Circuitos Digitales y Microcontroladores

GRUPO 6

Chanquía, Joaquín (02887/7)

Forden Jones, Ian E. W. (02543/3)

Ollier, Gabriel (02958/4)

ÍNDICE

1. INTRODUCCIÓN

1.1 - Enunciado

Realizar un programa para controlar la intensidad y el color del LED RGB con la técnica de PWM. En el kit de clases el mismo se encuentra conectado a los terminales PB5, PB2 y PB1 (RGB) a través de resistencias de limitación de 220ohms y en forma ánodo común. Para la simulación del modelo en Proteus puede utilizar RGBLED-CA. Requerimientos detallados:

1. Genere en los tres terminales de conexión del LED, tres señales PWM de frecuencia mayor o igual a 50Hz y con una resolución de 8 bits cada una.

2. Seleccione la proporción de color de cada LED (de 0 a 255) para obtener un color resultante.

3. Mediante un comando por la interfaz serie UART0 deberá activar cual proporción de color desea modificar. Por ejemplo, envie ´R´ para modificar el rojo, ´G´ para el verde y ´B´ para el azul.

4. Utilice el potenciómetro (resistencia variable) del kit conectado al terminal ADC3 para modificar el brillo del color seleccionado vía comando serie.

1.2 - Interpretación

El objetivo del trabajo es crear un sistema que controle la intensidad y el color de un LED RGB mediante la técnica de PWM. Este sistema utilizará un microcontrolador que generará señales PWM en los terminales PB5, PB2 y PB1 para controlar los LEDs rojo, verde y azul respectivamente, con una frecuencia de al menos 50Hz y una resolución de 8 bits. El LED RGB está conectado a estos terminales a través de resistencias de 220 ohm y en configuración de ánodo común.

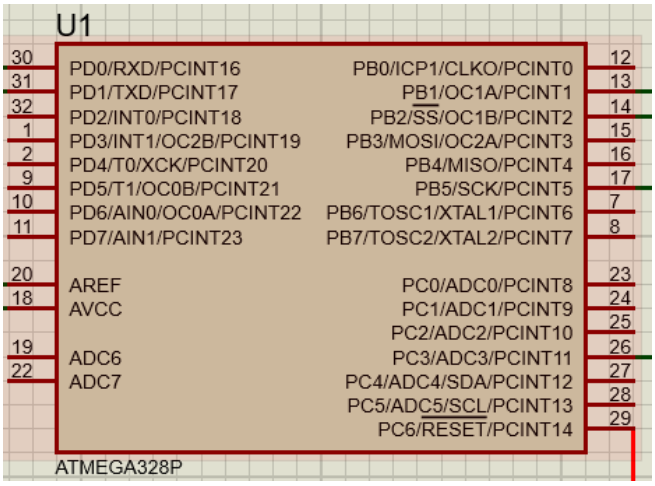
El microcontrolador debe programarse para permitir la selección y ajuste de la proporción de color de cada LED, utilizando valores en un rango de 0 a 255. Para lograr esto, el sistema debe implementar una interfaz de comunicación mediante UART0 que permita recibir comandos para seleccionar qué color modificar. Los comandos serán 'r' o 'R' para rojo, 'g' o 'G' para verde y 'b' o 'B' para azul. Esto implica que el microcontrolador debe ser capaz de manejar la comunicación UART0 y responder adecuadamente a los comandos recibidos para ajustar las señales PWM de los colores correspondientes.

Además, el sistema debe permitir la modificación del brillo del color seleccionado utilizando un potenciómetro conectado al terminal ADC3 del

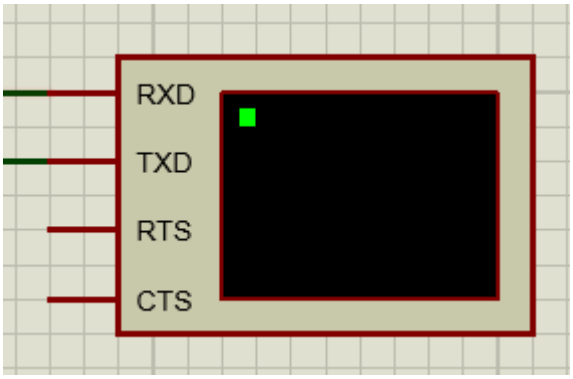
microcontrolador. El valor del potenciómetro debe ser leído y convertido a un valor digital mediante el ADC del microcontrolador. Este valor digital se utilizará para ajustar el brillo del color seleccionado vía comando serie, permitiendo un control dinámico y preciso de la intensidad del LED.

