CURSO BÁSICO

Capítulo 9 - Primeiro acionamento de um desvio

Chegou o momento de transformar um comando da ESP32 em movimento real na ferrovia.

Introdução	Nos capítulos anteriores aprendemos a controlar uma saída, trabalhar com temporização e ler botões e sensores. Agora vamos utilizar esses conhecimentos para movimentar, pela primeira vez, um desvio da ferrovia com um pequeno servomotor.
Objetivo	Compreender como a ESP32 controla um servomotor e realizar o primeiro movimento entre duas posições que poderão representar as posições de um desvio ferroviário.
Meta de aprendizagem	Ao final, você deverá ser capaz de identificar as ligações básicas de um servo, instalar a biblioteca necessária, criar um objeto do tipo `Servo`, associá-lo a um GPIO e comandar duas posições diferentes com `write()`.

1. Do comando elétrico ao movimento mecânico

Um desvio ferroviário possui agulhas móveis que precisam ser deslocadas de uma posição para outra. Na Ferrovia EFMN 74, esse movimento será realizado por pequenos servomotores instalados ao lado dos desvios.

A ESP32 não movimenta diretamente o desvio. Ela envia um sinal de controle ao servomotor, e o servo transforma esse comando em movimento mecânico.

> Podemos resumir o processo assim:

```
ESP32 → sinal de controle → servomotor → haste → agulhas do desvio
```

2. O servomotor SG90

Neste curso utilizaremos como referência o servomotor SG90, já empregado nos testes da Ferrovia EFMN 74. Ele possui um pequeno braço móvel que pode ser colocado em diferentes posições por meio do programa.

Normalmente o SG90 possui três fios:

Fio	Função
Vermelho	Alimentação de 5 V.
Marrom ou preto	GND.
Laranja, amarelo ou semelhante	Sinal de controle vindo da ESP32.

As cores podem variar entre fabricantes. Antes da ligação, confirme sempre a identificação dos fios do servo utilizado.

3. Alimentação: um cuidado muito importante

O fio de sinal do servo pode ser comandado por um GPIO da ESP32, mas a alimentação do servomotor deve vir de uma fonte de 5 V adequada.

> **Atenção: Não alimente o SG90 pelo pino de 3,3 V da ESP32.**

A fonte de 5 V e a ESP32 precisam ter o GND em comum. Sem essa referência comum, o sinal de controle pode não funcionar corretamente.

```
Fonte 5 V (+)   → fio vermelho do servo
Fonte GND       → fio marrom/preto do servo
ESP32 GND       → mesmo GND da fonte
ESP32 GPIO 18   → fio de sinal do servo
```

4. Biblioteca para controlar o servo

Para facilitar o controle do servomotor, utilizaremos uma biblioteca própria para servos compatível com a ESP32.

Na Arduino IDE, instale a biblioteca ESP32Servo pelo Gerenciador de Bibliotecas. Depois, no início do programa, inclua:

```
#include <ESP32Servo.h>
```

Essa instrução permite utilizar comandos específicos para controle de servomotores.

5. Criando o servo no programa

Depois de incluir a biblioteca, precisamos criar um objeto que representará o servo controlado pelo programa.

```
Servo desvio;
```

O nome `desvio` foi escolhido para deixar o programa fácil de compreender. Poderíamos utilizar outro nome, mas nomes claros ajudam muito quando o sistema crescer.

6. Ligando o servo a um GPIO

Neste primeiro exemplo utilizaremos o GPIO 18 como saída de controle. Dentro de `setup()`, associamos o servo a esse pino:

```
void setup() {
  desvio.attach(18);
}
```

A instrução `attach()` informa à biblioteca qual GPIO enviará o sinal de controle para o servomotor.

7. Comandando uma posição

Para posicionar o servo usamos a instrução `write()`.

```
desvio.write(67);
```

O número representa uma posição de referência para o braço do servo. Em outro momento podemos enviar um segundo valor:

```
desvio.write(97);
```

Nos testes realizados na Ferrovia EFMN 74, os valores 67 e 97 produziram um movimento adequado em uma montagem específica.

> Esses valores não devem ser considerados universais. A posição do servo, o comprimento da haste, a geometria da instalação e o próprio desvio alteram o ajuste necessário. Cada desvio deverá ser calibrado individualmente.

8. Primeiro programa de movimento

Vamos agora montar um programa simples para colocar o servo inicialmente em uma posição e, em seguida, comandar a outra posição com um botão.

