

PRÁCTICA DE ARDUINO - CONTROL DE LED CON BOTÓN

NIVEL: **1**

DURACIÓN: 2 horas

OBJETIVO: Los estudiantes aprenderán a controlar un LED mediante un botón utilizando una placa Arduino.

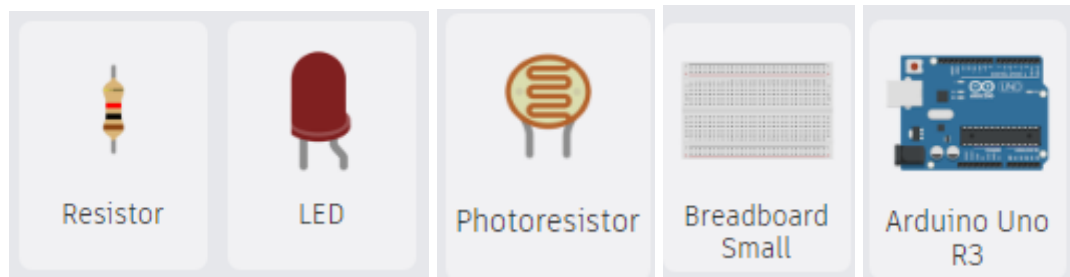
MATERIALES

- Placa Arduino Uno (o similar)
- 1 LED
- 1 Fotorresistor (LDR)
- 1 Resistencia de 10k ohmios (para crear un divisor de voltaje con el LDR)
- 1 Resistencia de 220 ohmios (para el LED)
- Protoboard y cables de conexión
- [ThinkerCad](#)

