CURSO BÁSICO

Capítulo 10 - Exercícios práticos aplicados à ferrovia

É hora de reunir os conhecimentos do Curso Básico e transformá-los em pequenas funções reais da Ferrovia EFMN 74.

Introdução	Ao longo do Curso Básico conhecemos a estrutura de um programa, variáveis, entradas, saídas, temporização, LEDs, botões, sensores e o primeiro acionamento de um desvio. Neste capítulo, esses conhecimentos serão retomados em exercícios progressivos.
Objetivo	Praticar os principais conceitos estudados e integrá-los em situações simples de automação ferroviária, observando sempre a ligação elétrica, o comportamento do programa e o movimento mecânico.
Meta de aprendizagem	Ao final, você deverá ser capaz de montar, testar e explicar pequenos programas que leem uma entrada, tomam uma decisão, controlam uma indicação luminosa e comandam as posições DI e AC de um desvio.

1. Como realizar os exercícios

Os exercícios devem ser feitos em sequência. Cada montagem acrescenta apenas um novo elemento, permitindo identificar com facilidade a origem de qualquer funcionamento incorreto.

1. Monte o circuito com a alimentação desligada.
2. Confira todas as ligações antes de energizar.
3. Compile o programa e observe as mensagens da Arduino IDE.

4. Envie o programa para a ESP32.
5. Teste uma função de cada vez.
6. Anote os resultados e qualquer ajuste realizado.

> Se um exercício não funcionar como esperado, não avance imediatamente para o seguinte. Confira primeiro a alimentação, o GND comum, os GPIOs utilizados e o programa.

2. Materiais utilizados

Componente	Utilização
ESP32	Executar os programas e controlar os componentes.
Protoboard e jumpers	Realizar as ligações de bancada.
LED e resistor adequado	Fornecer indicação visual.
Botão	Simular um comando manual ou uma entrada digital.
Servomotor SG90	Movimentar o desvio entre duas posições.
Fonte de 5 V adequada	Alimentar o servomotor.

3. Exercício 1 - Indicação luminosa

O primeiro exercício confirma o controle de uma saída digital. Utilize um LED no `GPIO 25`, com resistor adequado, e faça-o piscar a cada segundo.

```
const int ledPin = 25;

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

void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Na ferrovia, esse LED pode representar uma indicação de equipamento ativo ou um aviso visual. Observe se os tempos ligado e desligado possuem aproximadamente a mesma duração.

4. Exercício 2 - Comando por botão

Agora ligue um botão entre o `GPIO 27` e o GND. O botão comandará o LED, transformando uma entrada em uma ação visível.

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

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

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

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

Pressione e solte o botão várias vezes. Confirme que o LED acende enquanto o botão está pressionado e apaga quando ele é solto.

> O botão representa uma entrada simples. Mais adiante, um sensor poderá fornecer ao programa uma informação semelhante: presença ou passagem de uma composição.

5. Exercício 3 - Temporização sem bloquear o programa

Substitua o uso de `delay()` por `millis()`. Assim, a ESP32 poderá continuar lendo o botão enquanto controla o tempo do LED.

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

bool ledLigado = false;
unsigned long tempoAnterior = 0;
const unsigned long intervalo = 1000;

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

void loop() {
  unsigned long tempoAtual = millis();

  if (tempoAtual - tempoAnterior >= intervalo) {
    tempoAnterior = tempoAtual;
    ledLigado = !ledLigado;
    digitalWrite(ledPin, ledLigado);
  }

  int estadoBotao = digitalRead(botaoPin);

  if (estadoBotao == LOW) {
    // O botão continua sendo verificado enquanto o LED pisca.
  }
}
```

Altere o valor de `intervalo` para 500 e depois para 2000. Observe como a velocidade da indicação muda sem impedir a leitura contínua da entrada.

6. Exercício 4 - Movimento do desvio

Ligue o sinal do SG90 ao `GPIO 18`. Alimente o servo com uma fonte de 5 V adequada e una o GND dessa fonte ao GND da ESP32.

```
#include <ESP32Servo.h>

Servo desvio;

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

const int posicaoDI = 67;
const int posicaoAC = 97;

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

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

  if (estadoBotao == LOW) {
    desvio.write(posicaoAC);
  } else {
    desvio.write(posicaoDI);
  }
}
```

Faça primeiro o teste em bancada. Depois, com a haste ligada ao desvio, ajuste cuidadosamente os valores de `posicaoDI` e `posicaoAC` para obter encosto suave das agulhas.

