PRÁCTICA 5

Señales PWM con TM4C123C (tiva chica).

Universidad Autónoma de Querétaro

Microcontroladores.

Nieto Guerrero Andrea

29 de Enero del 2023

Metodología Práctica 5.

Experimento 1.

Para el experimento 1 se pide configurar el módulo cero con una frecuencia de 50Mhz y obtener un PWM junto el generador 1 cuya frecuencia sea de 10Khz.

Configuración PWM.

Lo primero que se debe hacer es activar el reloj para el módulo 0, para ello vamos a la página 355 de la datasheet.

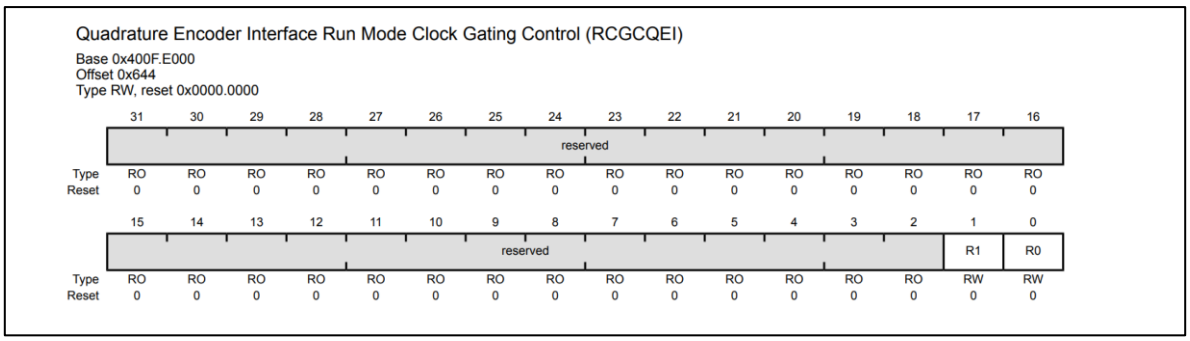


Figura 1. Registro RCGCPWM

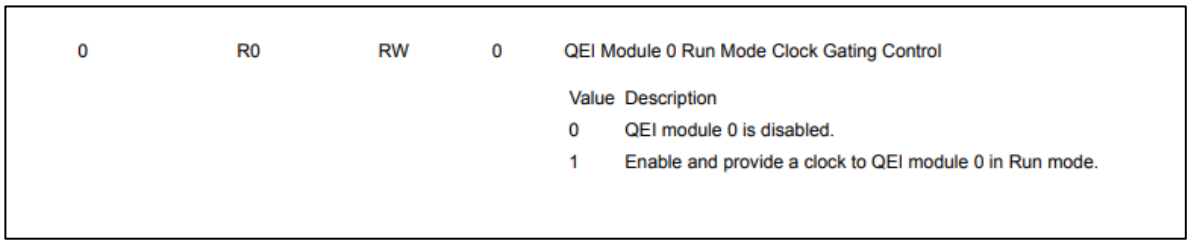


Figura 2. Bit 0 del registro RCGCPWM

La cual nos indica como se muestra en la figura 2, que se debe escribir un 1 en la posición cero para habilitar el reloj:

```
SYSCTL->RCGCPWM|= (1<<0);
```

Figura 3. Asignación del registro RCGCPWM en Visual Studio

Ahora es necesario habilitar el reloj para el puerto que vamos a utilizar, en la página 1233 de la datasheet nos indica que el pin correspondiente al módulo 0, generador 1 es el pin B4, por lo que el puerto que se va activar es el B.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
M0FAULT0	30 53 63	PF2 (4) PD6 (4) PD2 (4)	I	TTL	Motion Control Module 0 PWM Fault 0.
M0PWM0	1	PB6 (4)	O	TTL	Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0.
M0PWM1	4	PB7 (4)	O	TTL	Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0.
M0PWM2	58	PB4 (4)	O	TTL	Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1.

Figura4. Descripción de las señales PWM

El registro que activa el reloj de los puertos es RCCGPIO, lo encontramos en la página 340 de la datasheet, como se muestra en la imagen

General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)

Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 5. Registro RCGCGPIO

1	R1	RW	0	GPIO Port B Run Mode Clock Gating Control

Value	Description
0	GPIO Port B is disabled.
1	Enable and provide a clock to GPIO Port B in Run mode.

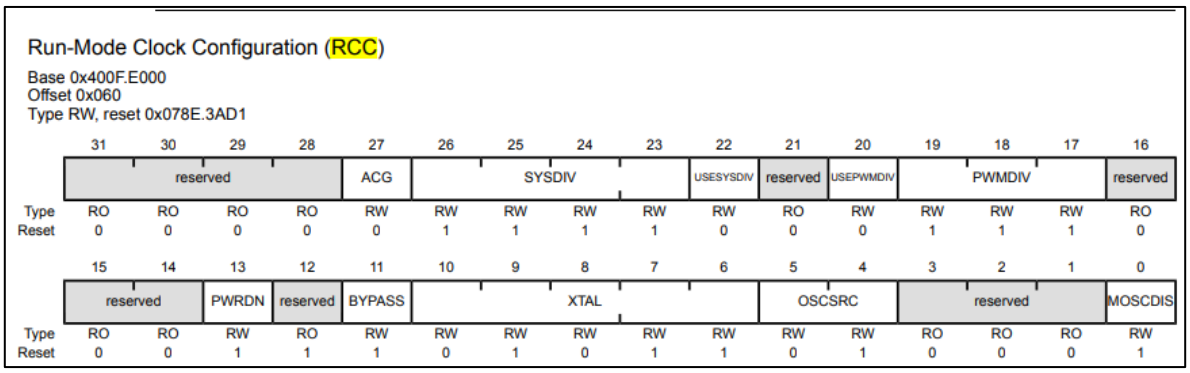
Figura 6. Bit 1 del registro RCGCGPIO

Como se muestra en la figura 6, para habilitar el reloj en el puerto B, asignamos un 1 al bit 1.

```
SYSCTL->RCGCGPIO |= (1<<1);//Enable reloj de GPIO Puerto B pag 340
```

Figura 7. Asignación del registro RCGCGPIO en Visual Studio

Como siguiente paso, vamos a indicar la forma de funcionamiento del reloj, para ello usaremos el registro RCC, el cual se encuentra en la página 254 de la datasheet:



Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

Figura 8. Registro RCC

Inicialmente se habilita el divisor del PWM, este valor se asigna en el pin 20 del registro

20	USEPWMDIV	RW	0	Enable PWM Clock Divisor

Value	Description
0	The system clock is the source for the PWM clock.
1	The PWM clock divider is the source for the PWM clock.

