Instituto Politécnico Nacional

Escuela Superior de Cómputo

Practica 3.- Iluminación nocturna.

Sistemas en chip
Profesora: Ana Luz Barrales Lopez

Luciano Hernández Jonathan
Rodriguez Morales Iñaki

16 de octubre de 2024

1 Introducción

En los **sistemas en chip (SoCs)**, la modulación por ancho de pulso (**PWM**, por sus siglas en inglés) es una técnica fundamental para controlar la potencia entregada a dispositivos como LEDs, motores o cualquier otro actuador que requiera variaciones en su funcionamiento. A través de PWM, el microcontrolador puede simular diferentes niveles de voltaje mediante la rápida conmutación entre encendido y apagado, ajustando el ciclo de trabajo (el tiempo que permanece en alto). En aplicaciones donde se desea controlar la intensidad de luces o la velocidad de motores, el uso de PWM es esencial, ya que permite un control preciso de forma eficiente y con bajo consumo energético.

Por otro lado, el **Watchdog Timer (WDT)** es un temporizador crucial en los sistemas embebidos que asegura la robustez del sistema. El WDT se configura para monitorear el funcionamiento del sistema y reiniciarlo si detecta que el programa ha dejado de funcionar correctamente, como en casos de cuelgues o errores inesperados. Si el microcontrolador no resetea el WDT dentro de un período específico, el watchdog genera un reinicio forzado, evitando que el sistema permanezca inactivo o en un estado indeterminado. Esta funcionalidad es clave en aplicaciones donde la continuidad operativa es crítica, garantizando que el sistema vuelva a un estado conocido tras un fallo.

1.1 Objetivos

- **Modulación PWM.-** Controlar el brillo de los LEDs de forma precisa utilizando PWM, permitiendo variaciones graduales en la intensidad lumínica basadas en el valor del potenciómetro.
- **Watchdog Refresh.-** Utilizar el Watchdog Timer para reiniciar el sistema automáticamente tras un ciclo de operación, garantizando la estabilidad y supervisión del sistema.

1.2 Materiales

Los materiales utilizados en la práctica son los siguientes:

- Arduino Uno
- Potenciómetro de 10k ohms
- 5 resistencias de 330 ohms
- 5 luces LED
- Protoboard
- Cables de conexión
- Arduino IDE
- Fotoresistencia

2 Desarrollo de la Práctica

2.1 Explicación de las conexiones

Para la conexión de este sistema, se deben conectar cinco LEDs a los pines PWM del Arduino, específicamente en los pines 3, 5, 6, 9 y 10. Los cátodos de los LEDs deben ir conectados a tierra, mientras que los ánodos se conectan a los pines correspondientes del Arduino para controlar su brillo utilizando modulación por ancho de pulso (PWM).

La fotoresistencia se conecta al pin A0 del Arduino. Además, se debe crear un divisor de voltaje utilizando una resistencia de 10kohms conectada entre la fotoresistencia y tierra, lo que permite al Arduino medir la cantidad de luz ambiental a través del pin analógico.

Por último, el potenciómetro se conecta al pin A1 del Arduino. Uno de los terminales del potenciómetro va a 5V, el otro terminal va a tierra, y el pin central (cursor) se conecta al pin A1 del Arduino para leer su valor analógico y ajustar el brillo de los LEDs según la posición del potenciómetro.Para la conexión de este sistema, se deben conectar cinco LEDs a los pines PWM del Arduino, específicamente en los pines 3, 5, 6, 9 y 10. Los cátodos de los LEDs deben ir conectados a tierra, mientras que los ánodos se conectan a los pines correspondientes del Arduino para controlar su brillo utilizando modulación por ancho de pulso (PWM).