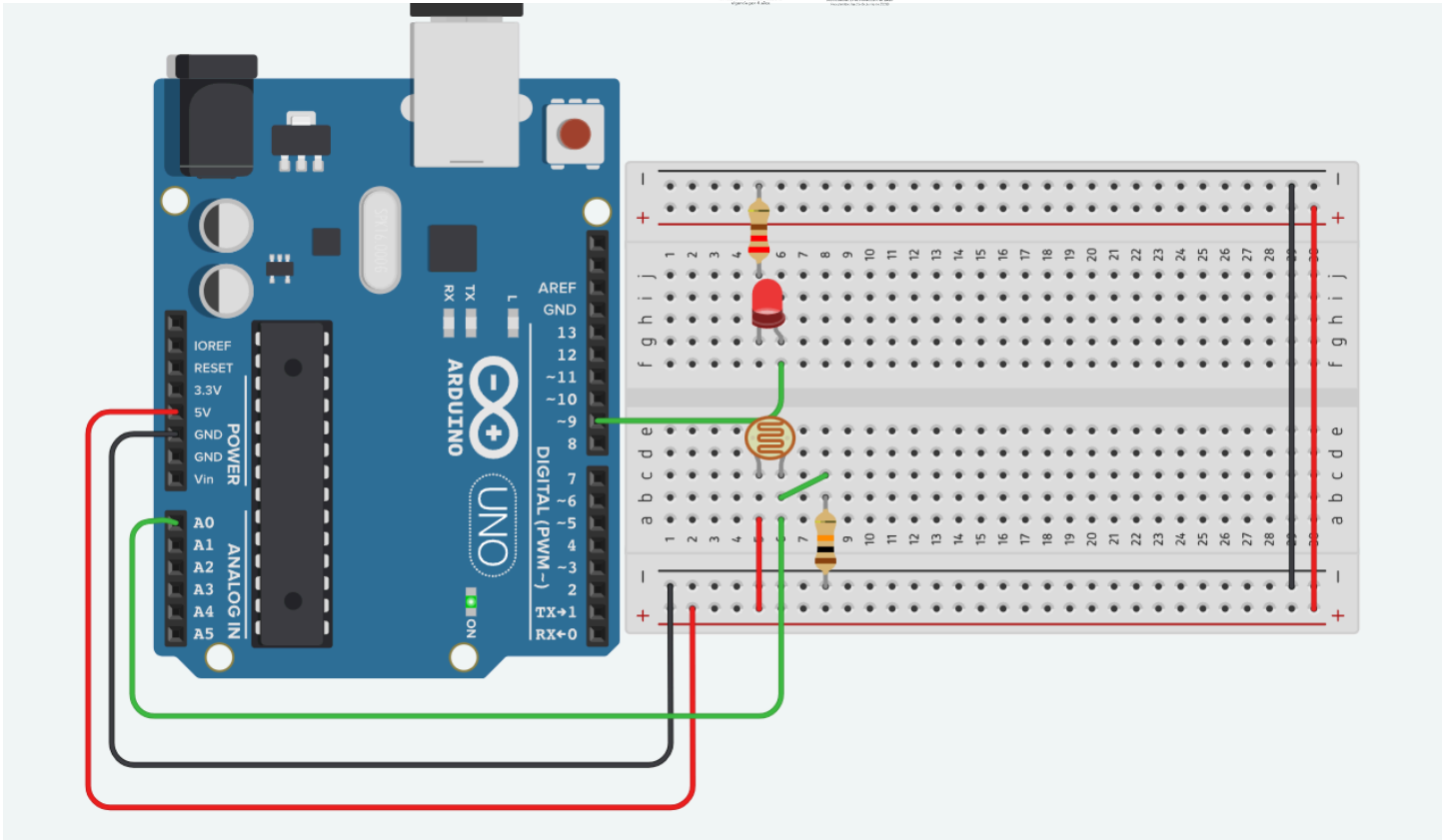
1. DESARROLLO DE LA PRÁCTICA

1.1 SIMULACIÓN EN THINKERCAD

1.1.1 COMPONENTES



1.1.2 MONTAJE



2.2 PROGRAMACIÓN EN ARDUINO IDE

```
// Definir los pines
const int ldrPin = A0; // Pin donde está conectado el LDR
const int ledPin = 9; // Pin PWM para el LED

void setup() {
  // Iniciar comunicación serie para monitorear los valores del sensor (opcional)
  Serial.begin(9600);
}

void loop() {
  // Leer el valor analógico del fotoresistor (0-1023)
  int ldrValue = analogRead(ldrPin);

  // Mapeo del valor del LDR (0-1023) al rango del PWM del LED (0-255)
  int ledBrightness = map(ldrValue, 0, 1023, 255, 0); // Menos luz -> más brillo

  // Ajustar el brillo del LED según el valor mapeado
  analogWrite(ledPin, ledBrightness);

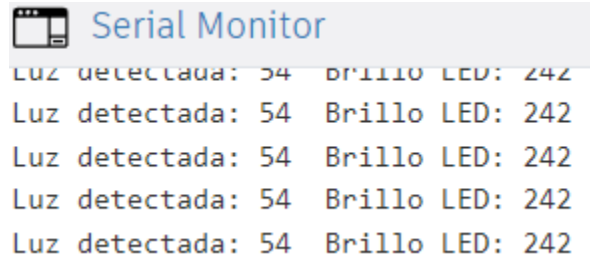
  // Monitorear el valor del LDR y el brillo en el monitor serie (opcional)
  Serial.print("Luz detectada: ");
  Serial.print(ldrValue);
  Serial.print(" Brillo LED: ");
  Serial.println(ledBrightness);

  // Pequeña pausa antes de la siguiente lectura
  delay(100);
}
```

Explicación

- Usamos `analogRead()` para leer el valor analógico del LDR, que nos proporciona un valor entre 0 y 1023 en función de la cantidad de luz detectada. Más luz produce valores más altos y menos luz produce valores más bajos.
- Utilizamos la función `map()` para ajustar el rango de valores del LDR (0-1023) al rango del PWM que controla el brillo del LED (0-255). De esta manera, cuanto más oscuro esté el ambiente, más brillará el LED, y viceversa.
- La función `analogWrite()` ajusta el brillo del LED en el pin PWM 9 basándose en el valor del sensor de luz. El PWM varía entre 0 (LED apagado) y 255 (máxima luminosidad).

