mostra o estado atual do LED");


    // Piscar LED 3 vezes para indicar que está pronto

    for(int i = 0; i < 3; i++) {

        digitalWrite(LED_PIN, HIGH);
```

```
    delay(200);  
    digitalWrite(LED_PIN, LOW);  
    delay(200);  
}  
}  
  
void loop() {  
    // Verificar se há dados disponíveis via Bluetooth  
    if (SerialBT.available()) {  
        receivedMessage = SerialBT.readString();  
        receivedMessage.trim(); // Remove espaços em branco  
        receivedMessage.toLowerCase(); // Converte para minúsculas  
  
        // Processar comandos recebidos  
        processCommand(receivedMessage);  
    }  
  
    // Verificar se há dados no Serial Monitor (para debug)  
    if (Serial.available()) {  
        String debugMessage = Serial.readString();  
        debugMessage.trim();  
        debugMessage.toLowerCase();  
        processCommand(debugMessage);  
    }  
  
    delay(20); // Pequeno delay para estabilidade  
}
```

```
void processCommand(String command) {

    Serial.println("Comando recebido: " + command);

    if (command == "ligar" || command == "1" || command == "on") {

        digitalWrite(LED_PIN, HIGH);

        ledState = true;

        SerialBT.println("LED ligado!");

        Serial.println("LED ligado!");

    } else if (command == "desligar" || command == "0" || command == "off") {

        digitalWrite(LED_PIN, LOW);

        ledState = false;

        SerialBT.println("LED desligado!");

        Serial.println("LED desligado!");

    } else if (command == "status" || command == "estado") {

        String status = ledState ? "ligado" : "desligado";

        SerialBT.println("LED está " + status);

        Serial.println("LED está " + status);

    } else if (command == "piscar" || command == "blink") {

        SerialBT.println("Piscando LED...");

        Serial.println("Piscando LED...");

        // Piscar 5 vezes

        for(int i = 0; i < 5; i++) {

            digitalWrite(LED_PIN, HIGH);

            delay(300);

        }

    }

}
```

```

    digitalWrite(LED_PIN, LOW);

    delay(300);
}

// Restaurar estado anterior
digitalWrite(LED_PIN, ledState);
SerialBT.println("Piscada concluída!");

} else if (command == "ajuda" || command == "help") {
    sendHelp();

} else {
    SerialBT.println("Comando não reconhecido!");
    SerialBT.println("Digite 'ajuda' para ver os comandos disponíveis.");
    Serial.println("Comando não reconhecido: " + command);
}
}

void sendHelp() {
    SerialBT.println("=== COMANDOS DISPONÍVEIS ===");
    SerialBT.println("ligar, 1, on - Liga o LED");
    SerialBT.println("desligar, 0, off - Desliga o LED");
    SerialBT.println("status, estado - Mostra estado do LED");
    SerialBT.println("piscar, blink - Pisca o LED 5 vezes");
    SerialBT.println("ajuda, help - Mostra esta ajuda");
    SerialBT.println("=====");
}

```

Configuração do Arduino IDE

1. Instalar ESP32 no Arduino IDE:

- Arquivo → Preferências
- URLs adicionais:
https://dl.espressif.com/dl/package_esp32_index.json
- Ferramentas → Placa → Gerenciador de Placas
- Procurar "ESP32" e instalar

2. Configurações da Placa:

- Placa: ESP32 Dev Module
- Porta: Selecione a porta USB do ESP32

Comandos Disponíveis

Comando Alternativas Função

ligar	1, on	Liga o LED
desligar	0, off	Desliga o LED
status	estado	Mostra estado atual
piscar	blink	Pisca 5 vezes
ajuda	help	Lista comandos

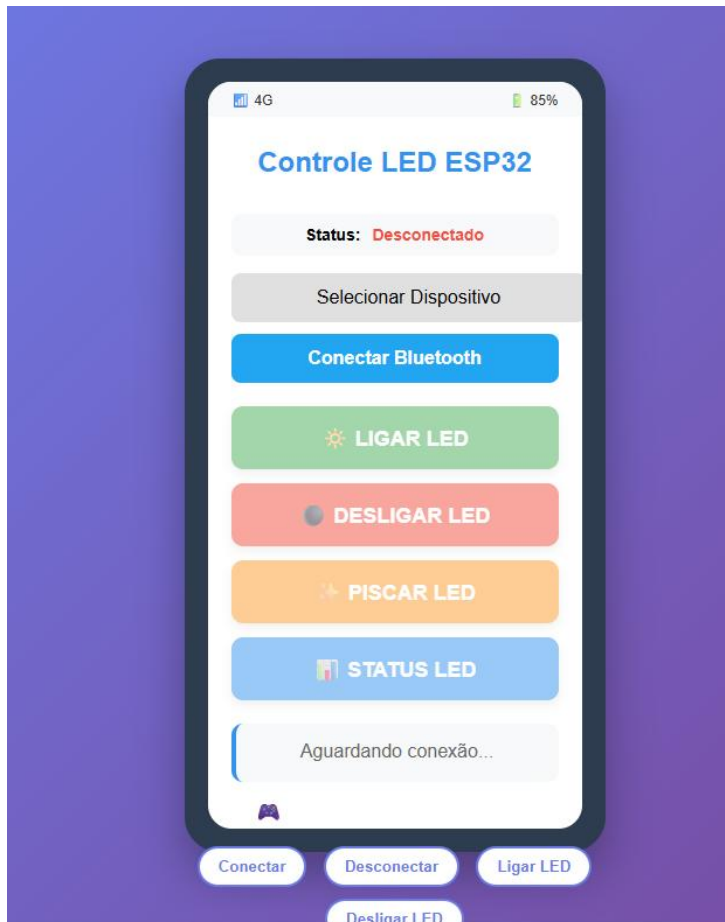
PARTE 2: APLICATIVO MIT APP INVENTOR

Visão Geral

Aplicativo Android usando MIT App Inventor para controlar o LED do ESP32 via Bluetooth com interface visual e intuitiva.

Design da Interface (Designer)

Estrutura da Tela:



Screen1

- └─ VerticalArrangement1 (Principal)
 - | └─ Label1 (Título: "Controle LED ESP32")
 - | └─ HorizontalArrangement1 (Status)
 - | | └─ Label2 ("Status: ")
 - | | └─ Label3 ("Desconectado")
 - | └─ ListPicker1 ("Selecionar Dispositivo")
 - | └─ Button1 ("Conectar Bluetooth")
 - | └─ VerticalArrangement2 (Controles)
 - | | └─ Button2 ("☀️ LIGAR LED")
 - | | └─ Button3 ("● DESLIGAR LED")
 - | | └─ Button4 ("🌟 PISCAR LED")

