Control programado con Arduino

Práctica 10: El Sistema Automatizado con Retorno de Datos

¡Has llegado a la última parada de nuestro viaje de aprendizaje! En esta práctica final integraremos prácticamente todo lo que has aprendido. Diseñaremos un sistema de seguridad inteligente. Para estructurar el código de manera impecable, crearemos una función de diagnóstico que evaluará el estado del pulsador y devolverá un valor lógico (`return`) al bucle principal, decidiendo si activa el relé de potencia y los LEDs de aviso.

Objetivos de Aprendizaje

Al finalizar esta sesión, serás capaz de:

- Diseñar funciones personalizadas que procesen información y devuelvan valores lógicos mediante la instrucción `return`.
- Comprender los tipos de retorno de datos en C++ (en especial, el tipo booleano `bool`).
- Integrar múltiples componentes de entrada y salida (pulsador, LEDs de señalización y relé de potencia) en un único circuito complejo.
- Desarrollar la lógica de un sistema industrial de seguridad de parada de emergencia.

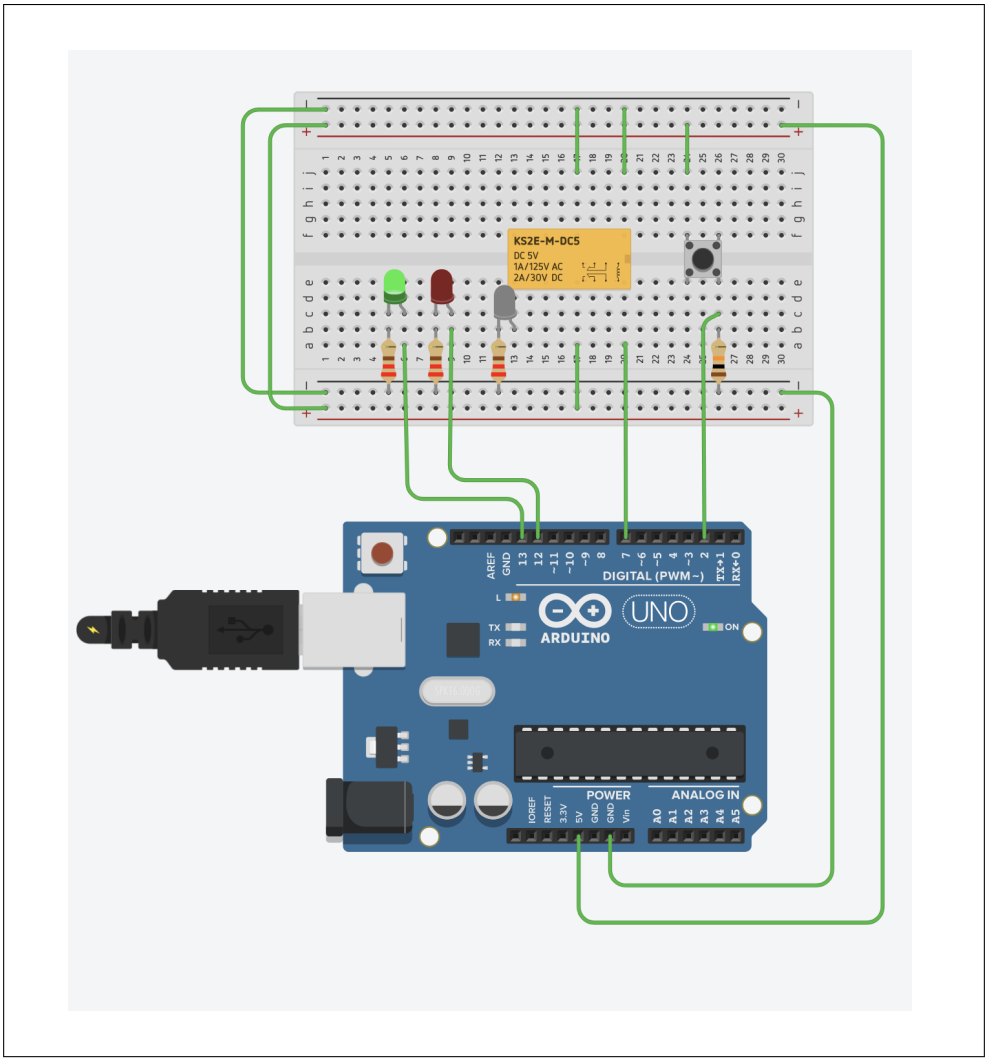
Componentes Necesarios

Este es el montaje más completo del curso. Coloca los siguientes componentes en tu simulador TinkerCad:

- 1 Placa Arduino Uno.
- 1 Placa de pruebas pequeña (Protoboard).
- 2 Diodos LED (1 Verde de ”Sistema Seguro” y 1 Rojo de ”Peligro”).
- 2 Resistencias de $220\,\Omega$ (para proteger los LEDs).
- 1 Pulsador de cuatro patillas (Botón de emergencia).
- 1 Resistencia de $10\,\text{k}\Omega$ (para la configuración Pull-Down del botón).
- 1 Relé SPDT de 5 Voltios (para simular el corte de alimentación de maquinaria de potencia).
- Cables de conexión virtuales.

