← Voltar ao curso

CURSO BÁSICO

Capítulo 8 - Leitura de botões e sensores

Agora vamos fazer a ESP32 receber informações do mundo real e utilizá-las dentro do programa.

Introdução	Nos capítulos anteriores vimos como a ESP32 pode comandar uma saída, como um LED. Agora faremos o caminho contrário: um botão ou sensor enviará uma informação para a ESP32. Essa leitura será a base para detectar acontecimentos na ferrovia, como a passagem de uma composição por determinado ponto.
Objetivo	Compreender como a ESP32 lê uma entrada digital e aprender a utilizar `digitalRead()` para identificar o estado de um botão ou de um sensor simples.
Meta de aprendizagem	Ao final, você deverá ser capaz de configurar um pino como entrada, ler seu estado, identificar os valores `HIGH` e `LOW` e utilizar essa informação para comandar uma saída simples.

1. A ESP32 também recebe informações

Até agora usamos a ESP32 principalmente para enviar comandos. Ao ligar um LED, por exemplo, o programa coloca uma saída em determinado estado.

Mas a automação ferroviária também precisa receber informações. É dessa forma que o programa poderá saber que alguma coisa aconteceu na maquete.

> Um botão pressionado ou um sensor acionado pode informar à ESP32 que existe uma nova condição que precisa ser analisada pelo programa.

2. Entrada e saída

É importante manter clara a diferença:

Tipo	Função	Exemplo
Entrada	Leva uma informação para a ESP32.	Botão ou sensor.
Saída	Recebe um comando enviado pela ESP32.	LED.

Neste capítulo, o foco será a entrada digital.

3. Preparando um pino como entrada

Para ler um botão, precisamos escolher um pino da ESP32 e configurá-lo como entrada. Neste exemplo utilizaremos o `GPIO 27`.

```
const int botaoPin = 27;

void setup() {
  pinMode(botaoPin, INPUT_PULLUP);
}
```

A instrução `INPUT_PULLUP` configura o pino como entrada e ativa internamente um recurso que ajuda a manter a leitura estável quando o botão não está pressionado.

4. Ligação simples do botão

Com `INPUT_PULLUP`, podemos fazer uma ligação simples:

```
GPIO 27 → botão → GND
```

Nesse tipo de ligação:

Situação do botão	Leitura da ESP32
Solto	`HIGH`
Pressionado	`LOW`

> Neste caso, `LOW` significa que o botão foi pressionado. Isso pode parecer invertido no início, mas ficará natural com a prática.

5. A função digitalRead()

Para descobrir o estado de uma entrada digital usamos:

```
digitalRead(botaoPin);
```

O resultado será `HIGH` ou `LOW`. Podemos guardar esse resultado em uma variável:

```
int estadoBotao = digitalRead(botaoPin);
```

Agora a variável `estadoBotao` representa a situação lida naquele instante.

6. Tomando uma decisão com a leitura

Depois de ler o botão, o programa pode verificar seu estado.

```
if (estadoBotao == LOW) {
  // botão pressionado
}
```

A palavra `if` significa “se”. Portanto, podemos ler essa instrução como:

> “Se o estado do botão for `LOW`, execute as instruções que estiverem dentro das chaves.”

7. Acendendo um LED ao pressionar o botão

Vamos aproveitar o LED utilizado no capítulo anterior. O botão será ligado ao `GPIO 27` e o LED continuará no `GPIO 25`.

```
const int botaoPin = 27;
const int ledPin = 25;

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

void loop() {
  int estadoBotao = digitalRead(botaoPin);

  if (estadoBotao == LOW) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Enquanto o botão estiver pressionado, o LED ficará aceso. Ao soltar o botão, o LED apagará.

8. Entendendo o else

No exemplo anterior aparece a palavra `else`. Ela significa “caso contrário”.

Podemos interpretar o trecho assim:

> Se o botão estiver pressionado, acenda o LED.
> Caso contrário, apague o LED.

Essa forma de decisão será muito usada na automação da ferrovia.

9. Do botão para o sensor

Para a ESP32, muitos sensores digitais funcionam de maneira semelhante a um botão: eles apresentam um estado elétrico que pode ser lido pelo programa.

O princípio é o mesmo:

1. o sensor percebe uma condição;
2. a entrada da ESP32 muda de estado;
3. `digitalRead()` faz a leitura;
4. o programa decide o que fazer.

Nem todos os sensores possuem exatamente a mesma forma de ligação. Por isso, antes de conectar um modelo específico, deve-se verificar sua tensão de alimentação e a forma correta de ligação do sinal.

10. Aplicação na Ferrovia EFMN 74

Na Ferrovia EFMN 74, os sensores terão uma função importante: informar ao sistema a presença ou a passagem das composições em pontos determinados da maquete.

> Imagine uma composição passando por um sensor. A ESP32 recebe essa informação e o programa pode usá-la como início de uma sequência: verificar uma condição, atualizar uma indicação, preparar um desvio ou executar outra ação programada.

Neste capítulo ainda não precisamos controlar toda essa sequência. O objetivo é dominar primeiro a leitura correta de uma única entrada.

11. Um princípio importante

Um sensor não decide o que a ferrovia deve fazer. Ele apenas fornece uma informação.

Quem interpreta essa informação é o programa executado pela ESP32.

```
sensor → entrada da ESP32 → programa → decisão → ação
```

Esse caminho será repetido muitas vezes nos próximos capítulos.

EXERCÍCIO PRÁTICO

Monte um botão no `GPIO 27` usando `INPUT_PULLUP` e utilize o LED do `GPIO 25` como indicação visual.

1. Configure o `GPIO 27` como `INPUT_PULLUP`.
2. Configure o `GPIO 25` como `OUTPUT`.
3. Leia o botão com `digitalRead()`.
4. Faça o LED acender enquanto o botão estiver pressionado.
5. Faça o LED apagar quando o botão for solto.
6. Pressione e solte o botão várias vezes e observe o comportamento.

Depois de compreender o funcionamento com o botão, pense nele como uma representação simplificada de um sensor instalado junto aos trilhos.

Resumo do capítulo

- Botões e sensores fornecem informações para a ESP32.
- Um pino de entrada pode ser configurado com `INPUT_PULLUP`.
- `digitalRead()` lê o estado de uma entrada digital.
- A leitura digital pode resultar em `HIGH` ou `LOW`.
- Na ligação apresentada, botão solto corresponde a `HIGH` e pressionado a `LOW`.
- `if` permite executar uma ação quando uma condição é atendida.
- `else` define o que fazer caso a condição não seja atendida.
- O mesmo princípio básico de leitura será usado posteriormente com sensores da ferrovia.

Próximo capítulo: Primeiro acionamento de um desvio.

← Voltar ao curso