Componentes utilizados:

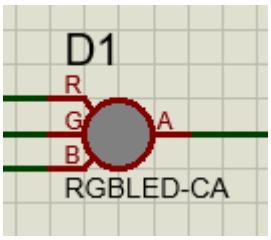
MCU Atmega328p



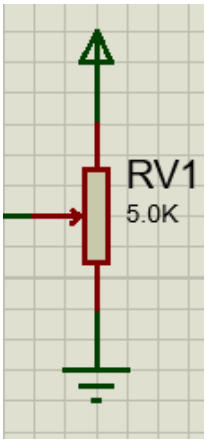
Terminal virtual



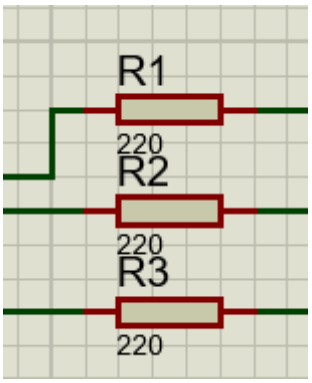
Led RGB



Resistencia variable (potenciómetro)



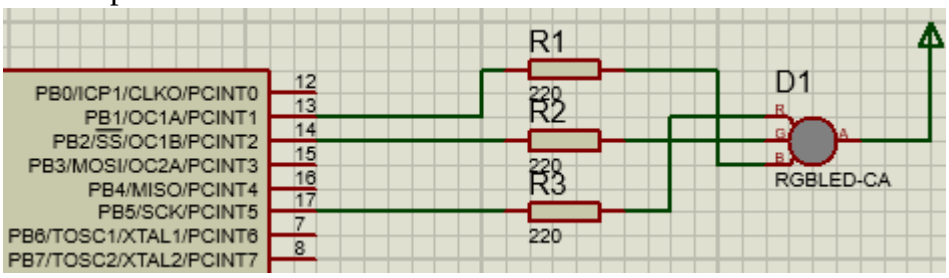
Resistencias



2. RESOLUCIÓN

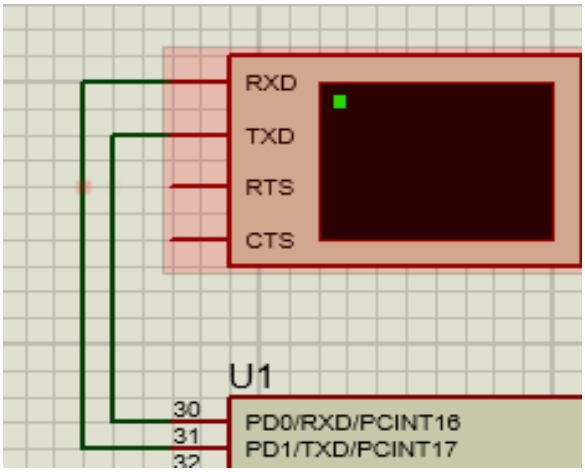
2.1 Conexionado de los periféricos

El led RGB que se utilizará está en forma de ánodo común por lo cual se conectará el ánodo a VCC y las terminales R, G y B se conectarán a los puertos PB5, PB2 y PB1 respectivamente:

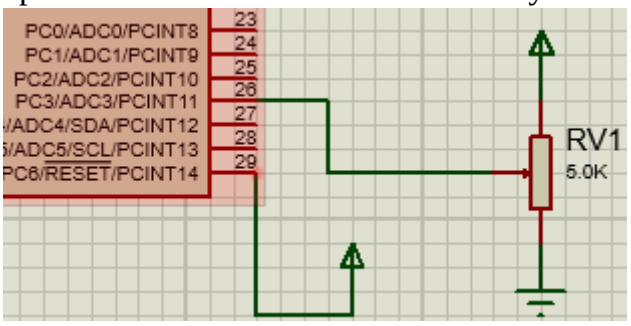


Debido a la conexión en forma de ánodo común se debe tener en cuenta al momento de implementar el código que los terminales R, G y B activan sus respectivos LEDs cuando el puerto al que cada uno esté conectado tenga un valor lógico bajo (0).