Note that when the PWM divisor is used, it is applied to the clock for both PWM modules.

Figura 9. BIT 20 del registro RCC

A continuación, mediante los bits 17, 18 y 19 asignamos el valor entre el cual se va dividir el reloj:

19:17	PWMDIV	RW	0x7	PWM Unit Clock Divisor

This field specifies the binary divisor used to predivide the system clock down for use as the timing reference for the PWM module. The rising edge of this clock is synchronous with the system clock.

Value	Divisor
0x0	/2
0x1	/4
0x2	/8
0x3	/16
0x4	/32
0x5	/64
0x6	/64
0x7	/64 (default)

Figura 10. BIT 17,18 y 19 del registro RCC

En este caso se requiere dividir/2, lo que la figura 10 nos indica es que se debe asignar un 0 para estos 3 bits:

```
SYSCTL->RCC &=0xFFF0FFFF;
```

Figura 11. Asignación del registro RCC en Visual Studio

Ahora, se va configurar el registro AFSEL para indicarle al puerto que va funcionar con una de las funciones alternas.

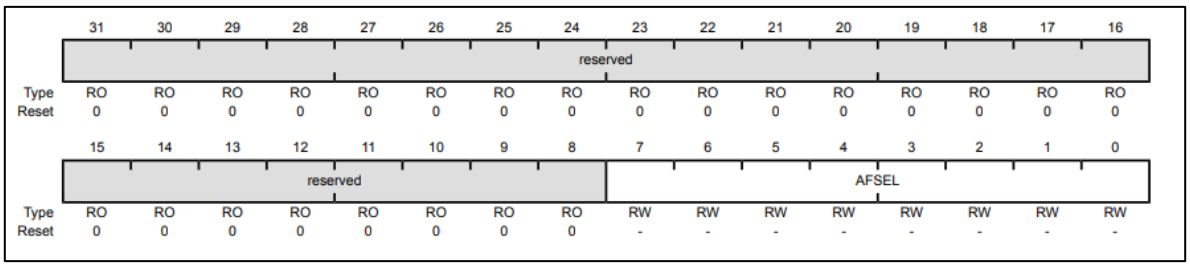


Figura 12. Registro AFSEL

7:0	AFSEL	RW	-	GPIO Alternate Function Select

Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Figura 13. BIT 0-7 del registro AFSEL

El pin al que le queremos asignar una función es el B4, por lo que se asigna un 1 en el bit 4 como se muestra en la figura 14.

```
GPIOB->AFSEL |= (1<<4); //Función alterna para B4 y B6 Pag 672/
```

Figura 14. Asignación del registro AFSEL en Visual Studio.

Ya indicado que el pin B4 va funcionar como función alterna, se le asigna la función alterna a utilizar, en este caso PWM.

En la página 1351 se presenta una tabla (figura 15) la cual describe las funciones alternas de cada uno de los pines, la función PWM es la función alterna #4, valor que será asignado en el registro PCTL.

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	CAN1Rx	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	CAN1Tx	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-
PB0	45	USB0ID	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-

Figura 15. Tabla de funciones alternas de los puertos.

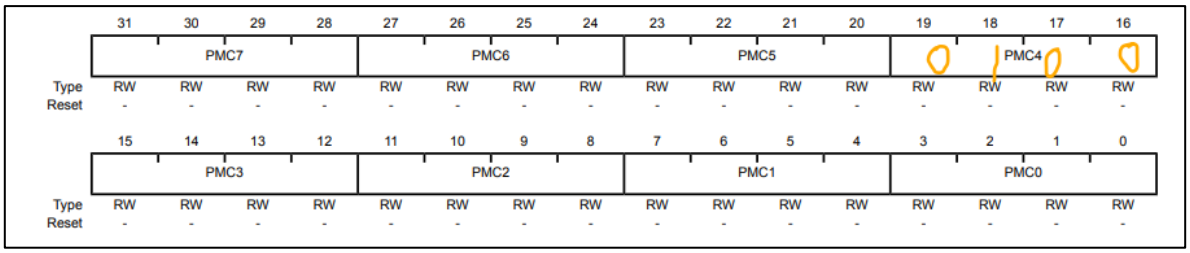


Figura 16. Registro PCTL.

Ya que la figura 15 nos muestra que la función alterna correspondiente al PWM es la #4, asignamos un 1 (0100=4 hexadecimal) en el bit 18, el cual el cual corresponde al pin B4. Adicionalmente se agrega una máscara, debido a que como se muestra en la figura 16, el registro PCTL, no cuenta con valores en el reset.

```
GPIOB->PCTL |=(0xFFF0FFFF) | 0x00040000;
```

Figura 17. Asignación del registro PCTL en Visual Studio.

Ahora le indicamos al puerto B mediante el registro DEN, página 683, que el puerto B4 va ser un puerto digital.