Esquema de Montaje en la Protoboard

Organiza el cableado con paciencia. Recuerda que la bobina del relé se activa mediante un pin de Arduino, y sus contactos gobiernan la potencia de forma totalmente independiente.



Código Base: Funciones con tipo de retorno 'bool'

Escribe el siguiente programa en TinkerCad. Observa que el tipo de datos de nuestra función ya no es `void`, sino `bool`, lo que significa que nos devolverá un valor de verdadero (`true`) o falso (`false`) utilizando la orden `return`.

```
// Practica 10: Sistema de seguridad industrial con retorno de estado
int pinBoton = 2;
int pinRele = 7;
int ledVerde = 13;
int ledRojo = 12;

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

void loop() {
  // 1. LLAMADA A LA FUNCION: Evaluamos el boton y guardamos el resultado
  bool alarmaActiva = evaluarSeguridad(pinBoton);

  // 2. ACCIONES EN BASE AL RETORNO DE LA FUNCION
  if (alarmaActiva == true) {
    // Si la alarma esta activa, disparamos el rele y encendemos el LED rojo
    digitalWrite(pinRele, HIGH);
    digitalWrite(ledRojo, HIGH);
    digitalWrite(ledVerde, LOW);
  } else {
    // Si todo esta seguro, el rele descansa y el LED verde brilla
    digitalWrite(pinRele, LOW);
    digitalWrite(ledRojo, LOW);
    digitalWrite(ledVerde, HIGH);
  }
}

// 3. DEFINICION DE LA FUNCION CON RETORNO
// Indicamos 'bool' al inicio porque la funcion debe devolver un booleano
bool evaluarSeguridad(int pinSensor) {
  int lectura = digitalRead(pinSensor);

  // Si se detecta una pulsacion (peligro detectado)
  if (lectura == HIGH) {
    return true;  // Devolvemos el valor 'verdadero' al loop()
  } else {
    return false; // Devolvemos el valor 'falso' al loop()
  }
}
```

Listing 1: Código utilizando una función con retorno de datos para evaluar el sistema.

¿Cómo funciona el código y la instrucción `return`?

Hasta ahora, todas nuestras funciones terminaban con la palabra `void` porque solo realizaban acciones físicas. Pero en programación es vital que las funciones realicen cálculos o lecturas y nos "devuelvan" una respuesta.

- **Tipo de retorno (`bool`)**: En la cabecera, sustituimos `void` por `bool`. Esto obliga a la función a devolver un dato booleano (verdadero o falso) antes de cerrarse.
- **La instrucción `return`**: Es la puerta de salida de la función. Envía el valor seleccionado de vuelta a la línea exacta donde se llamó. En cuanto el programa lee `return`, la función se cierra de inmediato.
- **La recepción del dato**: En la línea `bool` , el Arduino ejecuta la función y reemplaza todo ese bloque por el resultado devuelto (`true` o `false`), guardándolo directamente en la variable local .

El Reto de la Práctica 10: ¡El Desafío Final!

Consolida tus conocimientos completando este gran reto de programación estructurada de seguridad:

1. **Alarma Acústica Temporalizada**: Añade un patrón de parpadeo de emergencia. Si la función devuelve `true`, haz que el LED Rojo parpadee rápidamente de forma continua, mientras que el LED Verde se mantenga apagado.
2. **Función de Retorno Numérico**: Diseña una nueva función de diagnóstico llamada `comprobarErrores()` que no devuelva un valor booleano, sino un número entero (`int`). Si el sistema está apagado, debe devolver 0. Si el sistema se activa, debe contar los segundos que permanece activo y devolver ese valor numérico al `loop()` para mostrar el tiempo de retardo en la respuesta.
3. *Pregunta para reflexionar*: ¡Enhorabuena por completar las 10 prácticas! Escribe una breve valoración personal sobre cómo ha evolucionado tu forma de programar desde la Práctica 1 (donde todo era secuencial) hasta esta Práctica 10 (con funciones con paso de parámetros y retornos). ¿Cómo te ayuda esta estructura a pensar como un/a ingeniero/a de software?