Control programado con Arduino

Práctica 7: O semáforo interactivo (Petición de paso)

Ata agora programamos sistemas que funcionaban de forma illada: ou ben respondían inmediatamente a un botón, o ben seguían unha secuencia fixa no tempo. Hoxe uniremos ambos mundos para deseñar un semáforo interactivo. Por defecto, os vehículos terán sempre vía libre (luz verde continua), pero cando un peón prema o pulsador, o Arduino interromperá o fluxo normal para deter o tráfico e permitir un cruzamento seguro.

Obxectivos de Aprendizaxe

[1pt]

Ao rematar esta sesión serás capaz de:

- Integrar compoñentes de entrada (pulsador) e de saída (LEDs) nun mesmo sistema coordinado.
- Deseñar un algoritmo baseado en estados onde o programa "espera de forma activa" unha acción do usuario.
- Analizar criticamente o comportamento dun microcontrolador cando se combinan lecturas de sensores con pausas de tempo de longa duración (`delay`).
- Desenvolver solucións lógicas para simular prioridades de paso en contornas automatizadas urbanas.

Compoñentes Necesarios

[1pt]

Busca e coloca os seguintes compoñentes na túa mesa de traballo de TinkerCad:

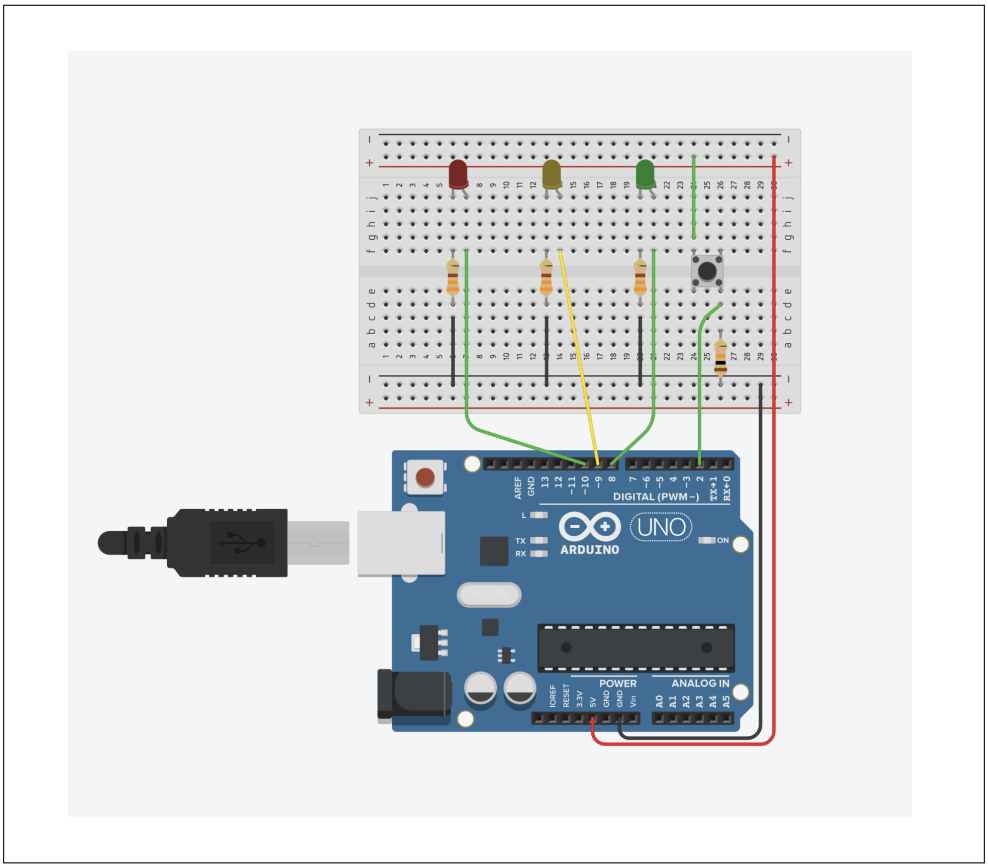
- 1 Placa Arduino Uno.
- 1 Placa de prototipado pequena (Protoboard).
- 3 Diodos LED (1 Vermello, 1 Amarelo, 1 Verde).
- 3 Resistencias de $220\,\Omega$ (para protexer os LEDs).