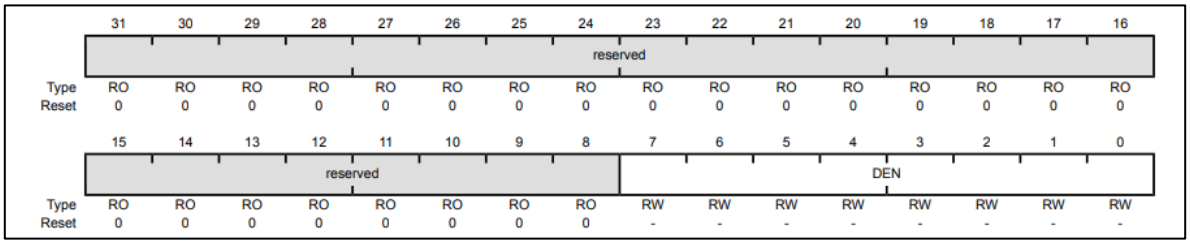


Figura 18. Registro DEN

7:0	DEN	RW	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Figura 19. Bit 0-7 del registro DEN.

Como se muestra en la figura 19, para que funcione como puerto digital, es necesario asignar un 1, en este caso al pin B4 (bit 4), como se muestra a continuación:

```
GPIOB->DEN |= (1<<4);
```

Figura 20. Asignación del registro DEN en Visual Studio.

A continuación, configuramos los registros correspondientes al PWM:

Comenzamos por el registro PWMnCTL, en este caso el generador que necesitamos es el uno por lo que su nomenclatura es PWM1CTL, el cual se encuentra en la página 1266.

PWMn Control (PWMnCTL)

PWM0 base: 0x4002.8000
PWM1 base: 0x4002.9000
Offset 0x040
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved													LATCH	MINFLTPER	FLTSRC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DBFALLUPD		DBRISEUPD		DBCTLUPD		GENBUPD		GENAUPD		CMPBUPD	CMPAUPD	LOADUPD	DEBUG	MODE	ENABLE
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 21. Registro PWMnCTL

0	ENABLE	RW	0	PWM Block Enable

Note: Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the **PWMnCOUNT** register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the **SRPWM** register in the System Control Module.

Value	Description
0	The entire PWM generation block is disabled and not clocked.
1	The PWM generation block is enabled and produces PWM signals.

Figura 22. Bit 0 del registro ENABLE.

Donde para configurarlo, inicialmente es necesario deshabilitarlo, como se muestra en la figura 22, asignamos un 0 en el bit 0. En este caso lo escribimos como un 1 negado, como se muestra a continuación:

```
PWM0->_1_CTL &=~(1<<0); //1266 Deshabilitamos registro CTL_generador1
```

Figura 23. Asignación del registro PWM1CTL en Visual Studio

Establecemos las características de uno de los generadores, en este caso el A.

PWMn Generator A Control (PWMnGENA)

PWM0 base: 0x4002.8000
PWM1 base: 0x4002.9000
Offset 0x060
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				ACTCMPBD		ACTCMPBU		ACTCMPAD		ACTCMPAU		ACTLOAD		ACTZERO	
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Figura 24. Registro PWMnGENA

Las características que requerimos son las siguientes: Una señal PWM con el generador 1, salida A, teniendo un contador descendente, cuando el contador llegue a un valor igual al valor del comparador, la señal pwm se pondrá en LOW. Para obtenerla escribimos el valor que se muestra en la figura 25.

```
PWM0->_1_GENA=0x0000008C;
```

Figura 25. Asignación del registro PWMnGENA en Visual Studio

Ahora establecemos la carga correspondiente al valor de PWM deseado mediante el registro PWMnLOAD en la página 1278. El valor de carga máximo que se puede escribir en este registro es de 65,536, valor correspondiente a los 16 bits asignados para ese registro.

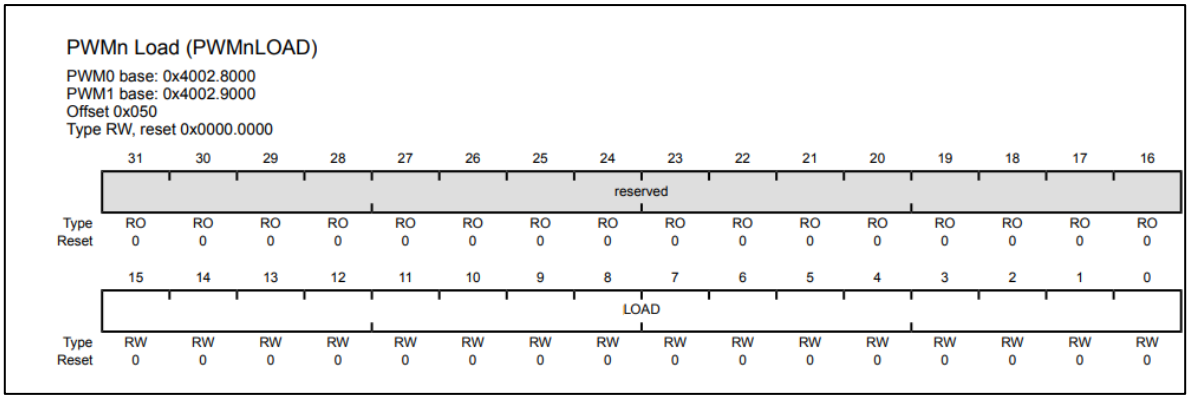


Figura 26. Registro PWMnLOAD

En este caso, se asigna un valor de 2500, porque como se describió anteriormente, el reloj de 50Mhz se dividió entre 2 lo que nos da un reloj de 25Mhz, resultado que es dividido entre la frecuencia del PWM (10Khz), obteniendo un valor de 2500.

```
PWM0->_1_LOAD = 2500; //   para 10khz cuentas = (25,000,000/10000)/
```

Figura 27. Asignación del registro PWM1LOAD en Visual Studio

El siguiente registro por configurar es el CMPA, valor que se debe comparar con el contador, cuando este valor coincide con el contador, se emite un pulso que genera las señales con el duty que se establezca. El registro se encuentra en la página 1280.

El valor de carga máximo que se puede escribir en este registro es de 65,536, valor correspondiente a los 16 bits asignados para ese registro.

