Control programado con Arduino

Práctica 5: O semáforo básico (Control secuencial avanzado)

Nas prácticas anteriores aprendemos a interactuar con botóns e a rexistrar estados lóxicos. Hoxe aplicaremos un enfoque de enxeñaría clásica para resolver un sistema secuencial coordinado no tempo: un semáforo de tráfico para vehículos. Conectaremos tres LEDs (Vermello, Amarelo e Verde) e programaremos o seu comportamento temporal simulando o ciclo real das rúas da nosa cidade.

Obxectivos de Aprendizaxe

[1pt]

Ao rematar esta sesión serás capaz de:

- Coordinar múltiples canles de saída dixital de forma síncrona e secuencial.
- Estruturar tempos de retardo diferenciados segundo a importancia ou perigo do estado físico controlado.
- Organizar un algoritmo secuencial en bloque pechado evitando estados indefinidos ou cruces perigosos de sinais.
- Practicar a declaración múltiple de variables de asignación de pins hardware.

Compoñentes Necesarios

[1pt]

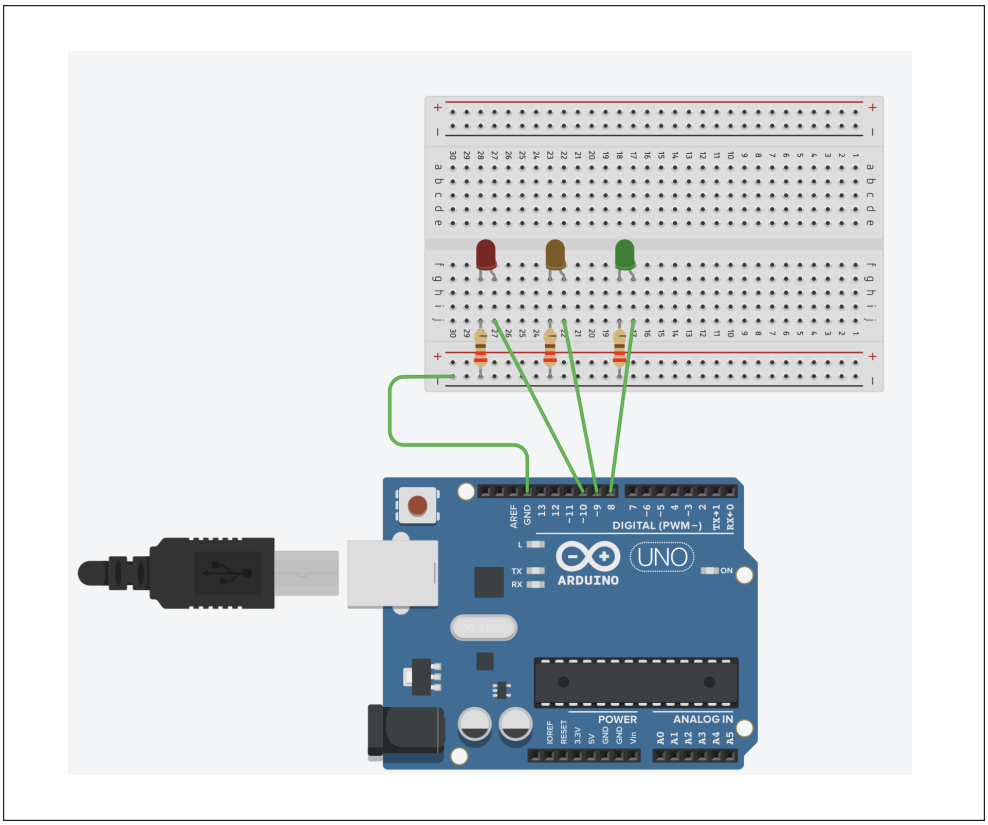
Busca e coloca os seguintes compoñentes na túa mesa de traballo de TinkerCad:

- 1 Placa Arduino Uno.
- 1 Placa de prototipado pequena (Protoboard).
- 3 Diodos LED (cores obrigatorias: 1 Vermello, 1 Amarelo e 1 Verde).
- 3 Resistencias de $220\,\Omega$ (unha para cada LED).
- Cables de conexión virtuais.

