UNIVERSIDAD
AUTÓNOMA
DE QUERÉTARO

FACULTAD
DE INGENIERÍA

PRÁCTICA 5: PWM
MICROCONTROLADORES

RAMÍREZ ÁLVAREZ CARLO IVÁN - 280847
ONTIVEROS MARTÍNEZ BEATRIZ - 244784
TREJO DOMÍNGUEZ NELLY BIBIANA - 242494

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
INGENIERÍA BIOMÉDICA

MICROCONTROLADORES
ING. JOSÉ DE JESÚS SANTANA RAMÍREZ
ENERO, 2023

I. OBJETIVOS

- Aprender a controlar el módulo PWM.
- Generar una señal PWM modificando el ciclo de trabajo por medio de un potenciómetro.

II. MARCO TEÓRICO

a. Modulación por ancho de pulso (PWM)

La modulación por ancho de pulso (PWM) es una técnica poderosa para codificar digitalmente niveles de señales analógicas.

Se utilizan contadores de alta resolución para generar una onda cuadrada y el ciclo de trabajo de la onda cuadrada se modula para codificar una señal analógica. Las aplicaciones típicas incluyen fuentes de alimentación conmutadas y control de motores.

El microcontrolador TM4C123GH6PM contiene dos módulos PWM, cada uno con cuatro bloques generadores PWM y un bloque de control, para un total de 16 salidas PWM. El bloque de control determina la polaridad de las señales PWM y qué señales pasan a los pines. Cada bloque generador PWM produce dos señales PWM que comparten el mismo temporizador y frecuencia y pueden programarse con acciones independientes o como un solo par de señales complementarias con retrasos de banda muerta insertados.Cada módulo PWM TM4C123GH6PM proporciona una gran flexibilidad y puede generar señales PWM simples y PWM emparejadas.

b. Descripción funcional

El PWM tiene dos opciones de fuente de reloj:

- El reloj del sistema
- Un reloj del sistema pre-dividido

La fuente de reloj se selecciona programando el bit USPWMDIV en el registro de configuración de reloj en modo de ejecución (RCC) en el desplazamiento de

control del sistema 0x060. El campo de bits PWMDIV especifica el divisor del reloj del sistema que se utiliza para crear el reloj PWM.

c. Ciclo de trabajo

El ciclo de trabajo describe la cantidad de tiempo que la señal está en un estado alto como un porcentaje del tiempo total que se tarda en completar un ciclo.

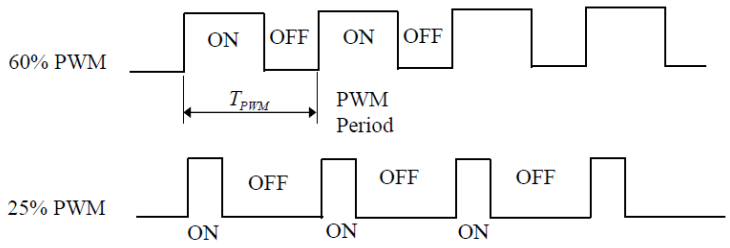

Fig 1. Modulación por ancho de pulso

III. MATERIALES

- osciloscopio
- servomotores SG90
- transistores BC548
- resistencias
- led RGB
- potenciometros

IV. DESARROLLO DE LA PRÁCTICA Y RESULTADOS

i. Experimento 1

Usando el módulo 0 de PWM con una frecuencia de reloj del sistema de 50,000,000 Hz junto con el generador 1 habilitar alguno de los PWM's asociados y obtener un PWM cuya frecuencia sea de 10 KHz.

```
duty0 = 10;  //%% para EXP1
duty1 = 0;  //%%
duty2 = 100;  //%%
```

Fig 2. Definimos nuestro ciclo de trabajo en la figura 5 se usó duty0.

Es importante mencionar que para obtener lo solicitado se realizaron los siguientes cálculos $\frac{\frac{1}{10000}}{\frac{1}{50000000}} = 5000$ cuentas.

```
51
52      SYSCTL->RCC &= ~(0x000e0000); //limpio
53      SYSCTL->RCC |= (1<<20); //Para indicarle que
54      SYSCTL->RCC |= (0x3<<17); //si se dividiera:
55
```

Fig 3. Se habilita el número que se divide entre 16.