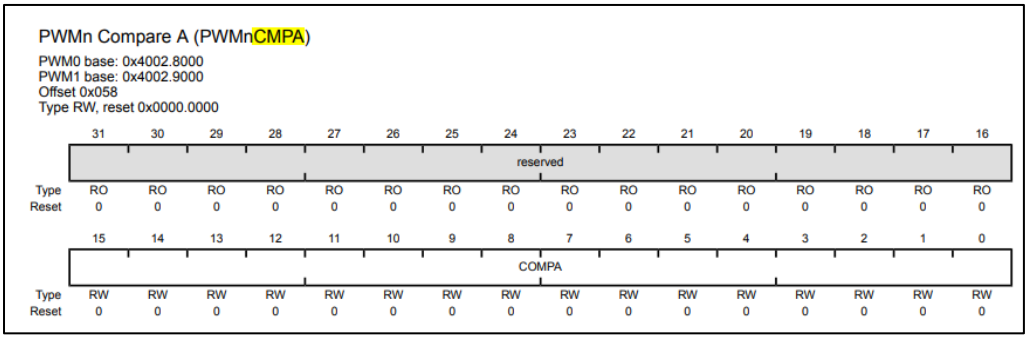


Figura 28. Registro PWMnCMPA.

15:0	COMPA	RW	0x00	Comparator A Value The value to be compared against the counter.

Figura 29. Bit del 0-15 del registro PWMnCMPA.

Ya que se establecieron todos los valores para configurar el PWM, activamos los bloques de generación del PWM mediante el registro PWMnCTL.

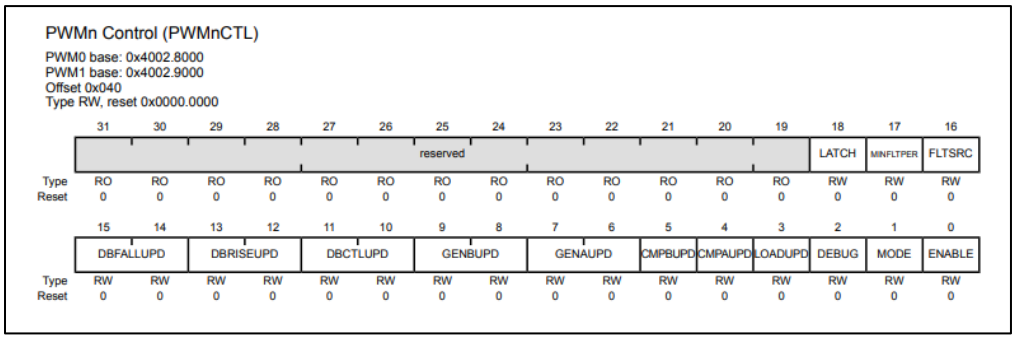


Figura 30. Registro PWMnCTL.

0	ENABLE	RW	0	PWM Block Enable **Note:** Disabling the PWM by clearing the ENABLE bit does not clear the COUNT field of the **PWMnCOUNT** register. Before re-enabling the PWM (ENABLE = 0x1), the COUNT field should be cleared by resetting the PWM registers through the **SRPWM** register in the System Control Module. Value Description 0 The entire PWM generation block is disabled and not clocked. 1 The PWM generation block is enabled and produces PWM signals.

Figura 31. Bit 0 del registro PWMnCTL.

Para activar el bloque de generación (generador 1) del PWM, asignamos un 1 en el bit 0 del registro PWMnCTL como se muestra en la figura 32.

```
PWM0->_1_CTL = (1<<0);// Se activa el generador 1
```

Figura 32. Asignación del registro PWMnCTL en Visual Studio

Para terminar con la configuración del PWM, habilitamos su salida mediante el registro PWMENABLE, el cual se encuentra en la página 1247.

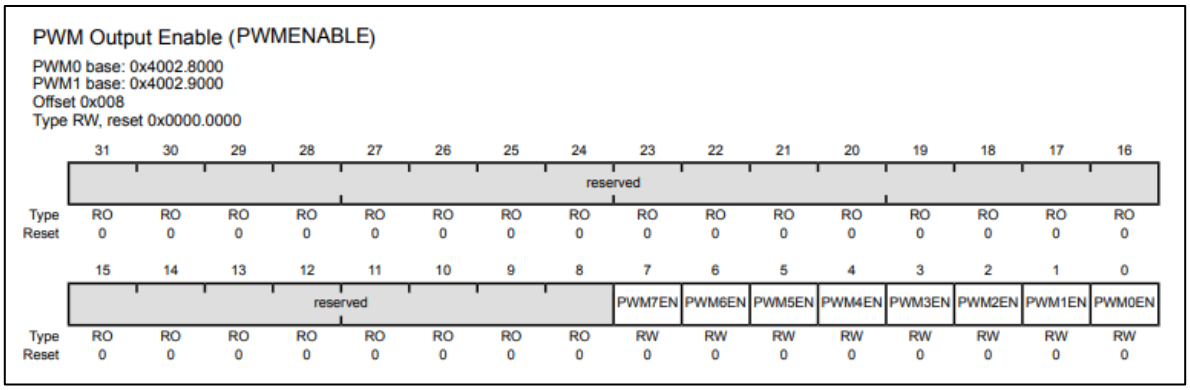


Figura 33. Registro PWMENABLE.

2	PWM2EN	RW	0	MnPWM2 Output Enable

Value	Description
0	The MnPWM2 signal has a zero value.
1	The generated pwm1A' signal is passed to the MnPWM2 pin.

Figura 34. Bit 2 del registro PWMENABLE.