O botão utilizará o mesmo princípio estudado no capítulo anterior: GPIO 27 configurado como `INPUT_PULLUP`.

```
#include <ESP32Servo.h>

Servo desvio;
const int servoPin = 18;
const int botaoPin = 27;

const int posicao1 = 67;
const int posicao2 = 97;

void setup() {
  pinMode(botaoPin, INPUT_PULLUP);
  desvio.attach(servoPin);
  desvio.write(posicao1);
}

void loop() {
  int estadoBotao = digitalRead(botaoPin);
  if (estadoBotao == LOW) {
    desvio.write(posicao2);
  } else {
    desvio.write(posicao1);
  }
}
```

9. Entendendo o programa

O programa reúne conhecimentos dos capítulos anteriores e acrescenta o controle do servo.

Trecho	Função
`#include <ESP32Servo.h>`	Disponibiliza os comandos de controle do servo.
`Servo desvio;`	Cria o objeto que representa o servomotor.
`desvio.attach(servoPin);`	Associa o servo ao GPIO escolhido.
`desvio.write(posicao1);`	Comanda a primeira posição.
`desvio.write(posicao2);`	Comanda a segunda posição.
`digitalRead(botaoPin)`	Lê o estado do botão.
`if / else`	Escolhe qual posição deverá ser comandada.

10. DI e AC: nomes ferroviários para as posições

Na Ferrovia EFMN 74, as duas posições de um desvio serão identificadas como DI e AC.

Por enquanto, o mais importante é compreender que o servo terá duas posições de trabalho. Em capítulos posteriores, o programa poderá substituir nomes genéricos como `posicao1` e `posicao2` por nomes diretamente relacionados à operação da ferrovia.

> A ideia será evoluir de `posicao1 / posicao2` para uma lógica ferroviária mais clara: `DI / AC`.

11. Ajuste mecânico sem forçar o desvio

O objetivo não é fazer o servo percorrer o maior ângulo possível. Ele deve movimentar a haste apenas o suficiente para deslocar as agulhas corretamente.

Se o servo continuar tentando avançar depois que a agulha já encostou no trilho, o mecanismo poderá ficar sob esforço desnecessário.

Por isso, faça os primeiros testes com pequenas diferenças entre as posições e aumente ou reduza os valores gradualmente.

> Um movimento correto deve encostar a agulha com firmeza suficiente para o funcionamento do desvio, mas sem impacto e sem manter o servo forçando o mecanismo.

12. Aplicação na Ferrovia EFMN 74

O primeiro teste com um único servo representa a menor unidade do futuro sistema de acionamento dos desvios.

Mais adiante, cada ESP32 poderá comandar vários desvios de sua área, mas o princípio básico continuará sendo o mesmo:

```
programa → posição desejada → GPIO → servo → desvio
```

Antes de trabalhar com vários desvios ao mesmo tempo, é importante dominar completamente o acionamento de apenas um.

EXERCÍCIO PRÁTICO

Monte um SG90 em bancada, sem ligá-lo inicialmente ao desvio da maquete, e faça o primeiro teste de duas posições.

1. Instale a biblioteca ESP32Servo na Arduino IDE.
2. Ligue o sinal do servo ao GPIO 18.
3. Alimente o servo com 5 V por uma fonte adequada.
4. Una o GND da fonte ao GND da ESP32.
5. Ligue um botão entre o GPIO 27 e o GND.
6. Compile e envie o programa apresentado neste capítulo.
7. Com o botão solto, observe a primeira posição do servo.
8. Pressione o botão e observe o deslocamento para a segunda posição.
9. Solte o botão e confirme o retorno à primeira posição.
10. Somente depois do teste em bancada, ligue a haste ao desvio e faça o ajuste mecânico com cuidado.

> Comece com os valores 67 e 97 apenas como referência. Se necessário, altere-os gradualmente até encontrar as duas posições adequadas para o desvio que estiver sendo testado.

Resumo do capítulo

- O servomotor transforma o comando da ESP32 em movimento mecânico.
- O SG90 utiliza alimentação, GND e um fio de sinal.
- O servo deve ser alimentado por 5 V adequado e não pelo pino de 3,3 V da ESP32.
- A fonte do servo e a ESP32 precisam compartilhar o mesmo GND.
- A biblioteca ESP32Servo facilita o controle do servomotor.
- `attach()` associa o servo a um GPIO.
- `write()` comanda uma posição do servo.
- Um botão pode selecionar entre duas posições.
- Os valores 67 e 97 são referências de teste, não valores fixos para todos os desvios.
- Cada desvio deverá ser ajustado individualmente para evitar esforço mecânico.

Próximo capítulo: Controle de um desvio com comandos DI e AC.