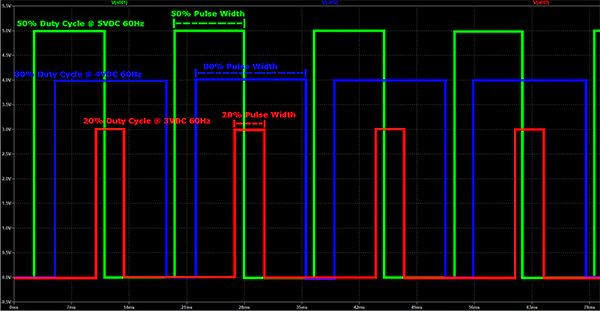
**Interpretación**

tp N°4 ENTREGABLE

1 de agosto de 20220



BLANCO Valentin Nicolas, PALADINO gABRIEL AGUSTIN

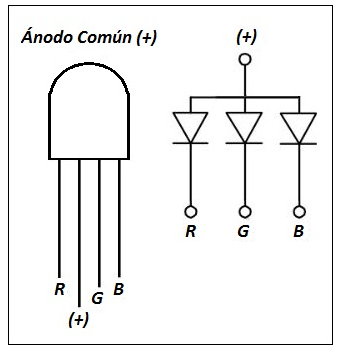
CIRCUITOS DIGITALES Y MICROCONTROLADORES

Se debería realizar un programa en c donde conectando un led rgb al Arduino y dando a este led distintas señales en cada una de sus entradas se conseguiría dar distintos colores al led. Dicho led cuenta con 3 pines para su control (uno para cada color) y la conexión de los mismo debe ser PB5 para el rojo, PB2 para verde y PB1 para azul.

Además de este periférico se debe utilizar un potenciómetro para controlar la intensidad de cada color del led y una terminal para definir a cuál color se le va a realizar dicha modificación. Para hacer funcionar la terminal se utiliza el periférico UART que trae incorporado el MCU y para leer los valores del potenciómetro se hace uso de un conversor analógico digital que también tiene incorporado el MCU.

**Resolución del Problema:**

**Corriente máxima para cada LED**

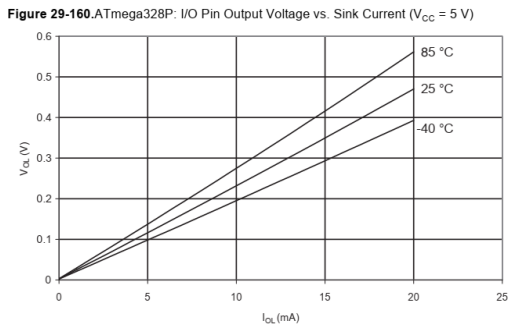


Ya que el led es de tipo ánodo común (como se observa en la imagen superior) la intensidad de los colores son activos en bajo, por lo que la manera en la que se calcula la corriente máxima es la siguiente:



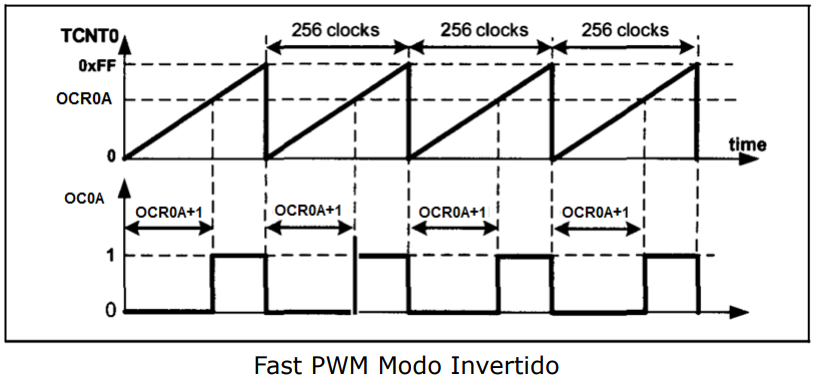
A partir de esta formula y siguiendo la figura 29-160 de la hoja de datos donde se observa que a temperaturas normales de operación el Vol maximo sera de aproximadamente 0.47 V y siendo Rled 220 Ω en los 3 casos se consguen las siguientes corrientes maximas para cada color:

|  |  |  |
| --- | --- | --- |
| Color | Voltaje color | Corriente maxima |
| Rojo | 2.049V | 11.27mA |
| Verde | 2.81V | 7.81mA |
| Azul | 2.871V | 7.54mA |