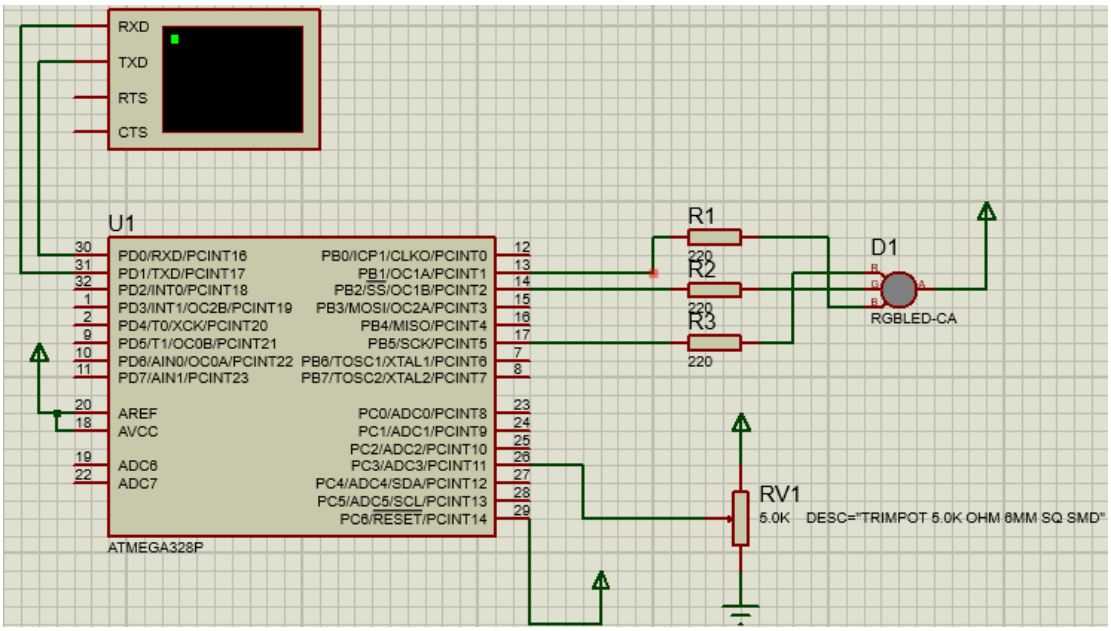
La terminal virtual que se utilizará para simular el envío de comandos por la interfaz UART0 se conectará a los puertos PD0 y PD1 mediante sus terminales RDX y TDX como se muestra a continuación:



Por último, el terminal variable del potenciómetro se conectará al puerto PC3 que se corresponde a su vez con el terminal ADC3 que se pide en la consigna mientras que las otras 2 terminales del potenciómetro se conectarán a VCC y a GND respectivamente:



A continuación, se muestra el conexionado final de todos los elementos mencionados anteriormente:



2.2 Funcionamiento del software

La resolución de este problema se llevó a cabo con el uso de 3 archivos de tipo .c y sus respectivos archivos cabecera.

- main.c: este archivo contiene la función principal que será ejecutada al momento de correr el programa.
- pot.c: Este archivo contiene las funciones necesarias para inicializar y encuestar el valor actual del potenciómetro incluido en el kit.
- PWM.c: Este archivo se encarga de manejar la intensidad de los distintos colores en el LED RGB

Además de estos archivos se utilizó el archivo serialPort.c dado por la catedra para el uso del módulo UART.

main.c

Este archivo contiene la función principal del programa, encargada de llamar a las funciones de inicialización de los distintos periféricos utilizados en este. En este archivo tambien se encuentra la rutina de interrupción encargada de la lectura del teclado.

```
int main(void)
{
        Inicializacion de las funciones PWM para la intensidad del led
        Inicializacion del modulo ADC para la medida del potenciómetro

        Se inicializa el formato 8N1 y BAUDRATE = 9600bps
        Se activa el Transmisor del Puerto Serie
        Se activa el Receptor del Puerto Serie
        Se activa la interrupción de recepcion.
        Se activan las interrupciones globales

        while (1) // loop infinito
        {
                Si se esta modificando un color{
                        Se recibe el valor del potenciometro
                        Se configura la intensidad del color correspondiente
                }
        }

}
```