- 1 Pulsador de catro patillas.
- 1 Resistencia de $10\,\mathrm{k}\Omega$ (para a configuración Pull-Down do pulsador).
- Cables de conexión virtuais.

Esquema de montaxe na Protoboard

[1pt]

Este circuíto é a combinación exacta das montaxes da Práctica 3 e da Práctica 5. Asegúrate de cablear con orde para evitar cruces!



Código Base: Espera Activa do Fluxo

[1pt]

Copia o seguinte programa no editor de TinkerCad. Observa que na función `loop` non hai ningún retardo de tempo inicial; o Arduino limítase a vixiar o botón á máxima velocidade posible.

```
// Definicion de pins de saida para os LEDs
int ledRojo = 10;
int ledAmarillo = 9;
int ledVerde = 8;

// Definicion do pin de entrada para o boton
int botonPeaton = 2;

void setup() {
  pinMode(ledRojo, OUTPUT);
  pinMode(ledAmarillo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(botonPeaton, INPUT);

  // Estado inicial por defecto: via libre para os coches
  digitalWrite(ledVerde, HIGH);
  digitalWrite(ledAmarillo, LOW);
  digitalWrite(ledRojo, LOW);
}

void loop() {
  // Lemos continuamente o estado do boton
  int peticionPaso = digitalRead(botonPeaton);

  // Se un peon prema o boton, iniciamos a secuencia de parada
  if (peticionPaso == HIGH) {
    delay(1000); // Pequeno tempo de cortesia antes de reaccionar

    // 1. Pasar de Verde a Amarelo
    digitalWrite(ledVerde, LOW);
    digitalWrite(ledAmarillo, HIGH);
    delay(2000); // 2 segundos en amarelo de advertencia

    // 2. Pasar de Amarelo a Vermello (Coches detidos / Peons cruzan)
    digitalWrite(ledAmarillo, LOW);
    digitalWrite(ledRojo, HIGH);
    delay(5000); // 5 segundos para que os peons crucen seguros

    // 3. Volver ao estado de via libre (Coches avanzan)
    digitalWrite(ledRojo, LOW);
    digitalWrite(ledVerde, HIGH);
  }
}
```

Listing 1: Código para un semáforo interactivo con petición de paso.

Como funciona o código?

[1pt]

Neste programa, o segredo está na estrutura da función `loop` :

- **Espera sen bloqueos**: Mentres ninguén pulse o botón, o microcontrolador le a liña `int digitalRead`, comproba que dá `LOW`, ignora o bloque `if` e volve empezar o bucle. Isto sucede miles de veces por segundo, polo que o LED verde mantense acendido sen pausa ningunha.
- **O problema da insensibilidade temporal**: Unha vez que o peón pulsa o botón e entramos dentro del bloque `if`, o fluxo secuencial toma o control e execútanse as ordes `delay` e `delay`. Durante eses 7 segundos totais, o Arduino está "conxelado" executando as pausas. Se outro peón volvese pulsar o botón nese intervalo, o Arduino non se decataría, xa que non pode executar a liña `digitalRead` mentres estea durmido por un `delay` .

O Reto da Práctica 7

Pon á proba as túas habilidades resolvendo estas dúas melloras lóxicas de enxeñaría urbana:

1. **Tempo de Cortesía de Tráfico (Cool-Down)**: Nunha cidade real, se un peón pulsa o botón, cruza e, xusto ao terminar, outro peón volve pulsar o botón, o tráfico colapsaríase. Modifica o código para engadir un tempo de espera obrigatorio de **5 segundos** ao final do ciclo (despois de volver acender o verde) durante o cal o sistema ignore por completo calquera pulsación do botón.
2. **Aviso acústico ou visual de fin de paso**: Engade un parpadeo rápido no LED vermello durante o último segundo da súa activación para avisar de forma visual aos peóns que o seu tempo de paso está a piques de esgotarse.
3. *Pregunta para reflexionar*: Como resolverías o problema da insensibilidade do botón sen recorrer a hardware complexo? Se dividises un atraso longo como `delay(5000)` en pequenos pasos de `delay(100)`, como podería axudar isto a comprobar o estado del botón máis a miúdo?