- d. A través del monitor serie, podemos observar los valores que está leyendo el fotoresistor y cómo se están mapeando al brillo del LED para depurar o verificar el comportamiento del sistema.



```
Serial Monitor
Luz detectada: 54 Brillo LED: 242
Luz detectada: 54 Brillo LED: 242
Luz detectada: 54 Brillo LED: 242
Luz detectada: 54 Brillo LED: 242
Luz detectada: 54 Brillo LED: 242
```

2. ANÁLISIS DE RESULTADOS

- ¿Cómo influye la cantidad de luz en el valor que devuelve el fotoresistor (LDR)?

El fotoresistor es un sensor cuya resistencia disminuye cuando recibe más luz y aumenta cuando recibe menos luz. Por lo tanto, cuando hay más luz, el valor analógico leído por el pin A0 será más alto (cercano a 1023), y cuando hay menos luz, el valor será más bajo (cercano a 0). Esto afecta directamente el voltaje en el divisor de voltaje.

- ¿Cómo se ajusta el brillo del LED en función del valor del LDR?

El valor del LDR se lee en un rango de 0 a 1023 (lectura analógica), y luego se utiliza la función `map()` para convertirlo a un rango de 0 a 255, que es el rango utilizado por el PWM para ajustar el brillo del LED. De este modo, a mayor luz ambiente, el LED se atenuará, y a menor luz, el LED se iluminará más.

- ¿Por qué se utiliza la función `map()` en el código?

La función `map()` se usa para convertir el valor leído del LDR (0 a 1023) al rango que puede utilizar el PWM para controlar el brillo del LED (0 a 255). Sin esta conversión, no podríamos ajustar correctamente el brillo del LED, ya que el valor leído del LDR estaría fuera del rango aceptable para el PWM.

- ¿Qué sucede si cambiamos las condiciones de luz rápidamente?

El LED debería responder en tiempo real a los cambios de luz ambiente. Si la cantidad de luz aumenta o disminuye, el valor del LDR cambiará, y el brillo del LED se ajustará en consecuencia. Esto permite que el LED actúe como una lámpara adaptativa que ajusta su brillo según la cantidad de luz disponible.

- ¿Qué se podría mejorar en el circuito o código para optimizar el comportamiento del sistema?

Podríamos agregar un filtro de software para suavizar las transiciones de brillo, de modo que el LED no parpadee o cambie abruptamente cuando los niveles de luz varían ligeramente. También podríamos ajustar los tiempos de respuesta reduciendo el retardo (delay(100) en el código) para una reacción más rápida o agregar un capacitor en el circuito para estabilizar las fluctuaciones de lectura.

3. CONCLUSIONES

- a. Los estudiantes han aprendido a conectar un sensor de luz (fotoresistor) con un actuador (LED) utilizando un Arduino. Esto refuerza la comprensión de cómo los sensores pueden influir en la respuesta de dispositivos conectados, mostrando un claro ejemplo de interacción entre hardware y software.
- b. La modulación por ancho de pulso (PWM) permite un control eficiente y gradual del brillo de un LED. Este concepto es fundamental en el control de dispositivos en proyectos de electrónica, ya que permite ajustar la intensidad de salida sin consumir mucha energía adicional.
- c. El uso de un divisor de voltaje con el LDR permitió convertir los cambios en la resistencia en valores analógicos que el Arduino puede leer. Este principio es clave en la lectura de sensores resistivos y se aplica a muchos otros componentes como termistores y sensores de presión.
- d. La función map() es esencial en muchos proyectos con Arduino para convertir rangos de valores y adaptar lecturas de sensores a los rangos de control de actuadores. Su uso en este proyecto permitió ajustar el brillo del LED de manera proporcional a la cantidad de luz detectada.
- e. Este proyecto se puede ampliar fácilmente para aplicaciones más complejas, como sistemas de iluminación automática en habitaciones o vehículos, donde los cambios en las condiciones de luz controlen el brillo de las luces de manera automática y eficiente.