```
| | └─ Button5 ("📶 STATUS LED")
| └─ Label4 ("Aguardando conexão...")
```

Componentes Necessários:

1. Layout Principal

- **VerticalArrangement1** (container principal)
 - Width: Fill parent
 - Height: Fill parent
 - AlignHorizontal: Center

2. Título do App

- **Label1** (título)
 - Text: "Controle LED ESP32"
 - FontSize: 24
 - FontBold: True
 - TextColor: Blue

3. Seção de Conexão Bluetooth

- **HorizontalArrangement1**
 - **Label2**: "Status: "
 - **Label3** (status): "Desconectado"
 - **Button1** (conectar): "Conectar Bluetooth"

4. Lista de Dispositivos

- **ListPicker1** (seletor de dispositivos)
 - Text: "Selecionar Dispositivo"
 - Width: Fill parent

5. Controles do LED

- **VerticalArrangement2**
 - **Button2** (ligar): "🔆 LIGAR LED"
 - BackgroundColor: Green
 - TextColor: White

- FontSize: 18
- **Button3** (desligar): "● DESLIGAR LED"
 - BackgroundColor: Red
 - TextColor: White
 - FontSize: 18
- **Button4** (piscar): "💡 PISCAR LED"
 - BackgroundColor: Orange
 - TextColor: White
 - FontSize: 18
- **Button5** (status): "📶 STATUS LED"
 - BackgroundColor: Blue
 - TextColor: White
 - FontSize: 18

6. Área de Feedback

- **Label4** (feedback)
 - Text: "Aguardando conexão..."
 - FontSize: 16
 - TextColor: Gray

7. Componentes Não-Visíveis

- **BluetoothClient1** (da paleta Connectivity)
- **Clock1** (da paleta Sensors)
- **Notifier1** (da paleta User Interface)

⚙️ Propriedades Detalhadas dos Componentes

Screen1 (Tela Principal)

- Title: "Controle LED ESP32"
- BackgroundColor: White
- ShowStatusBar: True

VerticalArrangement1

- Width: Fill parent
- Height: Fill parent
- AlignHorizontal: Center
- AlignVertical: Top

Label1 (Título)

- Text: "Controle LED ESP32"
- FontSize: 24
- FontBold: True
- TextColor: Blue
- TextAlignment: Center
- Width: Fill parent

HorizontalArrangement1

- Width: Fill parent
- Height: Automatic
- AlignHorizontal: Center

Label2 (Texto Status)

- Text: "Status: "
- FontSize: 14
- FontBold: True

Label3 (Indicador Status)

- Text: "Desconectado"
- FontSize: 14
- TextColor: Red
- FontBold: True

ListPicker1

- Text: "Seleccionar Dispositivo"
- Width: Fill parent

- Height: 40 pixels
- BackgroundColor: Light Gray
- TextColor: Black
- FontSize: 16


Button1 (Conectar)

- Text: "Conectar Bluetooth"
- Width: Fill parent
- Height: 50 pixels
- BackgroundColor: Light Blue
- TextColor: White
- FontSize: 16
- FontBold: True


VerticalArrangement2

- Width: Fill parent
- Height: Automatic
- AlignHorizontal: Center

Button2 (Ligar LED)

- Text: "  LIGAR LED"
- Width: Fill parent
- Height: 50 pixels
- BackgroundColor: Green
- TextColor: White
- FontSize: 18
- FontBold: True

Button3 (Desligar LED)

- Text: "  DESLIGAR LED"
- Width: Fill parent
- Height: 50 pixels

- BackgroundColor: Red
- TextColor: White
- FontSize: 18
- FontBold: True

Button4 (Piscar LED)

- Text: " ✨ PISCAR LED"
- Width: Fill parent
- Height: 50 pixels
- BackgroundColor: Orange
- TextColor: White
- FontSize: 18
- FontBold: True

Button5 (Status LED)

- Text: " 🇧🇷 STATUS LED"
- Width: Fill parent
- Height: 50 pixels
- BackgroundColor: Blue
- TextColor: White
- FontSize: 18
- FontBold: True

Label4 (Feedback)

- Text: "Aguardando conexão..."
- FontSize: 16
- TextColor: Gray
- TextAlignment: Center
- Width: Fill parent

Componentes Não-Visíveis

BluetoothClient1

- Localização: Connectivity
- Usado para conexão Bluetooth

Clock1

- Localização: Sensors
- TimeInterval: 1000 (1 segundo)
- TimerEnabled: True

Notifier1

- Localização: User Interface
 - Usado para exibir alertas
-

Programação (Blocks)

1. Variáveis Globais

global conectado (initialize to false)

global dispositivoSelecioneado (initialize to "")

2. Inicialização do App

when Screen1.Initialize do

set Label3.Text to "Desconectado"

set Label3.TextColor to Red

set Button2.Enabled to false

set Button3.Enabled to false

set Button4.Enabled to false

set Button5.Enabled to false

3. Seleção de Dispositivo Bluetooth

when ListPicker1.BeforePicking do

set ListPicker1.Elements to BluetoothClient1.AddressesAndNames

when ListPicker1.AfterPicking do

