Control programado con Arduino

Práctica 5: El Semáforo Básico (Control Secuencial Avanzado)

En las prácticas anteriores aprendimos a interactuar con botones y a registrar estados lógicos. Hoy aplicaremos un enfoque de ingeniería clásica para resolver un sistema secuencial coordinado en el tiempo: un semáforo de tráfico para vehículos. Conectaremos tres LEDs (Rojo, Amarillo y Verde) y programaremos su comportamiento temporal simulando el ciclo real de las calles de nuestra ciudad.

Objetivos de Aprendizaje

Al finalizar esta sesión, serás capaz de:

- Coordinar múltiples canales de salida digital de forma síncrona y secuencial.
- Estructurar tiempos de retardo diferenciados según la importancia o peligro del estado físico controlado.
- Organizar un algoritmo secuencial en bloque cerrado evitando estados indefinidos o cruces peligrosos de señales.
- Practicar la declaración múltiple de variables de asignación de pines hardware.

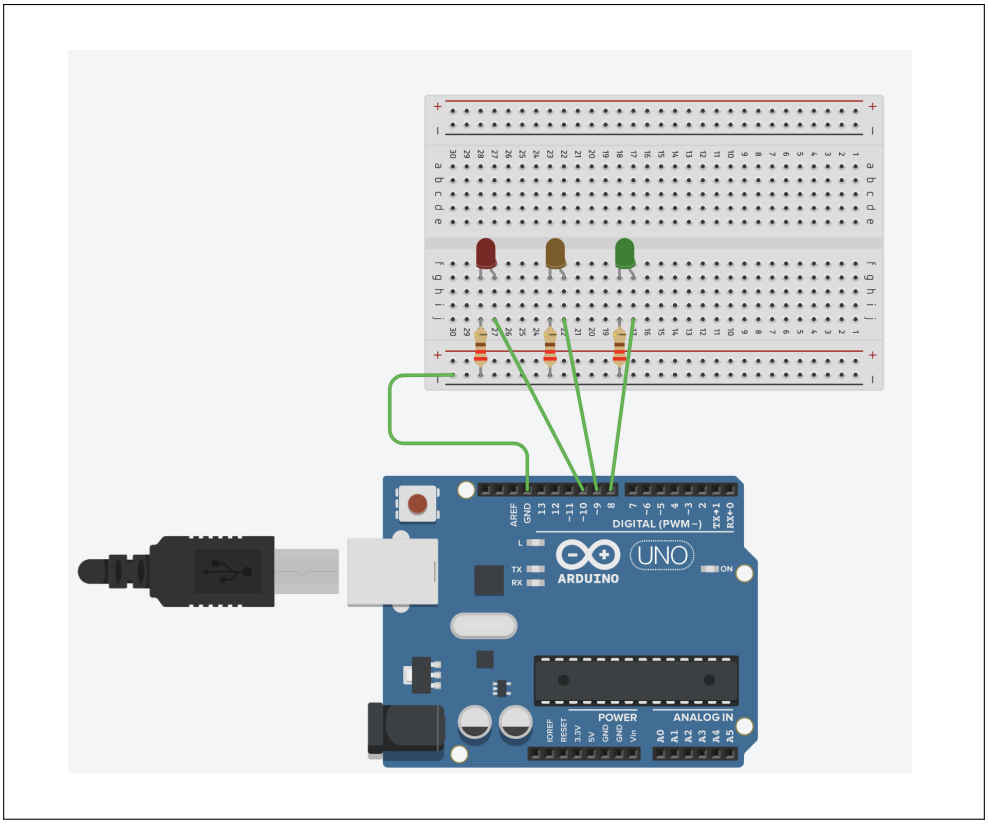
Componentes Necesarios

Busca y coloca los siguientes componentes en tu mesa de trabajo de TinkerCad:

- 1 Placa Arduino Uno.
- 1 Placa de pruebas pequeña (Protoboard).
- 3 Diodos LED (colores obligatorios: 1 Rojo, 1 Amarillo y 1 Verde).
- 3 Resistencias de $220\,\Omega$ (una para cada LED).
- Cables de conexión virtuales.

Esquema de Montaje en la Protoboard

Para controlar el semáforo de vehículos, asociaremos cada color a un pin digital secuencial del Arduino.



Código Base: Secuencia Temporal Estricta

Escribe el siguiente programa en tu editor de texto de TinkerCad. Observa con atención cómo en cada estado del semáforo nos aseguramos de apagar de forma explícita los LEDs que no corresponden para evitar encendidos simultáneos accidentales.

```
// Declaracion de pines fisicos para los LEDs del semaforo
int ledRojo = 10;
int ledAmarillo = 9;
int ledVerde = 8;

void setup() {
  // Configurar los tres pines como canales de salida
  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); // El verde dura 5 segundos activo

  // ESTADO 2: Amarillo activo (Aviso de parada inminente)
  digitalWrite(ledVerde, LOW);
  digitalWrite(ledAmarillo, HIGH);
  digitalWrite(ledRojo, LOW);
  delay(2000); // El amarillo dura solo 2 segundos

  // ESTADO 3: Rojo activo (Vehiculos detenidos)
  digitalWrite(ledVerde, LOW);
  digitalWrite(ledAmarillo, LOW);
  digitalWrite(ledRojo, HIGH);
  delay(5000); // El rojo dura 5 segundos activo
}
```

Listing 1: Código para la temporización de un semáforo de vehículos.

¿Cómo funciona el código?

El flujo de este programa es puramente secuencial y lineal. Al estar dentro de la función `loop` , se repite infinitamente en este orden exacto:

1. Se enciende el Verde y se apagan los demás $\rightarrow$ Espera de 5 segundos.
2. Se apaga el Verde, se enciende el Amarillo y se apaga el Rojo $\rightarrow$ Espera de 2 segundos.
3. Se apagan los dos anteriores, se enciende el Rojo $\rightarrow$ Espera de 5 segundos.
4. El ciclo termina y vuelve a empezar inmediatamente en el paso 1 (Verde).

Seguridad Eléctrica y Lógica

En sistemas industriales reales, un error de software jamás debe permitir que dos luces contradictorias (como el Verde y el Rojo) se enciendan juntas. Por eso, en cada cambio de fase apagamos explícitamente las salidas inactivas mediante instrucciones `digitalWrite pin, LOW` .

El Reto de la Práctica 5

Mejora el comportamiento del semáforo aplicando estas variaciones lógicas sobre tu código:

1. **Fase de Transición de Seguridad**: En muchos países, antes de pasar del semáforo Rojo al Verde, se encienden simultáneamente el Rojo y el Amarillo durante un segundo para indicar a los conductores que arranquen sus motores. Modifica tu código para introducir esta fase intermedia entre el Estado 3 (Rojo) y el Estado 1 (Verde).
2. **Modo Nocturno (Amarillo Parpadeante)**: Modifica el código para simular el comportamiento de un semáforo a medianoche: las luces Verde y Roja deben quedarse completamente inactivas, mientras que la luz Amarilla debe encenderse y apagarse de manera continua cada 500 ms.
3. *Pregunta para reflexionar*: Si quisieras que el semáforo fuera interactivo y un peatón pudiera cambiar el ciclo de luces pulsando un botón, ¿qué problemas crees que plantearía el uso de la instrucción `delay(5000)`? ¿Es capaz el Arduino de detectar una pulsación mientras está ejecutando esa pausa temporal?