Como se mostró en la figura 4, la señal que deseamos activar lleva el nombre de M0PWM2, el bit correspondiente para esta señal es el #2, por lo que asignamos un valor de 1 en el bit 2, como se muestra en la figura 35:

```
PWM0->ENABLE |= (1<<2); //habilitar el bloque
```

Figura 35. Asignación del registro PWMENABLE en Visual Studio

Las indicaciones para el experimento 1 solicita únicamente la generación del PWM de 10KHz con un reloj de 50Mhz, por lo que, en el programa principal, solo incluimos las librerías e invocamos la función correspondiente al PWM y PLL como se muestra en la figura 36. Así como un ciclo infinito para que la señal se genere sin pausas.

```
#include "lib/include.h"
int main(void)
{
    Configurar_PLL(_50MHZ);  //Confiuracion de velocidad de reloj EXP
    Configura_Reg_PWM1(10000);//Configuro a 10khz el pwm



    while(1)
    {


    }
}
```

Figura 36. Función main del programa.

Experimento 2.

Usando el módulo 0 de PWM con una frecuencia de reloj del sistema de 20,000,000Hz 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.

Configuración PWM.

Para la configuración del PWM de este experimento se deben llevar a cabo los mismos pasos que en el experimento anterior, pero para 3 señales. Las señales que se seleccionaron fueron M0PWM0 (PB6), M0PWM2 (PB4) y M0PWM4 (E4) como se muestra en la figura 37.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
M0FAULT0	30 53 63	PF2 (4) PD6 (4) PD2 (4)	I	TTL	Motion Control Module 0 PWM Fault 0.
M0PWM0	1	PB6 (4)	O	TTL	Motion Control Module 0 PWM 0. This signal is controlled by Module 0 PWM Generator 0.
M0PWM1	4	PB7 (4)	O	TTL	Motion Control Module 0 PWM 1. This signal is controlled by Module 0 PWM Generator 0.
M0PWM2	58	PB4 (4)	O	TTL	Motion Control Module 0 PWM 2. This signal is controlled by Module 0 PWM Generator 1.
M0PWM3	57	PB5 (4)	O	TTL	Motion Control Module 0 PWM 3. This signal is controlled by Module 0 PWM Generator 1.
M0PWM4	59	PE4 (4)	O	TTL	Motion Control Module 0 PWM 4. This signal is controlled by Module 0 PWM Generator 2.

Figura 37. Descripción de las señales PWM

1. *Habilitar reloj para el PWM.*

```
SYSCTL->RCGCPWM |= (1<<0); //Enable reloj
```

Figura 38. Asignación del registro RCGCPWM en Visual Studio

2. *Habilitar el reloj para el puerto B y E*

```
SYSCTL->RCGCGPIO |= (1<<1);//Enable reloj de GPIO Puerto B
```

Figura 39. Asignación del registro RCGCGPIO puerto B en Visual Studio

```
SYSCTL->RCGCGPIO |= (1<<4);//Enable reloj de GPIO Puerto E
```

Figura 40. Asignación del registro RCGCGPIO puerto E en Visual Studio

3. *Habilitar el divisor del reloj PWM*

```
SYSCTL->RCC &=0xFFF4FFFF; //Divisor/8
```

Figura 41. Asignación del registro RCC en Visual Studio

4. *Establecer el divisor de reloj de la unidad PWM*

   En este caso, para el registro RCC, se debe establecer un valor de 0x2 en los bits 17-19, para que el valor de cargas correspondiente en este experimento quepa en 16 bits, dividiendo así los 20Mhz entre 8. Pero como el valor de este registro se escribe nibble por nibble, se debe escribir un 0x4 en el programa, dado que el bit 16 es parte de este nibble a pesar de que este reservado.

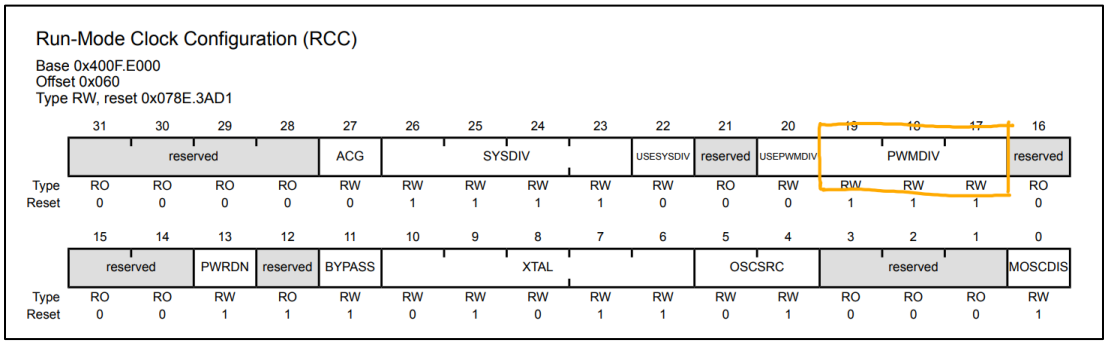


Figura 42. Registro RCC.

19:17	PWMDIV	RW	0x7	PWM Unit Clock Divisor This field specifies the binary divisor used to predivide the system clock down for use as the timing reference for the PWM module. The rising edge of this clock is synchronous with the system clock. Value Divisor 0x0 /2 0x1 /4 0x2 /8 0x3 /16 0x4 /32 0x5 /64 0x6 /64 0x7 /64 (default)

Figura 43. Bit 17-19 del registro RCC.

```
SYSCTL->RCC &=0xFFF4FFFF;
```

Figura 44. Asignación del registro RCC en Visual Studio

5. Establecer función alterna para los pines B4, B6 y E4.

```
GPIOB->AFSEL |= (1<<4)| (1<<6); //Función alterna para B4 y B6 Pag 672
```

Figura 45. Asignación del registro RCC, puerto B en Visual Studio

```
GPIOE->AFSEL |= (1<<4); //Función alterna para E4 Pag 672
```

Figura 46. Asignación del registro RCC, puerto E en Visual Studio

6. Establecer el PWM como función alterna para B4, B6 Y E4.

Mediante el registro PCTL, se establece la función alterna a utilizar, en este caso, para los 3 puertos, la señal PWM, es la número 4, como se muestra en la figura 43:

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-
PE4	59	AIN9	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-

Figura 47. Tabla de funciones alternas de los puertos.

```
GPIOB->PCTL |=(0xF0F0FFFF) | 0x04040000;  //Combinado con la tabla Pag 1351
```

Figura 48. Asignación del registro RCC, puerto E en Visual Studio

```
GPIOE->PCTL |=  (0xFFF0FFFF) | 0x00040000;  // Combinado con la tabla Pag 1351
```

Figura 49. Asignación del registro RCC, puerto E en Visual Studio

7. Establecer como pines digitales a B4, B6 y E4.

```
GPIOB->DEN |= (1<<4)|(1<<6); //para decirle si es digital o no
```

Figura 50. Asignación del registro DEN, puerto B en Visual Studio

```
GPIOE->DEN |= (1<<4); //para decirle si es digital o no
```

Figura 51. Asignación del registro DEN, puerto E en Visual Studio

8. Establecemos las características del generador 0, 1 y 2.

```
PWM0->_1_GENA=0x0000008C;
PWM0->_0_GENA=0x0000008C;
```

Figura 52. Asignación del registro PWMnGENA, generador 0 y 1 en Visual Studio

```
PWM0->_2_GENA=0x0000008C;
```

Figura 53. Asignación del registro PWMnGENA, generador 2 en Visual Studio

9. Deshabilitar los bloques de generación de la señal PWM para el generador 0, 1 y 2 de la señal PWM.

```
PWM0->_1_CTL &=~(1<<0); //1266 Deshabilitamos registro CTL_generador1
PWM0->_0_CTL &=~(1<<0); //1266 Deshabilitamos registro CTL_generador0
```

Figura 54. Asignación del registro PWMnCTL, generador 0 y 1 en Visual Studio

```
PWM0->_2_CTL &=~(1<<0); //1266 Deshabilitamos registro CTL_generador2
```

Figura 55. Asignación del registro PWMnCTL, generador 2 en Visual Studio

10. Establecemos la carga para el generador 0, 1 y 2.

```
PWM0->_1_LOAD = 50000;
PWM0->_0_LOAD = 50000;
```

Figura 56. Asignación del registro LOAD, para generador 0 y 1 en Visual Studio

```
PWM0->_2_LOAD = 50000;
```

Figura 57. Asignación del registro LOAD, para generador 2 en Visual Studio

11. Establecemos un valor de comparación para el generador 0, 1 y 2 (no importa cual, ya que se modificará mediante los potenciómetros)

```
PWM0->_1_CMPA = 5000; //

PWM0->_0_CMPA = 5000;
```

Figura 58. Asignación del registro CMPA, para generador 0 y 1 en Visual Studio

```
PWM0->_2_CMPA = 5000;
```

Figura 59. Asignación del registro CMPA, para generador 2 en Visual Studio

12. Habilitar los bloques de generación de la señal PWM para el generador 0, 1 y 2 de la señal PWM.

```
PWM0->_1_CTL = (1<<0);// Se activa el generador 1
PWM0->_0_CTL = (1<<0);// Se activa el generador 0
```

Figura 60. Asignación del registro PWMnCTL, generador 0 y 1 en Visual Studio

```
PWM0->_2_CTL = (1<<0);// Se activa el generador 1
```

Figura 61. Asignación del registro PWMnCTL, generador 2 en Visual Studio

13. Habilitamos la salida de la señal PWM de la señal M0MPW0, M0PWM2 y M0PWM4.

```
PWM0->ENABLE |= (1<<2)|(1<<0); //habilitar el bloque pa que pase Pag 1247
```

Figura 62. Asignación del registro PWMENABLE, señal 0 y 2 en Visual Studio

```
PWM0->ENABLE |=  (1<<4); //habilitar el bloque pa que pase Pag 1247
```

Figura 63. Asignación del registro PWMENABLE, 4 en Visual Studio

Configuración ADC.

El ADC de la tarjeta TM4C123C es de 12 bits de precisión. Para su configuración comenzamos por habilitar el reloj para el ADC mediante el registro RCGCADC, el cual se encuentra en la página 352.

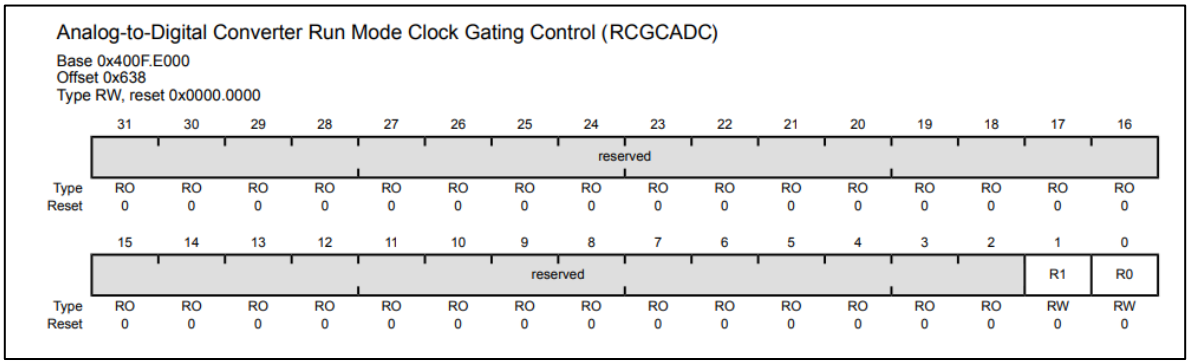


Figura 64. Registro RCGCADC

0	R0	RW	0	ADC Module 0 Run Mode Clock Gating Control

Value	Description
0	ADC module 0 is disabled.
1	Enable and provide a clock to ADC module 0 in Run mode.

Figura 65. Bit 0 del registro RCGCADC.