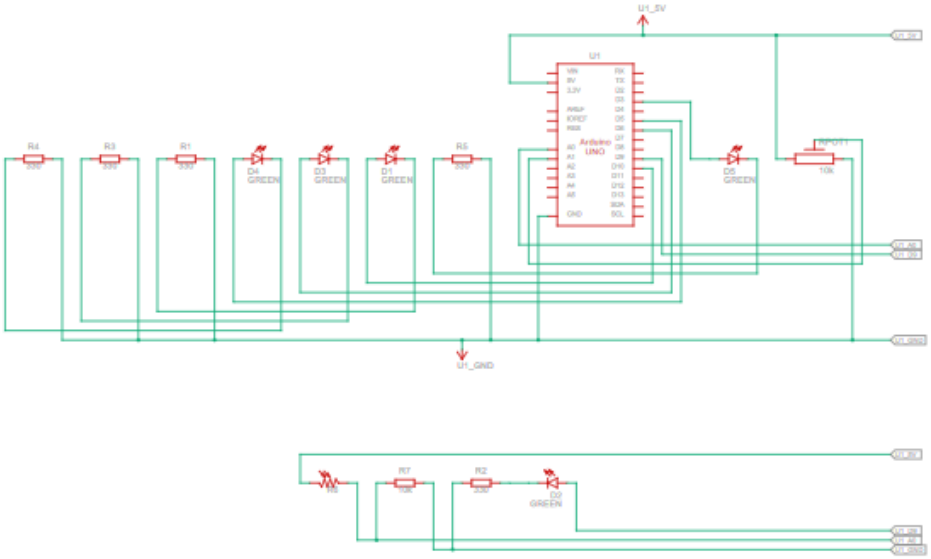

Figure 1: Diagrama del circuito: entrada analógica a salida digital.

2.2 Desarrollo del código

2.2.1 Declaración de pines y variables

En esta primera parte, se definen los pines a los que están conectados los LEDs, la fotoresistencia y el potenciómetro. Se utiliza un arreglo para almacenar los pines PWM (3, 5, 6, 9, 10), mientras que los pines A0 y A1 se utilizan para la fotoresistencia y el potenciómetro, respectivamente. Además, se declaran variables para el nivel de luz detectado, el valor del potenciómetro, el valor del PWM que se aplicará a los LEDs, y

la duración del ciclo de apagado, que es de 8 segundos (8000 milisegundos). También se declara una variable startTime que servirá para medir el tiempo transcurrido.

```
const int ledPins[] = {3, 5, 6, 9, 10}; // Pines de los LEDs
const int photoresistorPin = A0; // Pin de la fotoresistencia
const int potentiometerPin = A1; // Pin del potenci metro

int lightLevel = 0; // Variable para leer la luz de la fotoresistencia
int potValue = 0; // Valor del potenci metro
int pwmValue = 255; // Valor inicial PWM para los LEDs
unsigned long timeInterval = 8000; // Duraci n total de 8 segundos
unsigned long startTime;
```

2.2.2 Configuración inicial en setup()

En la función setup(), se configuran los pines de los LEDs como salidas y se apagan todos los LEDs al inicio para asegurar que estén apagados cuando se inicie el sistema. Se configuran los pines de la fotoresistencia y el potenciómetro como entradas, ya que se leerán valores de estos sensores durante el funcionamiento. Además, el watchdog se desactiva al comienzo para evitar que el sistema se reinicie prematuramente.

```
void setup() {
  for (int i = 0; i < 5; i++) {
    pinMode(ledPins[i], OUTPUT);
    analogWrite(ledPins[i], 0); // Apaga los LEDs al inicio
  }
  pinMode(photoresistorPin, INPUT);
  pinMode(potentiometerPin, INPUT);

  wdt_disable(); // Deshabilitar el watchdog al inicio
}
```