```
set global dispositivoSelecionado to ListPicker1.Selection  
set Button1.Text to join("Conectar a ", ListPicker1.SelectionIndex)
```

4. Conexão Bluetooth

```
when Button1.Click do  
  if global conectado then  
    call BluetoothClient1.Disconnect  
    set global conectado to false  
    set Label3.Text to "Desconectado"  
    set Label3.TextColor to Red  
    set Button1.Text to "Conectar Bluetooth"  
    set Button2.Enabled to false  
    set Button3.Enabled to false  
    set Button4.Enabled to false  
    set Button5.Enabled to false  
    set Label4.Text to "Desconectado do ESP32"  
  else  
    if length(global dispositivoSelecionado) > 0 then  
      call BluetoothClient1.ConnectWithUUID(  
        select list item list: global dispositivoSelecionado index: 2,  
        "00001101-0000-1000-8000-00805F9B34FB"  
      )
```

5. Verificação de Conexão

```
when Clock1.Timer do  
  if BluetoothClient1.IsConnected then  
    if not global conectado then  
      set global conectado to true  
      set Label3.Text to "Conectado"  
      set Label3.TextColor to Green
```

```
set Button1.Text to "Desconectar"

set Button2.Enabled to true

set Button3.Enabled to true

set Button4.Enabled to true

set Button5.Enabled to true

set Label4.Text to "Conectado ao ESP32!"

call Notifier1.ShowAlert("Conectado com sucesso!")

else

if global conectado then

set global conectado to false

set Label3.Text to "Desconectado"

set Label3.TextColor to Red

set Button1.Text to "Conectar Bluetooth"

set Button2.Enabled to false

set Button3.Enabled to false

set Button4.Enabled to false

set Button5.Enabled to false

set Label4.Text to "Conexão perdida!"
```

6. Controles do LED

```
when Button2.Click do (LIGAR LED)

if global conectado then

call BluetoothClient1.SendText("ligar")

set Label4.Text to "LED ligado!"

