Control programado con Arduino

Práctica 9: Modularización con Funcións (Paso de Parámetros)

A medida que os nosos proxectos medran en tamaño e complexidade, escribir todo o código seguido dentro da función `loop` fai que sexa difícil de ler, depurar e manter. Nesta práctica daremos o salto á programación estruturada. Aprenderás a dividir o teu programa en bloques independentes chamados **funcións**, deseñando o teu propio patrón de parpadeo personalizado mediante o paso de parámetros físicos (pin de saída e tempo de retardo).

Obxectivos de Aprendizaxe

Ao rematar esta sesión serás capaz de:

- Comprender o concepto de modularización e a estrutura dunha función en C++.
- Declarar e definir funcións personalizadas con tipo de retorno baleiro (`void`).
- Implementar o paso de parámetros (`int` , `int`) para flexibilizar o comportamento das funcións.
- Chamar a funcións personalizadas desde o bucle principal `loop` de forma limpa e organizada.

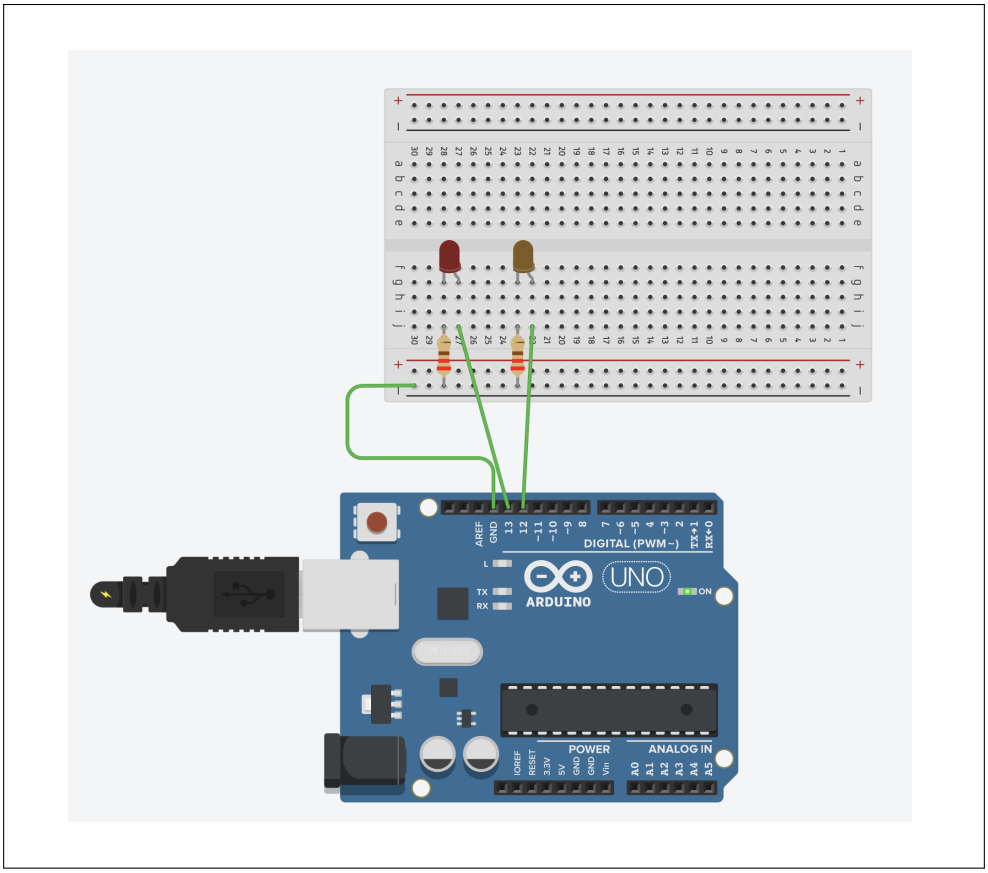
Compoñentes Necesarios

Para simplificar a parte física desta práctica e centrarnos de cheo na aprendizaxe del código, recuperaremos e ampliaremos o circuíto con dous diodos de saída da Práctica 2.