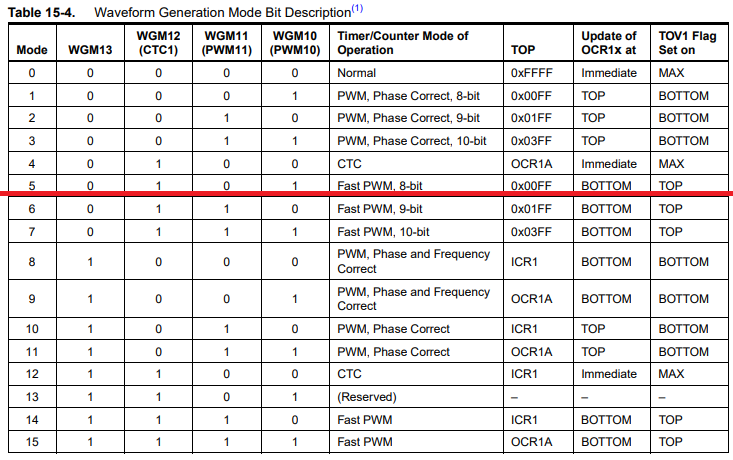


**Generación de señales**

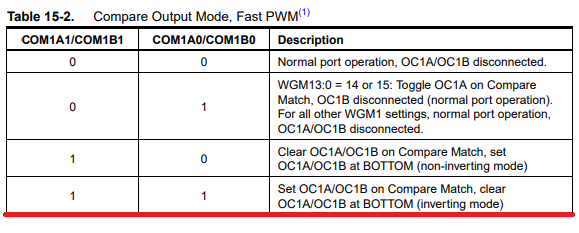
Para generar distintas frecuencias de tal manera que varíen los colores del led se utilizó la técnica PWM (pulse width modulation) lo que permite que se pueda obtener una señal analógica a partir de una señal digital. Como los colores son activos en bajo para solucionar el problema escogimos PWM invertido de modo tal que al poner el registro comparador del timer en 0 el led este apagado y en 255 este a la máxima intensidad. A continuación, se muestra un gráfico de cómo se generaría una señal PWM de 8 bits invertida con timer 0.



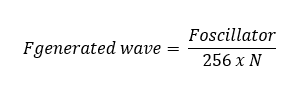
Generación de PWM para azul y verde (Timer1): Como los colores azul y verde corresponden a los puertos PB1 y PB2 respectivamente se puede utilizar la función PWM del timer1 ya que la señal generada por el mismo se refleja en estos puertos. El enunciado solicita una resolucion de 8 bits por lo que se puede configurar el timer en modo 5 (Fast PWM de 8 bits). Para configurarlo de dicho modo seguimos la tabla de configuración del timer1:



Luego de seleccionar dicho modo ademas se configuro el mismo para que la señal sea dada de manera invertida (COM1A1 y COM1A0 en 1 para el color azul, COM1A0 y COM1A0 en 1 para el verde) ya que como se explico anteriormente la conexión del led era anodo comun de modo tal que 0 indica apagado y 255 la intensidad maxima del color.

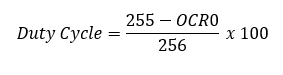


El enunciado solicita una frecuencia de 50 hz o mayor por lo que a partir de la siguiente formula se puede calcular la frecuencia de la señal generada por el timer1



De esta cuenta reemplazando Foscillator=16Mhz y N=1024 (siendo este el preescaler) se consiguen 61.03 Hz.

Con la siguiente formula se obtendria la intensidad del color ya que la misma representa cuanto tiempo del periodo estara en alto la señal.



Si queremos encender un color al máximo OCR0 debería valer 255 de modo tal que duty cycle sea 0 y la señal se encuentre siempre en bajo.

Generación de PWM para rojo por software:

Para este color se debe utilizar la salida PB5 del MCU y a diferencia de los otros colores esta salida no cuenta con PWM, por lo que la señal se debe generar por software de manera bloqueante.

Como periodo máximo fue utilizado por simplicidad el contador máximo es decir OCR0A=255 resultando una señal de periodo 16.384ms y frecuencia 61.03Hz.

Para saber cuánto tiempo se debería dejar en bajo y en alto la señal se realizaron las siguientes formulas:



Donde T hace referencia a el periodo total de la señal y rojo a la intensidad del color.

Y para conseguir los valores de OCRA0 se realizó las siguientes formulas:

Señal OCRA0 en bajo:



Reemplazando la formula de t\_bajo se obtuvo



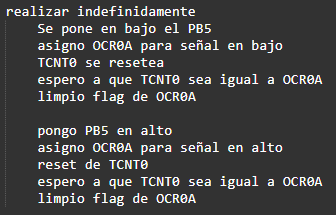
Señal OCRA0 en alto:



Reemplazando formula t\_bajo



De esta forma asignando TCNT0 en 0 cada vez que se deba contar la señal, una vez para bajo y otra para alto se obtiene la señal de intensidad pedida. Se adjunta pseudocodigo de esto en la imagen inferior.



**Uso de terminal**

Para esto se utilizó la misma librería del anterior TP (SerialPort.h junto a SerialPort.c) por lo que se utilizó también del periférico interno al MCU (UART0). Para enviar mensajes a la consola utilizamos del metodo bloqueante SerialPort\_Send\_String(Char \*); para enviar el menú de opciones y SerialPort\_Send\_uint8\_t(uint8\_t); para enviar el valor elegido del potenciómetro. Se utilizo de manera bloqueante ya que no se creyó necesario el uso de interrupciones para la transmisión en este caso debido a que no afecta a la funcionalidad y simplifica el programa. En cambio, para la lectura la misma fue realizada como en el anterior TP, de manera no bloqueante y siguiendo una estructura foreground/background.