En la línea 54, se observa que: se habilita que las cuentas se dividan sobre 16 pues en el experimento 2 y 3 la frecuencia de 50 Hz era muy pequeña y las cuentas muy grandes ocasionando que fallara el programa. Durante el experimento uno se quedó este divisor pues no afectaba al funcionamiento del programa.

```
62
63      PWM0->_1_LOAD = 312.5-1;
64
```

Fig 4. Instrucción para definir la carga del PWM.

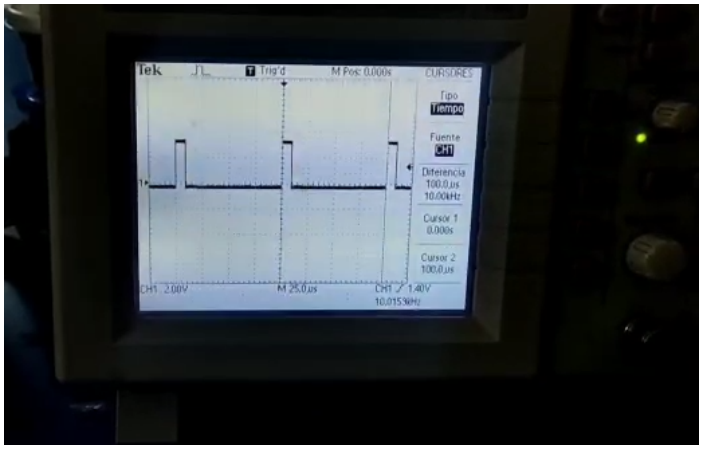

Fig 5. Obtención de los 10 KHz.

ii. Experimento 2

Usando el módulo 0 de PWM con una frecuencia de reloj del sistema de 20,000,000 Hz junto con el generador 0,1,2 habilitar alguno de los PWM's

asociados y obtener un PWM cuya frecuencia sea de 50Hz con tres potenciómetros variar el ciclo de trabajo para controlar la posición de tres servos sg90 u otros.

Los servos tienen un ciclo de trabajo de 0.5 a 2.5 ms como la tiva maneja 50Hz necesitamos bajar el ciclo de trabajo y al tener un comparador de bajada necesitamos restarle. Tenemos el equivalente de milisegundos de 0.5 y 2.5 ms es decir, 600 y 2550 cuentas respectivamente, al restal al ciclo de trabajo que es de 20 ms obtenemos que el estado bajo queda en un máximo de 17.5 ms y el máximo del ciclo de trabajo de 0.5 ms, como se muestra en la Fig 6.

```
55 ////////////////////          .5ms    2.5ms  -  convercion de 0 a 4095
56   PWM0->_0_CMPA = PWM0->_0_LOAD - 600 - 2550*((duty0/4095.0));
57   PWM0->_1_CMPB = PWM0->_1_LOAD - 600 - 2550*((duty1/4095.0));
58   PWM0->_2_CMPB = PWM0->_2_LOAD - 600 - 2550*((duty2/4095.0));
59
```

Fig 6. Configuración de cada servo para su determinado generador.

Para la parte electrónica (Fig 7)se utilizó un arduino mega como fuente de voltaje de 5V, conectamos los pines E1, E2 y E3 y los PWM0 B6, B5 y E5.

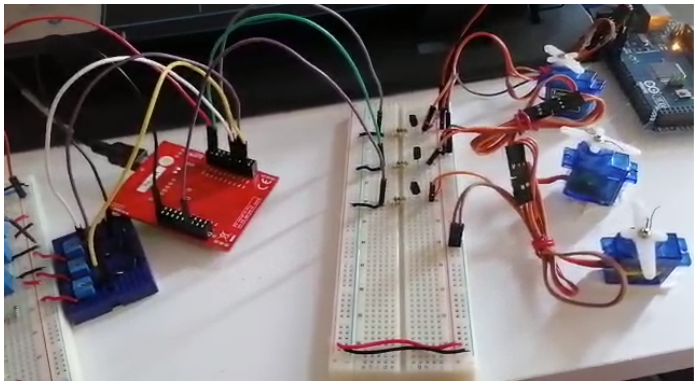

Fig 7. Circuito electrónico experimento 2.

iii. Experimento 3

Usando el módulo 0 de PWM con una frecuencia de reloj del sistema de 20,000,000 Hz junto con el generador 0,1 y 2 habilitar alguno de los PWM's

asociados y obtener un PWM cuya frecuencia sea de 50Hz , utilizando el uart de la práctica 3 se enviará dato desde interfaz de simulink para controlar la intensidad luminosa usando un led RGB externa.