set Label4.TextColor to Green


when Button3.Click do (DESLIGAR LED)

if global conectado then

call BluetoothClient1.SendText("desligar")
```

set Label4.Text to "LED desligado!"

set Label4.TextColor to Red

when Button4.Click do (PISCAR LED)

if global conectado then

call BluetoothClient1.SendText("piscar")

set Label4.Text to "LED piscando..."

set Label4.TextColor to Orange

when Button5.Click do (STATUS LED)

if global conectado then

call BluetoothClient1.SendText("status")

set Label4.Text to "Verificando status..."

set Label4.TextColor to Blue

7. Configuração do Clock

when Screen1.Initialize do

set Clock1.TimerInterval to 1000

set Clock1.TimerEnabled to true

PARTE 3: PASSO A PASSO COMPLETO



Passo a Passo para Criar

ETAPA 1: Preparar ESP32

1. Conecte ESP32 ao computador
2. Abra Arduino IDE
3. Configure placa ESP32
4. Cole o código fornecido
5. Compile e carregue no ESP32
6. Verifique no Serial Monitor se está funcionando

ETAPA 2: Criar App no App Inventor

1. Acesse: <http://ai2.appinventor.mit.edu>
2. Faça login com conta Google
3. Clique em "Start new project"
4. Nome: "ControleLED_ESP32"

ETAPA 3: Montar Interface

1. Arraste componentes da paleta para a tela
2. Configure propriedades conforme tabela acima
3. Organize layout visualmente
4. Adicione componentes não-visíveis

ETAPA 4: Programar Blocos

1. Clique em "Blocks" no canto superior direito
2. Crie variáveis globais
3. Programe inicialização
4. Configure conexão Bluetooth
5. Programe controles do LED
6. Adicione verificação de conexão

ETAPA 5: Testar Aplicativo

1. Instale "MIT AI2 Companion" no celular
2. Conecte celular na mesma rede WiFi
3. Escaneie QR code ou digite código
4. Teste todas as funcionalidades
5. Verifique conexão com ESP32

ETAPA 6: Gerar APK Final

1. Build → App (provide QR code for .apk)
2. Baixe o arquivo APK
3. Transfira para o celular Android
4. Instale o APK

5. Teste funcionamento completo

Resolução de Problemas

LED não liga:

- Verifique as conexões do circuito
- Confirme se o resistor está correto
- Teste com o LED interno primeiro (GPIO2)
- Verifique se o código foi carregado corretamente

Bluetooth não conecta:

- Certifique-se que o ESP32 está ligado e funcionando
- Verifique se o Bluetooth do celular está ativo
- Confirme se o dispositivo "ESP32_LED_Control" aparece
- Tente desconectar e conectar novamente
- Verifique se outro dispositivo não está conectado ao ESP32

App não funciona:

- Verifique se todas as permissões estão concedidas
- Confirme se o Bluetooth está habilitado
- Teste conexão com Serial Monitor primeiro
- Verifique se o UUID está correto
- Reinstale o aplicativo se necessário

Comandos não funcionam:

- Verifique se está conectado ao dispositivo correto
- Teste os comandos pelo Serial Monitor primeiro
- Confirme se está enviando apenas o comando
- Verifique se não há caracteres especiais

Melhorias e Personalizações

Cores Sugeridas:

- Fundo do App: Branco (#FFFFFF)
- Título: Azul (#2196F3)

- Botão Ligar: Verde (#4CAF50)
- Botão Desligar: Vermelho (#F44336)
- Botão Piscar: Laranja (#FF9800)
- Botão Status: Azul (#2196F3)

Ícones Emoji:

- Ligar: ☀️
- Desligar: ●
- Piscar: ✨
- Status: 📊
- Conectado: 🔗
- Desconectado: ❌

Funcionalidades Extras:

- Adicionar controle de brilho
- Incluir mais LEDs
- Criar padrões de piscada
- Adicionar sons de feedback
- Implementar timer automático

🔧 Dicas Importantes

Permissões:

- O app precisa de permissão Bluetooth
- Ative localização no Android 6+
- Conceda todas as permissões solicitadas

Compatibilidade:

- Funciona com Android 4.1+
- Requer Bluetooth clássico (não BLE)
- UUID padrão: "00001101-0000-1000-8000-00805F9B34FB"

Segurança:

- Código de pareamento padrão: 1234 ou 0000
- Alcance típico: 10 metros
- Conexão ponto-a-ponto (um dispositivo por vez)

Funcionalidades do Sistema Completo

✓ **Interface intuitiva com botões grandes** ✓ **Conexão automática via Bluetooth** ✓ **Feedback visual do status em tempo real** ✓ **Controle completo do LED (ligar/desligar/piscar/status)** ✓ **Tratamento de erros de conexão** ✓ **Design responsivo e moderno** ✓ **Compatibilidade com Serial Bluetooth Terminal** ✓ **Código ESP32 otimizado e confiável** ✓ **Documentação completa e detalhada**

Resultado Final

O sistema completo oferece:

- **Controle profissional** do LED via smartphone
- **Interface visual moderna** e fácil de usar
- **Conexão Bluetooth estável** e confiável
- **Feedback em tempo real** das ações
- **Código bem documentado** e expansível
- **Compatibilidade** com diferentes aplicativos Bluetooth