2.2.3 Lectura de la fotoresistencia y ciclo de apagado gradual en loop()

Dentro de la función loop(), el sistema primero lee el valor de la fotoresistencia para determinar si hay poca luz (valor inferior a 512). Si se detecta poca luz, comienza el ciclo de apagado gradual de los LEDs. Se guarda el tiempo de inicio del ciclo y se entra en un ciclo de 8 segundos donde los LEDs van disminuyendo su brillo de manera gradual, basado en el tiempo transcurrido. La variable fadeFactor se usa para reducir progresivamente el brillo de los LEDs.

```
void loop() {
  lightLevel = analogRead(photoresistorPin); // Leer el valor de la
   fotoresistencia

  if (lightLevel < 512) { // Solo inicia si no hay luz en la
   fotoresistencia
    startTime = millis(); // Guardar el tiempo de inicio del ciclo de
   apagado

    while (millis() - startTime <= timeInterval) { // Proceso de
   apagado gradual de LEDs durante 8 segundos
      // Leer el valor del potenci metro
      potValue = analogRead(potentiometerPin);
      int mappedPotValue = map(potValue, 0, 1023, 0, 255); // Ajustar
   el valor del potenci metro al rango de PWM
```

2.2.4 Ajuste del brillo de los LEDs con el potenciómetro

Durante el proceso de apagado gradual, el sistema sigue monitoreando el valor del potenciómetro para permitir ajustes en el brillo de los LEDs en cualquier momento. El valor leído del potenciómetro se convierte a un valor adecuado para el PWM (de 0 a 255) utilizando la función map(). El brillo de los LEDs se ajusta en función de este valor y el tiempo transcurrido en el ciclo de apagado.

```
      // Calcular el tiempo transcurrido y ajustar el brillo
      unsigned long elapsedTime = millis() - startTime;
      float fadeFactor = 1.0 - (float)elapsedTime / timeInterval;
      pwmValue = mappedPotValue * fadeFactor; // Ajustar el brillo
    seg n el potenci metro y el tiempo

      // Aplicar el brillo a los LEDs
      for (int i = 0; i < 5; i++) {
        analogWrite(ledPins[i], pwmValue);
      }
```

2.2.5 Activación del Watchdog y reinicio del sistema

Cuando el ciclo de apagado se acerca a su fin (en los últimos 500ms), se activa el Watchdog Timer con un retraso de 1 segundo, lo que forzará el reinicio del sistema. En este punto, los LEDs están casi apagados y, justo antes de apagarse por completo, el watchdog reinicia el sistema, haciendo que los LEDs vuelvan a su brillo máximo y el ciclo de apagado se repita.

```
      // Verificar si es tiempo de activar el watchdog
      if (elapsedTime >= timeInterval - 500) { // Justo antes de
    apagarse ( ltimos  500ms)
        wdt_enable(WDTO_1S); // Activar el watchdog con un retraso de 1
     segundo para el reinicio
      }
    }

    // Reinicio del ciclo, el watchdog se activar  y reiniciar  el
    sistema
  }
}
```

2.3 Simulación

La simulación del circuito fue realizada en la plataforma **Tinkercad**, un entorno de simulación en línea que permite construir y ejecutar circuitos electrónicos de manera virtual. En el que previo a implementarse con componentes reales, se llevo a cabo en esta plataforma.

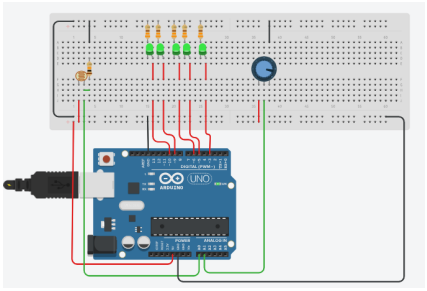

Figure 2: Funcionamiento del cricuito

Primera imagen donde se puede observar el funcionamiento del circuito de manera correcta en la simulación iluminandose al tener el fotoresistor sin luz.

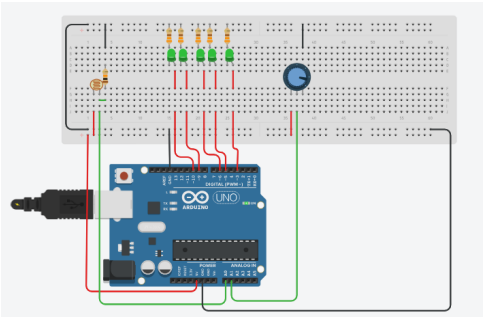

Figure 3: Funcionamiento del cricuito

Segunda imagen donde se puede observar la luz de los leds regulada mediante el potencioemtro.