Los comandos programados para su uso en la terminal fueron los siguientes:

R <Enter>: configuración del color rojo.

V <Enter>: configuración del color verde.

A<Enter>: configuración del color azul.

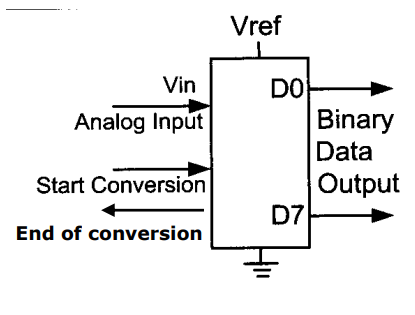
S<Enter>: asigno valor elegido para el potenciómetro en caso de encontrarse en alguno de los casos anteriores.

En caso de encontrarse en el modo R, V o A se imprime cada 1 segundo el valor del potenciómetro.

Y en caso de no recibir ningún comando correcto se ignora el mismo siguiendo en el modo en el que se encuentre.

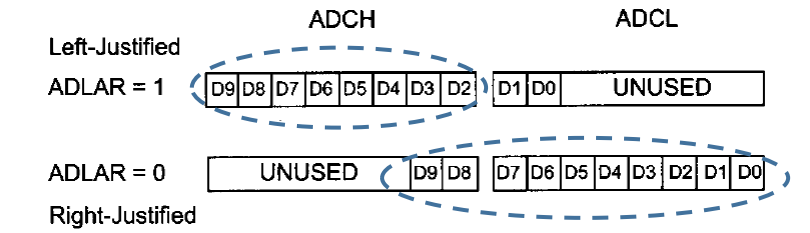
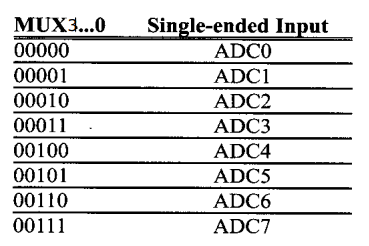
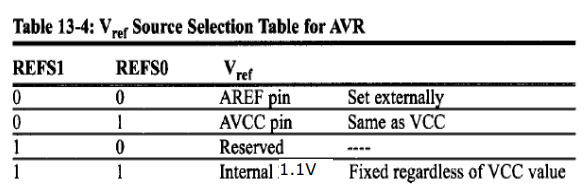
**Uso de Periferico ADC**

Para la utilización del potenciómetro se hizo uso de un conversor Analógico - Digital que nos permitió convertir magnitudes analógicas a valores digitales para procesarse en el MCU y con este enviarlos al LED. Se puede observar un esquema del periférico en la imagen inferior.

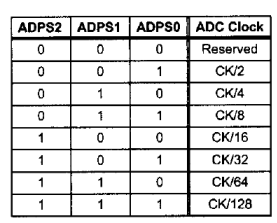


En el enunciado se pidió el uso del ADC3 y para ello se configuro el mismo de la siguiente manera:

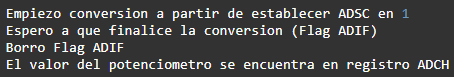
En ADMUX se configure en 1 ADLAR (esto significa que se utilizan solo 8 bits para la conversión siendo estos la parte alta de ADC), en 1 MUX1 y en 1 MUX0 (Seleccionando de esta manera ADC3), y poniendo refs0 en 0 y refs1 en 0 para utilizar el voltaje VCC como referencia. En las siguientes imágenes se puede observar las diferentes configuraciones para los registros y también como se utilizó la parte alta del registro ADC para simplificación para codificar los datos.



Por ultimo se configuro tambien el registro ADCSRA en el cual se establecio el bit ADEN en 1 (con tal de habilitar el conversor analogico-digital) y los bits ADPS2, ADPS1 y ADPS0 en 1 para establecer el reloj con un preescalador de 128 consiguiendo una frecuencia de 125KHz obteniendo por consiguiente una mejor perfomance del conversor. En la siguiente imagen se observa las distintas configuraciones de preescalador.

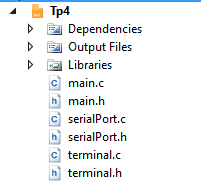


Una vez configurado los anteriores registros ya se pudo utilizar el conversor y esto se hizo de la siguiente manera:



**Modularizacion**

El programa fue modularizado como se puede observar en la siguiente imagen:



* Main hace referencia al programa principal donde se realizan las tareas background es decir al llegar un comando se configuraban los colores.
* serialPort es la librería brindada por la catedra para configurar y utilizar el puerto serie del MCU.
* Terminal hace referencia a la librería donde hay un método que se encarga de procesar el comando presente en el buffer(Terminal\_leerComando()), otro metodo para inicializar los registros del MCU (Terminal\_configurarRegs()), la rutina de interrupción para la recepción de datos y un ultimo metodo para la lectura del potenciómetro (Terminal\_leerPotenciometro()).

**Validacion:**