← Voltar ao curso

CURSO BÁSICO

Capítulo 6 - Temporização: delay() e millis()

Agora vamos controlar o tempo das ações da ferrovia sem obrigar a ESP32 a ficar parada esperando.

Introdução	Em uma ferrovia automatizada, muitas ações precisam acontecer no momento certo. Uma luz pode permanecer acesa por alguns segundos, um aviso sonoro pode ocorrer antes da partida e um desvio pode precisar de um pequeno intervalo para completar seu movimento. Neste capítulo veremos duas formas de trabalhar com tempo: `delay()` e `millis()`.
Objetivo	Compreender como criar intervalos de tempo no programa e perceber a diferença entre uma espera que interrompe o fluxo e uma temporização que permite à ESP32 continuar executando outras tarefas.
Meta de aprendizagem	Ao final, você deverá ser capaz de usar `delay()` em testes simples e criar uma temporização com `millis()` para executar uma ação periódica sem bloquear o programa.

1. O tempo dentro do programa

Quando a ESP32 executa um programa, as instruções acontecem muito rapidamente. Em várias situações precisamos introduzir um intervalo entre uma ação e outra.

Por exemplo: acender uma luz, aguardar um determinado tempo e depois apagá-la.

2. A função delay()

A função `delay()` cria uma espera. O valor informado é dado em milissegundos.

> `delay(1000);` significa esperar 1 segundo.
>
> `delay(5000);` significa esperar 5 segundos.

Exemplo:

```
digitalWrite(25, HIGH);
delay(1000);

digitalWrite(25, LOW);
delay(1000);
```

Nesse exemplo, a saída fica ligada durante um segundo e depois desligada durante outro segundo.

3. A limitação do delay()

O `delay()` é muito útil para os primeiros testes. Porém, enquanto essa espera acontece, o programa fica parado naquele ponto.

> Se o programa executar `delay(5000)`, ele ficará cinco segundos aguardando antes de continuar.

Em uma ferrovia maior isso pode ser inconveniente, pois durante esse tempo a ESP32 pode precisar ler sensores, atualizar sinais, controlar iluminação ou executar outras ações.

4. A função millis()

A função `millis()` informa quantos milissegundos se passaram desde que a ESP32 começou a executar o programa.

```
unsigned long tempoAtual = millis();
```

Esse valor pode ser guardado em uma variável e comparado com um valor anterior.

5. Criando um intervalo sem parar o programa

```
unsigned long tempoAnterior = 0;
const unsigned long intervalo = 1000;

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

  if (tempoAtual - tempoAnterior >= intervalo) {
    tempoAnterior = tempoAtual;

    // ação que deve ocorrer a cada intervalo
  }
}
```

O programa verifica continuamente se o intervalo já terminou. Enquanto isso, ele pode continuar executando outras instruções.

6. Exemplo prático: piscar uma saída sem delay()

```
const int ledPin = 25;

bool ledLigado = false;

unsigned long tempoAnterior = 0;
const unsigned long intervalo = 1000;

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

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

  if (tempoAtual - tempoAnterior >= intervalo) {
    tempoAnterior = tempoAtual;

    ledLigado = !ledLigado;
    digitalWrite(ledPin, ledLigado);
  }

  // Outras tarefas podem ser executadas aqui.
}
```

Não existe `delay()` nesse programa. A ESP32 verifica o tempo, altera a saída quando necessário e continua livre para executar outras tarefas.

7. Aplicação na Ferrovia EFMN 74

Na Ferrovia EFMN 74 haverá situações em que várias ações precisarão ocorrer ao mesmo tempo ou em sequência. Um exemplo é a partida de uma composição na estação.

> O sistema pode iniciar um aviso sonoro, contar o intervalo programado e depois permitir a próxima ação, enquanto continua verificando sensores e outras condições da ferrovia.

Esse tipo de temporização será importante à medida que a automação crescer e várias funções passarem a trabalhar simultaneamente.

EXERCÍCIO PRÁTICO

Use o exemplo com `millis()` e altere o intervalo para `2000` milissegundos.

1. Compile o programa.
2. Envie para a ESP32.
3. Observe se a saída muda aproximadamente a cada dois segundos.

4. Depois altere o intervalo para `500` milissegundos e compare o resultado.

O exercício deve ser feito sem utilizar `delay()`.

Resumo do capítulo

- `delay()` cria uma espera simples.
- Durante essa espera, o fluxo do programa fica bloqueado naquele ponto.
- `millis()` permite consultar o tempo decorrido.
- Comparando o tempo atual com um tempo anterior, podemos criar intervalos sem parar o programa.
- Essa técnica será útil quando a ESP32 precisar executar várias tarefas na ferrovia.

Próximo capítulo: LEDs e sinalização visual na ferrovia.

← Voltar ao curso