El módulo que vamos a utilizar es el 0, por lo que se establece un 1 en el bit 0, como se muestra en la figura 66.

```
SYSCTL->RCGCADC |= (1<<0);
```

Figura 66. Asignación del registro RCGCADC en Visual Studio

Las señales analógicas que se van a utilizar son AIN0, AIN1 y AIN2, señales que se encuentran en el puerto E, como se muestra en la figura 67.

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
AIN0	6	PE3	I	Analog	Analog-to-digital converter input 0.
AIN1	7	PE2	I	Analog	Analog-to-digital converter input 1.
AIN2	8	PE1	I	Analog	Analog-to-digital converter input 2.

Figura 67. Señales analógicas.

Ya que sabemos en que puerto van a ingresar las señales analógicas (puerto E), habilitamos el reloj para el puerto mediante el registro RCGCGPIO, que se encuentra en la página 340.

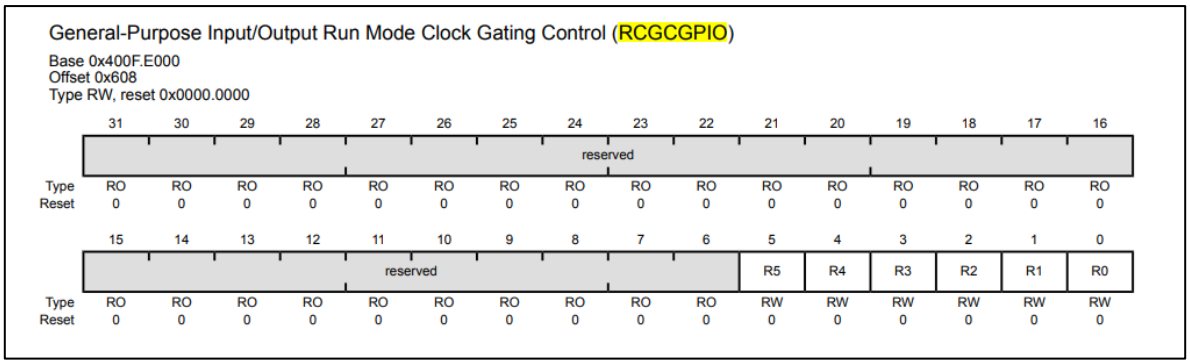


Figura 68. Registro RCGCGPIO.

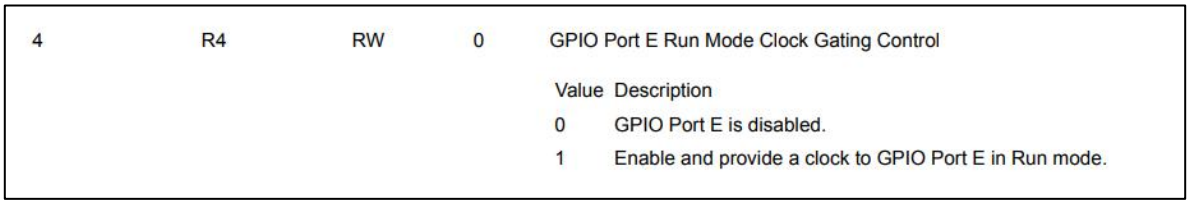


Figura 69. Bit 4 del registro RCGGPIO.

Como se muestra en la figura 69, se asigna un 1 en el puerto E, para habilitar el reloj de este puerto:

```
SYSCTL->RCGCGPIO |= (1<<4);
```

Figura 70. Asignación del registro RCGCGPIO en Visual Studio

Ahora, se va a configurar el registro AFSEL para indicarle al puerto que va funcionar con una de las funciones alternas.

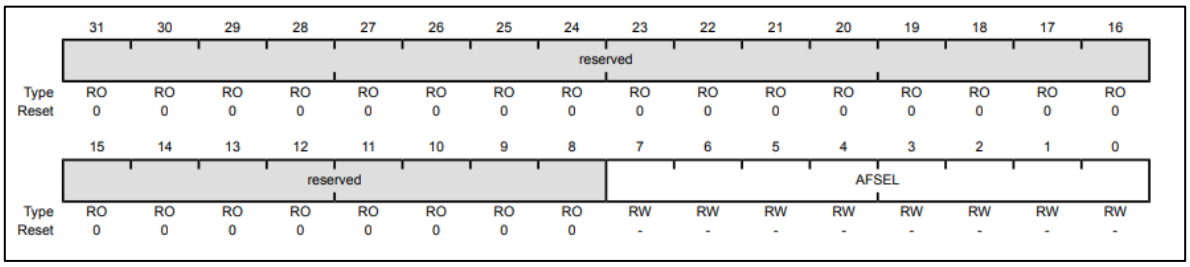


Figura 71. Registro AFSEL

7:0	AFSEL	RW	-	GPIO Alternate Function Select

Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.
	The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Figura 72. BIT 0-7 del registro AFSEL

El pin al que le queremos asignar una función es el E1, E2 y E3, por lo que se asigna un 1 en el bit 0, 1 y 2 como se muestra en la figura 73.

```
GPIOE->AFSEL |= 0x0e;
```

Figura 73. Asignación del registro AFSEL en Visual Studio.

Ahora le indicamos al puerto E mediante el registro DEN, página 683, que el puerto B4 va ser un puerto digital.

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Figura 74. Registro DEN

7:0	DEN	RW	-	Digital Enable

Value	Description
0	The digital functions for the corresponding pin are disabled.
1	The digital functions for the corresponding pin are enabled.
	The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 650.

Figura 75. Bit 0-7 del registro DEN.

Como se muestra en la figura 75, para que funcione como puerto digital, es necesario asignar un 1, en este caso al pin E1, E2 y E3, como se muestra a continuación:

```
GPIOE->DEN &= ~0x0e;
```

Figura 76. Asignación del registro DEN en Visual Studio

Para habilitar la función analógica del pin hacemos uso del registro AMSEL, que se encuentra en la página 687.

