Controlo programado com Arduino

Prática 2: O piscar alternado (Variáveis e Controlo Sequencial)

Na prática anterior aprendemos a manter um único LED aceso de forma contínua. Nesta sessão daremos um passo em frente introduzindo un segundo LED para criar um efeito de piscar alternado (como as luzes de uma passagem de nível ferroviária ou de um veículo de emergência). Para conseguir um código organizado e escalável, aprenderás a declarar e a inicializar múltiplas variáveis independentes.

Objetivos de Aprendizagem

Ao finalizar esta sessão, serás capaz de:

- Declarar e utilizar múltiplas variáveis do tipo inteiro (`int`) no teu código.
- Desenhar sequências temporais lógicas e organizadas utilizando a instrução `delay` .
- Compreender o comportamento físico de saídas independentes ligadas em paralelo.
- Otimizar a manutenção do teu programa através do uso de nomes de variáveis descritivos.

Componentes Necesários

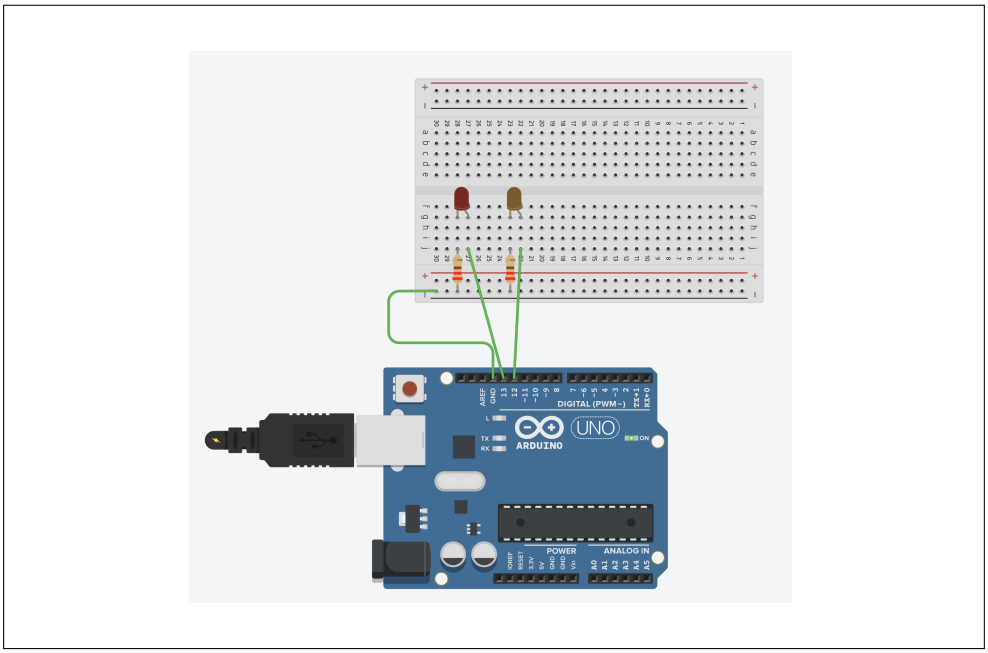
Procura e coloca os seguintes componentes na tua mesa de trabalho do TinkerCad:

- 1 Placa Arduino Uno.
- 1 Placa de ensaio pequena (Protoboard).
- 2 Díodos LED (recomendamos que uses duas cores diferentes, por exemplo: Verde e Vermelho).
- 2 Resistências de $220\,\Omega$ (uma para proteger cada LED).
- Cabos de ligação virtuais.

Esquema de Montagem na Protoboard

Adicionaremos un segundo LED ao circuito que montaste na Prática 1. Certifica-te de ligar cada ânodo a um pino digital de controlo independente.

Configuração Inicial do objeto Terminal



Código Base: Declaração de Variáveis e Alternância

Abre o editor de código no TinkerCad em modo texto. Apaga o programa anterior e escreve o seguinte código sequencial. Analisa como o estado de cada saída digital se coordena de forma síncrona no tempo.

```
// 1. DECLARAÇÃO DE VARIÁVEIS
// Reservamos espaços de memória para guardar os pinos dos LEDs
int ledVermelho = 13;
int ledVerde = 12;

void setup() {
  // Configuramos ambos os pinos como saídas de corrente
  pinMode(ledVermelho, OUTPUT);
  pinMode(ledVerde, OUTPUT);
}

void loop() {
  // ESTADO 1: Vermelho aceso e Verde apagado
  digitalWrite(ledVermelho, HIGH);
  digitalWrite(ledVerde, LOW);
  delay(1000); // Pausa de 1 segundo neste estado

  // ESTADO 2: Vermelho apagado e Verde aceso
  digitalWrite(ledVermelho, LOW);
  digitalWrite(ledVerde, HIGH);
  delay(1000); // Pausa de 1 segundo neste estado
}
```

Listing 1: Código para controlar o piscar alternado de dois LEDs.

Como funciona o código?

Neste programa, o Arduino executa as instruções de forma estritamente linear dentro da função `loop` :

1. **Variáveis globais**: Declaramos `int` e `int` no início. Isto permite-nos usar palavras compreensíveis em vez de escrever números fixos (*hardcoded*) ao longo do código.
2. **Alternância síncrona**: Para conseguir que um LED se apague enquanto o outro se acende, enviamos sinais opostos de forma simultânea. No Estado 1 aplicamos `HIGH` ao vermelho e `LOW` ao verde. Após um segundo de pausa, invertemos os valores no Estado 2.
3. **Loop contínuo**: Ao terminar o Estado 2, o fluxo volta de imediato ao início do `loop` , acendendo o vermelho e apagando o verde, criando assim o ciclo infinito de alternância.

Importância da ordem lógica

Se não apagássemos explicitamente o LED contrário ao mudar de estado (ou seja, omitindo as instruções `LOW`), o programa simplesmente acenderia ambos os LEDs sequencialmente e deixá-los-ia acesos de forma indefinida, perdendo o efeito de piscar alternado.

O Desafio da Prática 2

Verifica se compreendes o fluxo temporal e o uso de variáveis resolvendo estas três propostas no teu simulador:

1. **Efeito Policial (Piscar Estroboscópico)**: Modifica os tempos dos atrasos (`delay()`) para conseguir uma alternância muito rápida que emule as luzes de emergência de uma patrulha (por exemplo, 150 ms de pausa).
2. **O poder de las variáveis**: Muda o cabo do LED Verde para o **pino digital 8** no TinkerCad. Realiza uma única alteração no teu código (aproveitando as variáveis) para que o teu programa continue a funcionar perfeitamente sem alterar o `setup()` nem o `loop()`.
3. **Fase de Apagado Completo**: Modifica o programa para que, antes de cada mudança de luz, exista uma breve fase intermédia onde os dois LEDs permaneçam apagados ao mesmo tempo durante meio segundo (500 ms).