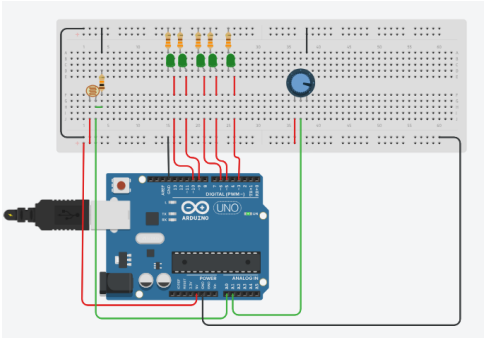

Figure 4: Funcionamiento del cricuito

Tercera imagen donde se puede observar el sistema a punto de apagarse por completo tras irse apagando gradualmente.

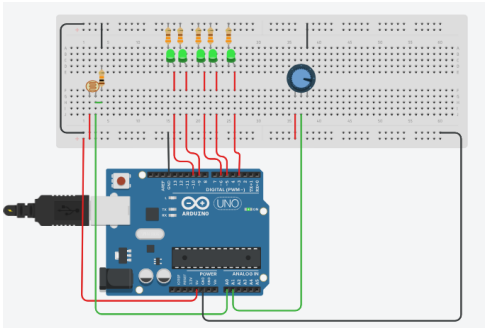

Figure 5: Funcionamiento del cricuito

Cuarta imagen donde se puede observar el sistema reiniciandose tras activarse el **watchdog** tras irse apagando gradualmente.

2.4 Implementación

Una vez implementado la conexion y desarrollo del codigo dentro de la simulación, se elaboró el circuito de manera practica con componentes reales para observar su funcionamiento, a continuación, se muestran las imagenes del citcuito en funcionamiento conectado a a un arduino uno.

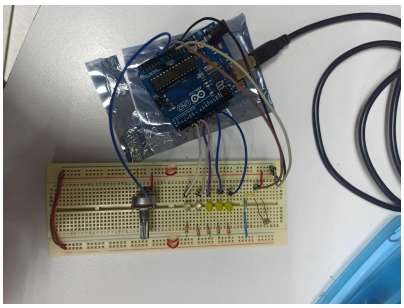

Figure 6: Funcionamiento del cricuito

Primer fotografía donde se puede observar el sistema apagado tras recibir luz directa.

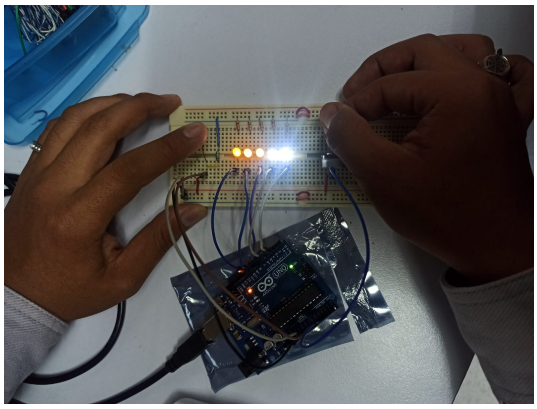

Figure 7: Funcionamiento del cricuito

Segunda fotografía donde se puede observar como se enciende en niveles maximos tras no recibir luz la fotoresistencia.

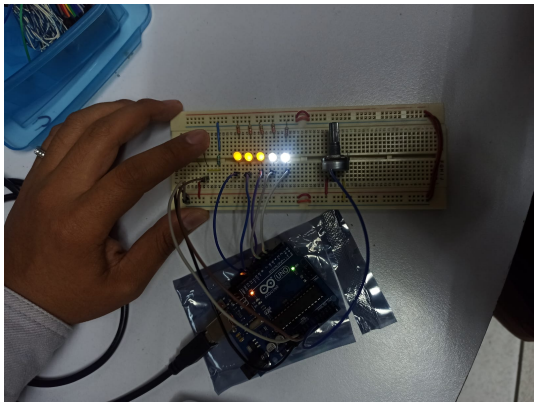

Figure 8: Funcionamiento del cricuito

En esta fotografía se puede observar como se regula la luz de los leds mediante la manpulación del potenciometro.