7. Exercício 5 - Botão, desvio e indicação visual

Neste exercício final, o botão comandará o desvio e o LED indicará a posição AC. Com o botão solto, o desvio permanecerá em DI e o LED ficará apagado. Ao pressionar o botão, o desvio será comandado para AC e o LED acenderá.

```
#include <ESP32Servo.h>

Servo desvio;

const int servoPin = 18;
const int botaoPin = 27;
const int ledPin = 25;

const int posicaoDI = 67;
const int posicaoAC = 97;

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

  desvio.attach(servoPin);
  desvio.write(posicaoDI);
  digitalWrite(ledPin, LOW);
}

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

  if (estadoBotao == LOW) {
    desvio.write(posicaoAC);
    digitalWrite(ledPin, HIGH);
  } else {
    desvio.write(posicaoDI);
    digitalWrite(ledPin, LOW);
  }
}
```

8. Entendendo a sequência completa

Etapa	Ação
1. Entrada	O botão fornece um estado ao GPIO 27.
2. Leitura	`digitalRead()` lê o estado do botão.
3. Decisão	`if` e `else` escolhem a ação correspondente.
4. Movimento	`write()` comanda o servo para DI ou AC.
5. Indicação	`digitalWrite()` apaga ou acende o LED.

```
botão → leitura → decisão → posição do desvio → indicação visual
```

9. O que deve ser observado

Durante os testes, não observe apenas se o servo se movimenta. Verifique todo o comportamento do conjunto.

- O desvio inicia na posição DI?
- O botão é lido corretamente?
- O servo se desloca no sentido esperado?
- As agulhas encostam sem impacto?
- O servo deixa de forçar o mecanismo após o movimento?
- O LED corresponde à posição comandada?
- O sistema retorna a DI quando o botão é solto?

> Neste estágio, o LED informa a posição comandada pelo programa. Ele não confirma mecanicamente que as agulhas chegaram à posição desejada.

10. Registro dos resultados

Para cada desvio testado, anote os valores e o comportamento observado. Esse registro ajudará na calibração individual dos servos durante a instalação definitiva.

Informação	Registro
Identificação do desvio	Exemplo: D1
GPIO do servo	Exemplo: 18
Valor da posição DI	Valor ajustado no teste
Valor da posição AC	Valor ajustado no teste
Sentido do movimento	Correto ou invertido
Encosto das agulhas	Suave, insuficiente ou com esforço

DESAFIO PRÁTICO FINAL

Monte e teste o programa completo do Exercício 5. Depois, faça as seguintes alterações, uma de cada vez:

1. Ajuste os valores das posições DI e AC de acordo com o desvio utilizado.
2. Inverta a lógica do LED para que ele acenda em DI e apague em AC.
3. Troque o botão por uma entrada de sensor já testada em bancada.
4. Explique, com suas próprias palavras, o caminho entre a informação recebida e a ação executada.
5. Registre os valores finais de calibração do desvio.

Realize cada modificação separadamente. Assim, será possível compreender o efeito de cada mudança e retornar ao programa anterior caso o resultado não seja o esperado.

Resumo do capítulo

- Uma automação pode ser construída e testada em pequenas etapas.
- Entradas fornecem informações para a ESP32.
- O programa interpreta as informações e toma decisões.
- Saídas transformam as decisões em indicações ou movimentos.
- `millis()` permite controlar o tempo sem interromper outras verificações.
- O botão pode ser substituído por um sensor quando sua ligação estiver corretamente definida.
- O servo movimenta o desvio entre as posições DI e AC.
- Os valores de cada servo precisam ser calibrados individualmente.
- A posição comandada não é o mesmo que uma confirmação mecânica da posição.
- Registrar os testes facilita a futura instalação dos desvios na ferrovia.

Conclusão do Curso Básico: você já possui os fundamentos necessários para iniciar a integração progressiva das funções de automação da Ferrovia EFMN 74.