Esquema de montaxe na Protoboard

[1pt]

Para controlar o semáforo de vehículos, asociaremos cada cor a un pin dixital secuencial do Arduino.



Código Base: Secuencia Temporal Estrita

[1pt]

Escribe o seguinte programa no teu editor de texto de TinkerCad. Observa con atención como en cada estado do semáforo nos aseguramos de apagar de forma explícita os LEDs que non corresponden para evitar acendidos simultáneos accidentais.

```
// Declaracion de pins fisicos para os LEDs do semaforo
int ledRojo = 10;
int ledAmarillo = 9;
int ledVerde = 8;

void setup() {
  // Configurar os tres pins como canles de saida
  pinMode(ledRojo, OUTPUT);
  pinMode(ledAmarillo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
}

void loop() {
  // ESTADO 1: Verde activo (Paso libre de vehiculos)
  digitalWrite(ledVerde, HIGH);
  digitalWrite(ledAmarillo, LOW);
  digitalWrite(ledRojo, LOW);
  delay(5000); // O verde dura 5 segundos activo

  // ESTADO 2: Amarelo activo (Aviso de parada inminente)
  digitalWrite(ledVerde, LOW);
  digitalWrite(ledAmarillo, HIGH);
  digitalWrite(ledRojo, LOW);
  delay(2000); // O amarelo dura so 2 segundos

  // ESTADO 3: Vermello activo (Vehiculos detidos)
  digitalWrite(ledVerde, LOW);
  digitalWrite(ledAmarillo, LOW);
  digitalWrite(ledRojo, HIGH);
  delay(5000); // O vermello dura 5 segundos activo
}
```

Listing 1: Código para a temporización dun semáforo de vehículos.

Como funciona o código?

[1pt]

O fluxo deste programa é puramente secuencial e lineal. Ao estar dentro da función `loop` , repítese infinitamente nesta orde exacta:

1. Acéndese o Verde e apáganse os demais $\rightarrow$ Espera de 5 segundos.
2. Apágase o Verde, acéndese o Amarelo e apágase o Vermello $\rightarrow$ Espera de 2 segundos.

3. Apáganse os dous anteriores, acéndese o Vermello $\rightarrow$ Espera de 5 segundos.
4. O ciclo remata e volve empezar inmediatamente no paso 1 (Verde).

Seguridade Eléctrica e Lóxica

En sistemas industriais reais, un erro de software xamais debe permitir que dous luces contraditorias (coma o Verde e o Vermello) se acendan xuntas. Por iso, en cada cambio de fase apagamos explicitamente as saídas inactivas mediante instrucións `digitalWrite(pin, LOW)`.

O Reto da Práctica 5

Mellora o comportamento do semáforo aplicando estas variacións lóxicas sobre o teu código:

1. **Fase de Transición de Seguridade**: En moitos países, antes de pasar do semáforo Vermello ao Verde, acéndense simultaneamente o Vermello e o Amarelo durante un segundo para indicar aos condutores que arranquen os seus motores. Modifica o teu código para introducir esta fase intermedia entre o Estado 3 (Vermello) e o Estado 1 (Verde).
2. **Modo Nocturno (Amarelo Parpadeante)**: Modifica o código para simular o comportamento dun semáforo á media noite: as luces Verde e Vermella deben quedar completamente inactivas, mentres que a luz Amarela debe acenderse e apagarse de xeito continuo cada 500 ms.
3. *Pregunta para reflexionar*: Se quixeses que o semáforo fose interactivo e un peón puidese cambiar o ciclo de luces premendo un botón, que problemas cres que plantexaría o uso da instrución `delay(5000)`? É capaz o Arduino de detectar unha pulsación mentres está executando esa pausa temporal?