← Voltar ao curso

CURSO BÁSICO

Capítulo 5 - Entradas e saídas digitais

Agora vamos entender como a ESP32 recebe informações da ferrovia e como envia comandos para os componentes ligados aos seus pinos.

Introdução	A ESP32 se comunica com muitos componentes por meio de pinos. Alguns pinos podem receber informações, funcionando como entradas, e outros podem enviar comandos, funcionando como saídas. Neste capítulo veremos essa diferença e aprenderemos a preparar os pinos no programa.
Objetivo	Compreender o conceito de entradas e saídas digitais e aprender a configurar pinos da ESP32 com `pinMode()`.
Meta de aprendizagem	Ao final, você deverá conseguir identificar se um componente fornece uma informação ou recebe um comando, escolher entre `INPUT` e `OUTPUT` e configurar pinos digitais em um programa simples.

1. A ESP32 precisa conversar com a ferrovia

Nos capítulos anteriores vimos que a ESP32 executa um programa e pode guardar informações. Agora precisamos criar a ligação lógica entre o programa e os componentes reais da ferrovia.

Essa comunicação acontece principalmente pelos pinos da placa. Em muitos casos, um pino será usado para receber uma informação e outro será usado para enviar um comando.

> **Exemplo ferroviário:** um sensor pode informar que uma composição passou por determinado ponto. A ESP32 recebe essa informação por uma entrada. Depois, o programa poderá mandar acender um sinal ou comandar outro dispositivo por uma saída.

2. O que é um GPIO?

Nos esquemas e programas da ESP32 aparece frequentemente a sigla `GPIO`. Para este curso, basta entender que GPIO é um pino da placa que pode ser usado pelo programa para receber ou enviar sinais digitais.

Os GPIOs são identificados por números. Assim, expressões como `GPIO25` ou `GPIO26` indicam pinos específicos da ESP32.

Nem todos os pinos da placa devem ser usados de qualquer maneira. Ao montar cada exercício, utilizaremos apenas pinos adequados à função proposta.

3. Entrada digital

Uma entrada digital é usada quando a ESP32 precisa receber uma informação externa.

Na ferrovia, entradas poderão ser utilizadas para receber informações de:

- botões de comando;
- sensores de passagem ou presença;
- contatos de confirmação;
- outros dispositivos que informem apenas dois estados ao programa.

Quando um pino será usado como entrada, podemos configurá-lo com:

```
pinMode(sensorPin, INPUT);
```

A instrução `pinMode()` informa à ESP32 qual função aquele pino deverá exercer.

4. Saída digital

Uma saída digital é usada quando a ESP32 precisa enviar um comando para outro componente.

Na ferrovia, uma saída poderá ser usada, diretamente ou por meio de um circuito apropriado, para comandar:

- LEDs e sinais luminosos;
- relés;
- buzzers e indicações sonoras simples;
- circuitos que acionem outros equipamentos da maquete.

Quando um pino será usado como saída, configuramos:

```
pinMode(signalPin, OUTPUT);
```

5. INPUT e OUTPUT

As palavras `INPUT` e `OUTPUT` indicam a direção da informação em relação à ESP32.

Configuração	Sentido da informação	Exemplo na ferrovia
`INPUT`	Do componente para a ESP32.	Um sensor informa uma passagem.
`OUTPUT`	Da ESP32 para o componente.	A ESP32 envia um comando para uma indicação luminosa.

Uma forma simples de guardar essa ideia é:

> **Entrada = a ESP32 recebe informação.**
> **Saída = a ESP32 envia comando.**

6. HIGH e LOW

Em uma entrada ou saída digital trabalhamos normalmente com dois estados lógicos, representados no programa pelas palavras `HIGH` e `LOW`.

Neste momento, não é necessário associar automaticamente `HIGH` a “ligado” e `LOW` a “desligado” em todas as situações. Dependendo de como um componente estiver ligado, a interpretação prática poderá ser diferente.

O importante agora é reconhecer que um sinal digital trabalha com dois estados e que o programa poderá ler ou definir esses estados.

7. digitalRead() e digitalWrite()

Depois que um pino foi configurado, duas instruções serão muito importantes nos próximos exercícios.

Instrução	Função
`digitalRead()`	Lê o estado de uma entrada digital.
`digitalWrite()`	Define o estado de uma saída digital.

Por enquanto, vamos apenas reconhecer essas instruções. A leitura prática de botões e sensores será estudada mais adiante, e o acionamento de LEDs terá um capítulo próprio.

8. Configurando dois pinos no setup()

Como a função de cada pino precisa estar definida antes do funcionamento normal do programa, a configuração com `pinMode()` é feita dentro de `setup()`.

```
const int sensorPin = 25;
const int signalPin = 26;

void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(signalPin, OUTPUT);
}
```

```
void loop() {
  // Nos próximos capítulos iremos ler entradas
  // e comandar saídas.
}
```

Observe que os números dos pinos foram guardados em constantes, aproveitando o que aprendemos no capítulo anterior.

O programa ainda não executa uma ação visível. Seu objetivo é apenas mostrar como declarar os pinos e definir a função de cada um.

9. Pensando como o sistema de automação

A partir deste capítulo, podemos enxergar uma parte importante da lógica da ferrovia:

> **Entrada → programa → saída**
>
> Um componente envia uma informação para a ESP32; o programa analisa essa informação; depois, quando necessário, a ESP32 envia um comando por uma saída.

Nos capítulos seguintes iremos transformar gradualmente essa sequência em testes reais de bancada.

EXERCÍCIO PRÁTICO

Meta: criar e compilar um programa simples que configure uma entrada e uma saída digital.

1. Abra a Arduino IDE.
2. Crie um novo programa.
3. Digite o programa apresentado no item 8.
4. Salve como `basico-05-entradas-saidas`.
5. Clique em Verificar/Compilar.
6. Troque o nome `sensorPin` por `buttonPin` em todos os pontos necessários e compile novamente.

7. Confirme que o pino de entrada continua configurado como `INPUT` e o pino de saída como `OUTPUT`.

Resultado esperado: o programa deve compilar sem erros e você deverá conseguir explicar qual pino recebe informação e qual pino envia comando.

Resumo do capítulo

- GPIO é um pino da ESP32 que pode participar da comunicação com os componentes.
- Uma entrada recebe informação; uma saída envia comando.
- `pinMode()` define a função de um pino.
- `INPUT` configura uma entrada e `OUTPUT` configura uma saída.
- `HIGH` e `LOW` representam os dois estados lógicos de um sinal digital.
- `digitalRead()` será usado para ler entradas e `digitalWrite()` para comandar saídas.

Próximo capítulo: Temporização e uso de millis().

← Voltar ao curso