Lectura de teclado:

La rutina de interrupción de lectura del teclado funciona recibiendo un carácter y cambiando el color que será modificado con la posición del potenciómetro siguiendo las iniciales de RGB (R de Red/Rojo, G de Green/Verde y B de Blue/Azul). Si se ingresa una letra F se dejará estático el color del LED y no se seguirá modificando ninguno de los colores hasta que se vuelva a ingresar su inicial. Si se esta modificando un color y se ingresa la inicial de otro, el primer color queda fijo en la intensidad actual y se pasa a modificar el nuevo color ingresado.

```
ISR(USART_RX_vect){
	char RX_Buffer = UDR0; //la lectura del UDR borra flag RXC
	// Si se presiona 'f' se alterna la transmision de datos
	if((RX_Buffer == 'f')||(RX_Buffer == 'F')){ // Si se presionó una F
		active = 0; // Se finaliza la modificacion
		SerialPort_Send_String(msgF);   // Envío el mensaje de Bienvenida
	}
	// Si se presiona 'R' se alterna la transmision de datos
	if((RX_Buffer == 'r')||(RX_Buffer == 'R')){ // Si se presionó una R y no se está modificando otro color
		active = 1; // Se activa la modificacion de color
		CAct = 'R';
		SerialPort_Send_String(msgR);   // Envío el mensaje de Bienvenida
	}
	// Si se presiona 'G' se alterna la transmision de datos
	if((RX_Buffer == 'g')||(RX_Buffer == 'G')){ // Si se presionó una G
		active = 1; // Se activa la modificacion de color
		CAct = 'G';
		SerialPort_Send_String(msgG);   // Envío el mensaje de Bienvenida
	}
	// Si se presiona 'B' se alterna la transmision de datos
	if((RX_Buffer == 'b')||(RX_Buffer == 'B')){ // Si se presionó una B
		active = 1; // Se activa la modificacion de color
		CAct = 'B';
		SerialPort_Send_String(msgB);   // Envío el mensaje de Bienvenida
	}
	RX_Buffer=0;
}
```

pot.c

Este archivo contiene las dos funciones necesarias para la lectura del potenciómetro incorporado en el kit.

- void ADC_Init(): Esta funcion es la encargada de inicializar la configuración del ADC. La configuración esta realizada como para ser usada con el método de polling:
  Bits en ADCSRA:
  - ADEN en 1: Se habilita el ADC.
  - ADPS[2..0] en 111: Se establece el prescaler en un valor de 128.

  Bits en ADMUX:
  - ADLAR en 1: Se programa la justificación como Left-Justified para que pueda usarse el resultado con los 8 bits mas significativos.
  - REFS[1..0] en 01: Se establece VCC como el valor de referencia.
  - MUX[3..0] en 0011: Se conecta el canal 3 al ADC, el cual es el conectado al potenciómetro del kit.
- uint8_t getCant(): Esta es la funcion encargada de devolver el valor actual del potenciómetro cuando la misma es llamada. Para ello primero se habilita el bit ADSC en ADCSRA para inicializar una conversión. Luego de esto se espera en un while por la finalización de la conversión cuando el valor del bit ADIF en ADCSRA pase a ser 1. Luego de esto se devuelve el valor de la conversión hallado en ADCH.

PWM.c

Este archivo contiene las funciones necesarias para el manejo de la intensidad de los colores del LED RGB mediante el protocolo PWM. Dos de estos colores están conectados a los pines que son salida del protocolo PWM del timer 1. Estando el color azul conectado a OC1A y el color verde a OC1B. El pin conectado al color rojo en cambio no está conectado a la salida PWM de ningún timer por lo que la intensidad del mismo fue controlada con un PWM por software.

- PWM_Init():

  Configuracion de los leds azul y verde:

Para configurar el PWM del timer 1 se modificaron los registros:

- WGM1[3..0] en 0101: Se establece el modo fast PWM de 8 bits.
- COM1A/B[1..0] en 11: Establece que OC1A/B se encienda al momento de llegar al valor de OCR1A/B respectivamente.
- CS1[2..0] en 001: para establecer que se utiliza el clk sin preescaling.

