← Voltar ao curso

CURSO BÁSICO

Capítulo 7 - Acionamento simples de LEDs

Agora vamos usar as saídas da ESP32 para criar indicações visuais e começar a aplicá-las diretamente à ferrovia.

Introdução	Um LED é uma maneira simples de visualizar o estado de uma saída da ESP32. Na ferrovia, LEDs poderão ser utilizados para indicar situações como equipamento ligado, estado de uma função, autorização de movimento, alerta ou sinalização no painel. Neste capítulo vamos aprender a ligar, desligar e piscar um LED por meio do programa.
Objetivo	Compreender como controlar um LED através de uma saída da ESP32 e relacionar os estados ligado e desligado com informações visuais da ferrovia.
Meta de aprendizagem	Ao final, você deverá ser capaz de configurar uma saída da ESP32, ligar e desligar um LED pelo programa e fazê-lo piscar utilizando a temporização aprendida no capítulo anterior.

1. O LED como indicador visual

O LED pode ser utilizado durante os primeiros testes de programação porque permite ver imediatamente o resultado de uma instrução enviada pela ESP32.

Quando programamos uma saída para ficar ligada, o LED acende. Quando a saída é desligada, o LED apaga.

> Na Ferrovia EFMN 74, esse mesmo princípio poderá ser utilizado posteriormente para mostrar visualmente diferentes estados do sistema de controle.

2. Ligação básica do LED

Para fazer o teste podemos utilizar um LED externo ligado a uma saída da ESP32.

O LED deve ser utilizado com um resistor limitador de corrente ligado em série. Esse resistor protege o LED e também a saída da ESP32.

> Para os exercícios deste capítulo utilizaremos a saída `GPIO 25`.

A ligação básica será:

```
GPIO 25 → resistor → LED → GND
```

O LED possui polaridade. Portanto, suas duas pernas não devem ser ligadas de qualquer maneira.

- A perna mais longa normalmente corresponde ao lado positivo.
- A perna mais curta normalmente corresponde ao lado negativo.

3. Preparando a saída no programa

Antes de controlar o LED precisamos informar à ESP32 que o pino escolhido será utilizado como saída.

```
const int ledPin = 25;

void setup() {
  pinMode(ledPin, OUTPUT);
}
```

A linha:

```
pinMode(ledPin, OUTPUT);
```

informa que o pino indicado pela variável `ledPin` funcionará como uma saída.

4. Ligando o LED

Depois de configurar o pino como saída podemos utilizar `digitalWrite()`.

```
digitalWrite(ledPin, HIGH);
```

O comando `HIGH` coloca a saída em nível ligado.

No circuito utilizado neste exercício, isso fará o LED acender.

5. Desligando o LED

Para desligar a saída utilizamos:

```
digitalWrite(ledPin, LOW);
```

O comando `LOW` coloca a saída em nível desligado e o LED apaga.

6. Primeiro programa completo

Vamos montar um programa simples no qual o LED permanece aceso.

```
const int ledPin = 25;

void setup() {
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
}

void loop() {

}
```

Quando o programa for enviado para a ESP32, o LED deverá permanecer ligado.

Agora altere:

```
digitalWrite(ledPin, HIGH);
```

para:

```
digitalWrite(ledPin, LOW);
```

Envie novamente o programa para a ESP32 e observe a diferença.

7. Fazendo o LED piscar

No capítulo anterior aprendemos a utilizar `millis()` para controlar o tempo sem interromper desnecessariamente o funcionamento do programa.

Podemos aplicar exatamente o mesmo princípio ao LED.

```
const int ledPin = 25;

bool ledLigado = false;

unsigned long tempoAnterior = 0;
const unsigned long intervalo = 1000;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {

  unsigned long tempoAtual = millis();

  if (tempoAtual - tempoAnterior >= intervalo) {

    tempoAnterior = tempoAtual;

    ledLigado = !ledLigado;

    digitalWrite(ledPin, ledLigado);
```

```
  }
}
```

Nesse programa, o estado do LED é alterado aproximadamente a cada segundo.

Enquanto o LED pisca, a ESP32 continua livre para executar outras instruções.

8. Entendendo a variável ledLigado

Observe a seguinte linha:

```
bool ledLigado = false;
```

A variável `ledLigado` pode guardar apenas dois estados: verdadeiro ou falso.

Podemos pensar nesses estados de maneira simples:

Valor	Situação
`false`	LED desligado
`true`	LED ligado

A linha:

```
ledLigado = !ledLigado;
```

faz a variável trocar de estado.

Se ela estiver em `false`, passa para `true`. Se estiver em `true`, passa para `false`.

9. Aplicação na Ferrovia EFMN 74

Um LED não precisa representar apenas uma lâmpada de teste. Ele pode funcionar como uma indicação do estado da ferrovia.

Por exemplo, um LED poderá indicar:

- que determinada função está ligada;
- que uma ação está sendo executada;

• que existe uma condição de alerta;

• que determinado equipamento está ativo;

• que uma condição necessária para uma operação foi atendida.

> O importante neste momento não é construir toda a sinalização da ferrovia, mas compreender que o programa pode transformar uma condição interna da ESP32 em uma informação visual facilmente observada pelo operador.

10. Dois estados, uma informação

Em automação é comum uma informação possuir apenas duas condições.

Estado da saída	Indicação visual
`HIGH`	LED aceso
`LOW`	LED apagado

Mais adiante poderemos combinar várias saídas e criar indicações visuais mais completas. Por enquanto, dominar corretamente uma única saída é o passo mais importante.

EXERCÍCIO PRÁTICO

Monte um LED na saída `GPIO 25` da ESP32, utilizando o resistor de proteção.

1. Configure o `GPIO 25` como saída.
2. Faça o LED permanecer ligado.
3. Altere o programa para deixá-lo desligado.
4. Utilize o exemplo com `millis()` para fazer o LED piscar a cada 1 segundo.
5. Altere o intervalo para `500` milissegundos.
6. Depois altere o intervalo para `2000` milissegundos.

7. Observe e compare os três comportamentos.

O exercício de piscar o LED deve ser realizado sem utilizar `delay()`.

Resumo do capítulo

- Um LED permite visualizar o estado de uma saída da ESP32.
- `pinMode()` configura o pino que será utilizado.
- `OUTPUT` define o pino como saída.
- `digitalWrite()` altera o estado da saída.
- `HIGH` liga a saída.
- `LOW` desliga a saída.
- Um LED deve ser utilizado com resistor limitador de corrente.
- `millis()` permite fazer o LED piscar sem interromper o programa.
- Na ferrovia, LEDs poderão representar visualmente diferentes estados do sistema.

Próximo capítulo: Entradas digitais e leitura de sensores.

← Voltar ao curso