Gracias al UART es posible recibir una cadena de caracteres, en el que definirá el color principal, ya sea “r” para rojo, “g” para verde o “b” para azul.
En la Fig 7 observamos que la función *atoi* nos permitirá convertir una cadena a un valor numérico para así obtener un factor que definirá la máxima ó mínima potencia del led al modificar el comparador asociado y así editar el ciclo de trabajo.

```
74
75        char c;
76      c= Rx_char();
77
78        switch(c)
79        {
80
81        case 'r': {  ///led r
82            data_str[0] = Rx_char();
83
84            data_str[1] = Rx_char();
85
86            data_str[2] = Rx_char();
87
88           data_str[3] = '\0';
89           uint32_t duty0= atoi(data_str);
90
91            dc1= (100*duty0)/255.0;
92
93  PWM0->_0_CMPA = PWM0->_0_LOAD - (249.99*dc1) -1;  /// 0
```

Fig 8. Definición del color según el primer caracter enviado para ‘r’, ‘g’ y ‘b’.

Finalmente, si se ingresan datos que no corresponden ya sea, una letra erronea o datos errantes como lo sería el mandar más o menos números de los solicitados se enviará a 0% todos los ciclos de trabajo provocando que los leds se apaguen o se mantengan inactivos.

```
134                 }
135     default:
136
137         dc=0;
138         PWM0->_0_CMPA =PWM0->_0_LOAD - (249.99*dc) -1;;
139         PWM0->_1_CMPB =PWM0->_1_LOAD - (249.99*dc) -1;;
140         PWM0->_2_CMPB = PWM0->_2_LOAD - (249.99*dc) -1;;
141
142         break;
143
```

Fig 9. Caso para caracteres no identificados

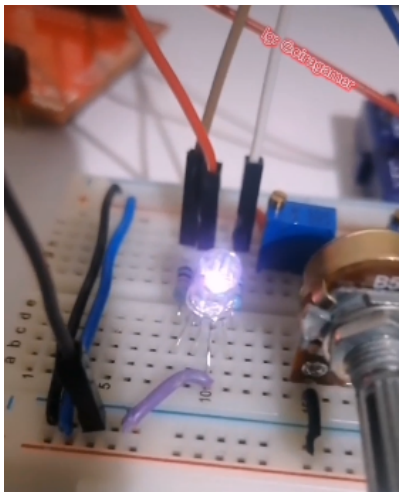

Fig 10. Envío de datos g025, r115 y b035 de para obtener una tonalidad morada.

Se realizó un script (Fig 11) en MatLab que envía las instrucciones, como primer paso se solicita los valores del LED de la manera correcta y después se activa el puerto, donde una vez activado gracias a las funciones envía la variable correspondiente a “r”, “g” o “b” (el valor) que se dirige al código principal.

```
function RGB
clc;

fprintf('Controlador de intensidad de un led RGB\n');
fprintf('Intensidad desde 0 hasta 255, las x representan numeros enteros\n ');
R=input('Coloca valor del led rojo de la forma rxxx:','s');
G=input('Coloca valor del led verde de la forma gxxx:','s');
B=input('Coloca valor del led azul de la forma bxxx:','s');

puerto = serialport("COM6",19200);

fwrite(puerto,R)
fwrite(puerto,G)
fwrite(puerto,B)

 fclose (puerto) ;
 delete (puerto) ;

end
```

Fig 11. Script en MatLab.

V. BIBLIOGRAFÍA

Texas Instruments Incorporated. (2007). *Tiva™ TM4C123GH6PM Microcontroller - Data Sheet*. -. Recuperado 20 de enero de 2023, de https://www.ti.com/lit/ds/spms376e/spms376e.pdf?ts=1646079818207&ref_url=https%253A%252F%252Fwww.google.com%252F