Luego de esto se estableció un valor inicial para los registros de comparación OCR1A y OCR1B en 0 para iniciar el programa con estos colores apagados.

Configuracion del led rojo:

Para realizar el PWM por software se configuro una interrupción por timer utilizando el timer 0, esta fue programada con el modo CTC y se realiza cada 40 us. Esta interrupción utiliza una variable de 8 bits que cuenta cada vez que se llama a la rutina y establece el valor del pin conectado al led rojo en 0 si el contador tiene un valor menor al configurado en la variable de comparación o en 1 si es mayor o igual.

```
ISR(TIMER0_COMPA_vect) { // Rutina de servicio a interrupcion
    cuenta++;
    if (cuenta < DELTA){
        PORTB &= ~(1<<PORTB5); // Se pone el pin en 0 para prender el led
    }else{
        PORTB |= (1<<PORTB5); // Se pone el pin en 1 para apagar el led
    }
}
```

Configuracion de intensidad de los colores:

Para la configuración de la intensidad de cada color se incluyeron tres funciones, una para cada color, que establecen la intensidad del color cambiando OCR1A y OCR1B en el caso de los colores azul y verde respectivamente y el valor de la variable DELTA utilizada en la rutina de interrupción para el led rojo.

```
void SetCT_Red(uint8_t comp){
    DELTA = comp;
}
void SetCT_Green(uint8_t comp){
    OCR1B = comp;
}
void SetCT_Blue(uint8_t comp){
    OCR1A = comp;
}
```

3. VALIDACIONES

Al correr el programa, se observa desde la terminal virtual un mensaje indicando "Control de color de un RGB", lo cual confirma el inicio del sistema y su preparación para recibir comandos. A continuación, se detallan las observaciones realizadas durante la simulación:

- Control de Color:

Al presionar la tecla 'r' o 'R' para rojo, 'g' o 'G' para verde y 'b' o 'B' para azul, el sistema responde adecuadamente, cambiando la intensidad del LED correspondiente y mostrando un mensaje en la terminal que indica que se está modificando el color rojo, verde o azul respectivamente.

La intensidad del color seleccionado puede ajustarse utilizando el potenciómetro conectado al terminal ADC3. Se observa que, al cambiar la posición del potenciómetro, el brillo del LED correspondiente varía de manera coherente con el valor leído.

- Funcionalidad Adicional de Control:

Al presionar la tecla 'f' o 'F' en la terminal, la modificación de intensidad se detiene de inmediato. El sistema no permite que el valor fluctúe, aunque se esté variando el potenciómetro, mostrando que la funcionalidad de pausa funciona correctamente.

Estas características del funcionamiento se ven demostradas en las siguientes pruebas, una muestra el programa funcionando en el kit entregado por la catedra y el otro muestra una simulación realizada en el programa Proteus.

Prueba de funcionamiento en el kit:
https://drive.google.com/file/d/1aLw9RoRvS1lLXUVbqmuz85NVEXVYQYTi/view?usp=sharing

Prueba de funcionamiento en Proteus:

https://drive.google.com/file/d/1wP952X4qMr7LPN1ENik0v5lhHnU94q3d/view?usp=sharing

4. CONCLUSIÓN

El sistema desarrollado cumple con todos los requerimientos establecidos en la consigna del trabajo práctico. La implementación de señales PWM en los pines PB5, PB2 y PB1 con la frecuencia y resolución adecuadas permite controlar la intensidad de cada color del LED RGB de manera efectiva. La comunicación UART0 funciona correctamente, permitiendo seleccionar y ajustar los colores mediante comandos desde la terminal. Además, la lectura del potenciómetro y el ajuste del brillo en tiempo real añaden un control dinámico y preciso.

La funcionalidad adicional de pausar y reanudar la lectura del potenciómetro mediante un comando serie demuestra la flexibilidad y robustez del sistema. Las pruebas realizadas en Proteus 8 validan que el sistema responde adecuadamente a todos los comandos y ajustes, garantizando su correcto funcionamiento.

El siguiente repositorio en GitHub contiene el código implementado y el entorno de pruebas en proteus

https://github.com/joacochanquia/CDyM-TP4-Grupo6