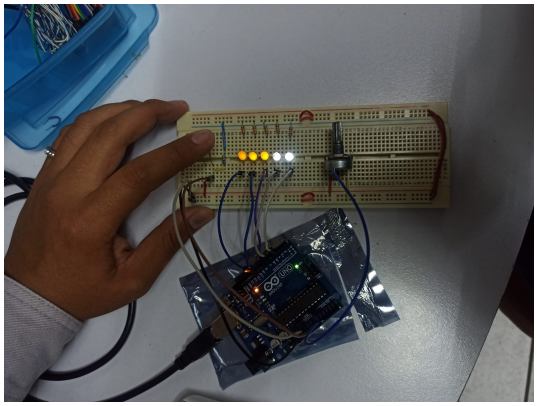

Figure 9: Funcionamiento del cricuito

En esta fotografía se puede observar el sistema apunto de apagarse donde entra en funcionamiento el **watchdog**

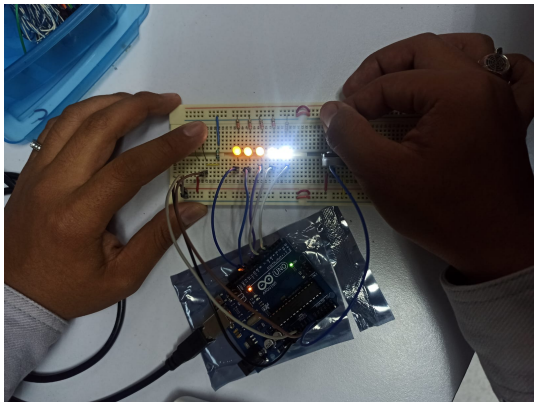

Figure 10: Funcionamiento del cricuito

Finalmente podemos ver como se reainicia el sistema justo antes de apagarse, poniendo los leds al maximo de nuevo, comenzando con el ciclo de nuevo.

3 Conclusiones

- **Morales Rodriguez Iñaki** Con esta práctica, se logró comprender el funcionamiento de la modulación por ancho de pulso (PWM) y su aplicación en el control preciso de dispositivos como LEDs. Al variar el ciclo de trabajo, se puede regular la intensidad de las luces de manera eficiente, lo que demuestra el potencial de PWM en la gestión de energía dentro de sistemas embebidos. Además, se entendió cómo el Watchdog Timer (WDT) es un recurso vital en los SoCs, ya que garantiza la estabilidad del sistema reiniciándolo de forma automática ante cualquier mal funcionamiento, asegurando su operación continua y robusta en entornos críticos.

- **Luciano Hernández Jonathan** Esta práctica permitió asimilar de manera efectiva el uso de la modulación por ancho de pulso (PWM) para controlar la potencia de salida de los pines PWM, aplicándolo al control de brillo en LEDs. Asimismo, se logró integrar y comprender el rol esencial del Watchdog Timer (WDT) dentro de los SoCs, evidenciando su importancia al monitorear el correcto funcionamiento del sistema y realizar reinicios automáticos cuando es necesario. Con esta experiencia, se pudo apreciar cómo ambas funciones contribuyen a la eficiencia y seguridad en el diseño de sistemas embebidos.

4 Referencias

1 A. Hernández, "Entradas Analógicas y Convertidor Analógico-Digital (ADC) en Arduino," Control Automático Educación, 17-Oct-2023. [Enlace: https://control automaticoeducacion.com/sistemas-embebidos/arduino/entradas-analogicas-adc/].

- Código Electrónica. "Señal Digital a Display de 7 Segmentos."https://tinyurl.com/4n92v39d, Accedido el 17 de octubre de 2024.

- G. M.1-5611CURG. "Microcontroladores." AG Electrónica, https://agelectronica.lat/pdfs/textos/G/GM1-5611CURG.PDF. Accedido el 17 de octubre de 2024.