- 1 Placa Arduino Uno.
- 1 Placa de prototipado pequena (Protoboard).
- 2 Diodos LED (suxerímosche usar 1 Verde e 1 Vermello).
- 2 Resistencias de $220\,\Omega$ (unha para protexer cada LED).
- Cables de conexión virtuais.

Esquema de montaxe na Protoboard

Asociaremos cada cor a un pin dixital secuencial do Arduino para facilitar o direccionamento por software.



Código Base: Declaración, Parámetros e Chamada

Escribe o seguinte programa no teu editor de TinkerCad. Observa que agora a función `loop` é extremadamente curta e fácil de ler, xa que o traballo pesado delegouse nunha nova función chamada .

```
// Practica 9: Modularizacion dun patron de parpadeo con funcións
int ledVerde = 13;
int ledRojo = 12;

void setup() {
  pinMode(ledVerde, OUTPUT);
  pinMode(ledRojo, OUTPUT);
}

void loop() {
  // Chamamos á nosa función personalizada indicando que pin e que tempo usar
  parpadearLed(ledVerde, 1000); // Parpadea o led verde cada 1 segundo
  parpadearLed(ledRojo, 500);   // Parpadea o led vermello cada 500 milisegundos
  parpadearLed(ledVerde, 200);  // Parpadea o led verde moi rapido (200 ms)
}

// 1. DEFINICION DA FUNCIÓN PERSONALIZADA
// 'void' significa que a función non devolve ningún valor ao terminar.
// Entre parénteses definimos dúas variables locais (parámetros de entrada).
void parpadearLed(int pinLed, int tiempoRetardo) {
  digitalWrite(pinLed, HIGH);  // Acende o LED indicado pelo parámetro
  delay(tiempoRetardo);        // Espera o tempo indicado polo parámetro
  digitalWrite(pinLed, LOW);   // Apaga o LED indicado
  delay(tiempoRetardo);        // Espera o tempo indicado
}
```

Listing 1: Código utilizando unha función personalizada con paso de parámetros.

Como funciona o código?

Unha función é un subprograma que agrupa un conxunto de instrucións baixo un nome único para realizar uma tarefa específica.

- **Estrutura da cabeceira**: `void` `int` `int`
  - `void`: É o tipo de datos que devolve a función. Neste caso non devolve nada, só executa accións, polo que usamos a palabra clave `void` (baleiro).
  - : É o nome identificador que eliximos para a nosa función.
  - e : Son os **parámetros de entrada**. Actúan como variables locais dentro da función. Reciben os seus valores no mesmo instante no que chamamos á función desde o bucle principal.
- **A chamada á función**: Cando o programa le , detense momentaneamente na función `loop` , copia o valor de (13) dentro de e o valor 1000 dentro de , e salta a executar as instrucións da función personalizada. Ao terminar, regresa de inmediato á seguinte liña de `loop` .

O Reto da Práctica 9

Pon á proba os teus dotes de deseño de software modular superando as seguintes propostas de programación:

1. **Función de Acendido Simultáneo**: Deseña unha nova función personalizada chamada `encenderAmbos(int pinA, int pinB, int tiempo)` que acenda os dous LEDs á vez, espera o tempo indicado por parámetro, os apague á vez, e volva esperar ese mesmo tempo. Próbaa chamándoa desde o teu `loop()`.
2. **Función de Código Morse**: Deseña unha función especializada chamada `emitirSOS(int pinLed)` que, ao recibir un pin de saída, reproduza nese LED a secuencia clásica de socorro en código Morse (S.O.S): 3 parpadeos moi curtos, 3 parpadeos longos e 3 parpadeos moi curtos.
3. *Pregunta para reflexionar*: Se en lugar de usar unha función personalizada quixeses facer parpadear os dous LEDs con tres ritmos de tempo diferentes (como no `loop()` base), cantas liñas de código terías que escribir de forma repetitiva? Que vantaxes achega modularizar en termos de limpeza e prevención de erros tipográficos?