Curso Básico

Capítulo 3 - Estrutura básica de um programa em C++

Agora vamos escrever o primeiro programa do curso e entender como a ESP32 organiza as instruções que recebe.

Introdução	Todo programa usado na ESP32 possui uma estrutura. Neste capítulo, vamos conhecer as duas partes principais dessa estrutura: **setup()** e **loop()**. Também veremos chaves, ponto e vírgula e comentários, sem entrar ainda em comandos mais avançados.
Objetivo	Compreender a estrutura mínima de um programa em C++ usado na Arduino IDE.
Meta de aprendizagem	Ao final, você deverá reconhecer **setup()** e **loop()**, saber onde colocar instruções e conseguir verificar e enviar um programa simples para a ESP32.

1. Todo programa precisa de uma estrutura

A ESP32 executa as instruções na ordem definida pelo programa. Para organizar esse funcionamento, a Arduino IDE trabalha com duas funções fundamentais: **setup()** e **loop()**.

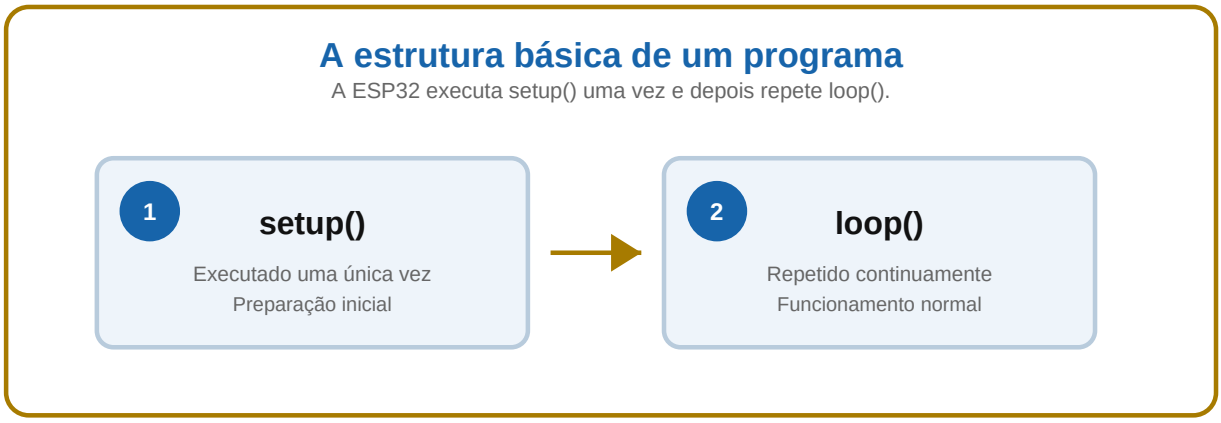

Figura 1 - Fluxo básico: preparação inicial em setup() e repetição contínua em loop().

2. A função setup()

setup() é executada apenas uma vez, logo depois que a ESP32 é ligada ou reiniciada. Ela é usada para preparar o sistema antes do funcionamento normal.

Mais adiante, por exemplo, poderemos usar **setup()** para definir quais pinos receberão sensores, quais controlarão sinais ou como será iniciada uma comunicação.

Por enquanto, basta guardar a regra: **setup() = preparação inicial, executada uma vez.**

3. A função loop()

loop() começa a ser executada depois de **setup()**. Quando chega ao final, a ESP32 volta automaticamente ao início de **loop()** e repete tudo novamente enquanto estiver ligada.

É nessa parte que, futuramente, o programa poderá verificar sensores, conferir posições de desvios, comandar sinais e tomar decisões continuamente.

Regra principal: **loop() = funcionamento contínuo, repetido sem parar.**

4. Chaves { }

As chaves indicam onde começa e onde termina um bloco de instruções. Tudo o que pertence a **setup()** fica entre suas chaves; o mesmo acontece com **loop()**. Sempre que abrir uma chave **{**, deve existir uma chave de fechamento **}** correspondente.

5. Ponto e vírgula ;

Muitas instruções em C++ terminam com ponto e vírgula. Ele indica que aquela instrução terminou. As linhas que apenas abrem uma função, como **void setup() {**, não recebem ponto e vírgula ao final.

6. Comentários

Comentários são anotações escritas no código para ajudar quem estiver lendo o programa. A ESP32 ignora essas linhas. Um comentário de uma única linha começa com ***//***.

Estrutura mínima do programa

```
void setup() {
  // Esta parte será executada uma vez.
}

void loop() {
  // Esta parte será repetida continuamente.
}
```

Figura 2 - Primeiro programa do curso: a estrutura mínima aceita pela Arduino IDE.

7. O que significa void?

Neste momento, não precisamos estudar todos os detalhes da palavra **void**. Basta saber que ela faz parte da forma como **setup()** e **loop()** são declaradas. Mais adiante, quando estudarmos funções, voltaremos a esse assunto.

O importante neste capítulo é reconhecer visualmente a estrutura e não decorar explicações complexas antes da hora.

8. Verificar, enviar e executar

Digite o programa exatamente como mostrado na Figura 2. Depois utilize **Verificar/Compilar**. Se não houver erros, conecte a ESP32, confirme a placa e a porta e utilize **Enviar/Upload**.

Mesmo sem acionar nenhum componente, esse primeiro programa já é válido. Após o envio, a ESP32 executa **setup()** uma vez e passa a repetir **loop()**.

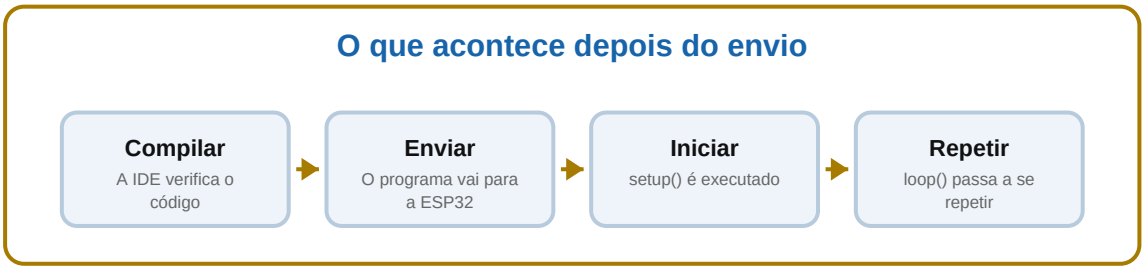

Figura 3 - Sequência básica desde a compilação até a execução contínua na ESP32.

9. Erros simples que podem aparecer

Nesta etapa, confira primeiro os erros mais simples: chave **}** esquecida, parêntese apagado, escrita incorreta de **setup** ou **loop**, caracteres extras e placa ou porta selecionadas incorretamente.

EXERCÍCIO PRÁTICO
Meta: criar, verificar e enviar à ESP32 o primeiro programa do curso.
1. Abra a Arduino IDE. 2. Crie um novo programa. 3. Digite a estrutura mínima mostrada na Figura 2. 4. Salve como **basico-03-primeiro-programa**. 5. Clique em Verificar/Compilar. 6. Conecte a ESP32 e confirme placa e porta. 7. Clique em Enviar/Upload.
Resultado esperado: a Arduino IDE deve compilar o programa sem erros e transferi-lo para a ESP32.

Resumo do capítulo

• **setup()** é executada uma vez quando a ESP32 inicia.
• **loop()** é repetida continuamente.
• Chaves delimitam blocos; comentários iniciados por **//** documentam o código.
• O primeiro exercício permite compilar e enviar um programa válido à ESP32.

Próximo capítulo: Variáveis, constantes e tipos de dados.