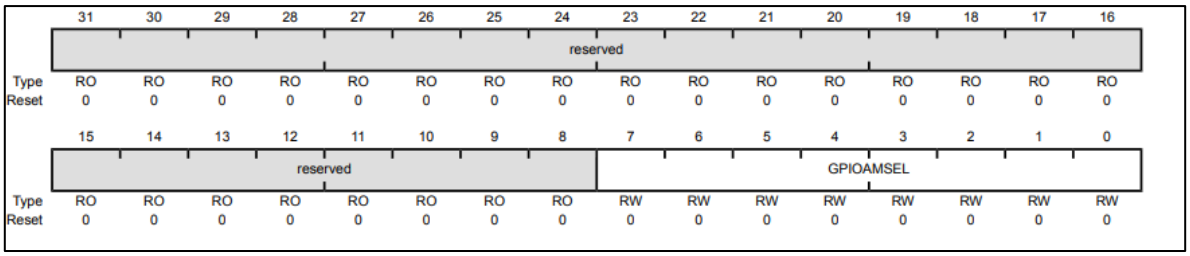


Figura 77. Registro AMSEL.

7:0	GPIOAMSEL	RW	0x00	GPIO Analog Mode Select

Value	Description
0	The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.
1	The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.

Note: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.

The reset state of this register is 0 for all signals.

Figura 78. Bit 0-7 del registro AMSEL.

Como se muestra en la figura 78 es necesario asignar un 1 al pin E1, E2 y E3, como se muestra a continuación

```
GPIOE->AMSEL |= 0x0e;
```

Figura 79. Asignación del registro AMSEL en Visual Studio

Continuamos deshabilitando los secuenciadores de las muestras mediante el registro ADCACTSS, el cual encontramos en la página 821, para así poder configurarlo.

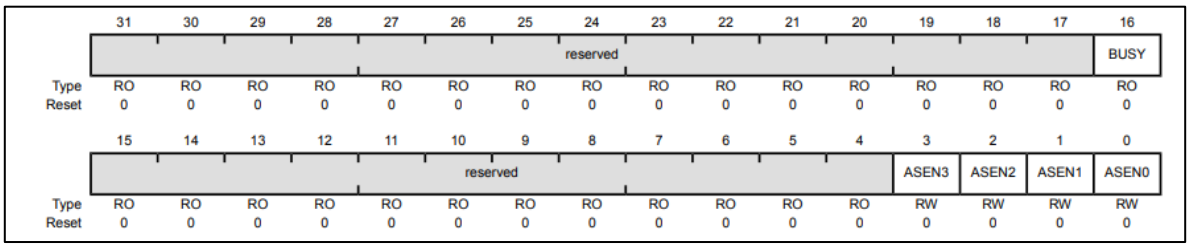


Figura 80. Registro ADCACTSS.

Para deshabilitarlo, asignamos 0 del bit 0 al bit 3.

```
ADC0->ACTSS &= ~(0x0f);
```

Figura 81. Asignación del registro ADCACTSS en Visual Studio.

El siguiente registro al cual se le van a asignar valores es a ADCEMUX, página 833, el cual especifica el evento que se va encargar de iniciar el muestreo para cada secuenciador de muestras

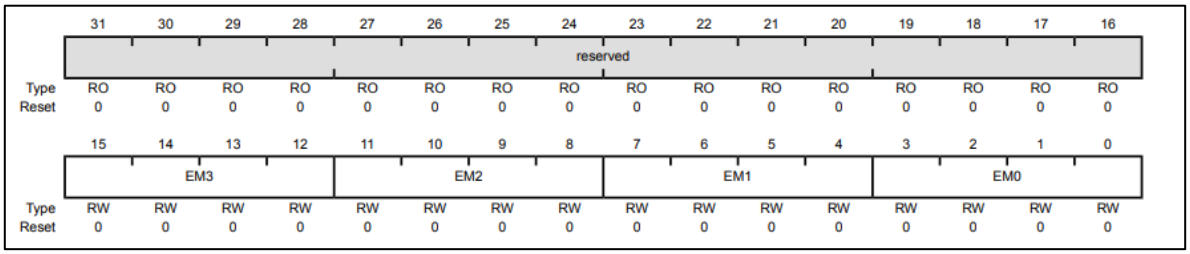


Figura 82. Registro ADCEMUX.

Bit/Field	Name	Type	Reset	Description
15:12	EM3	RW	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are: Value Event 0x0 Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.

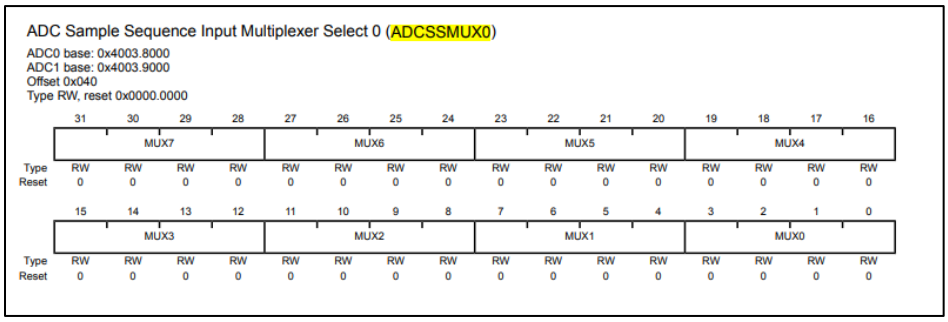
Figura 83. Bit 12-15 del registro ADCEMUX.

En este caso, el evento que va iniciar el muestreo va ser por procesador, el valor correspondiente es 0.

```
ADC0->EMUX |= 0x0000;
```

Figura 84. Asignación del registro ACDCEMUX en Visual Studio.

El registro ADCSSMUX0 define la configuración de entrada analógica para cada muestra en una secuencia ejecutada para el secuenciador 0, este registro lo encontramos en la página 851.


ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x040
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7				MUX6				MUX5				MUX4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX3				MUX2				MUX1				MUX0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

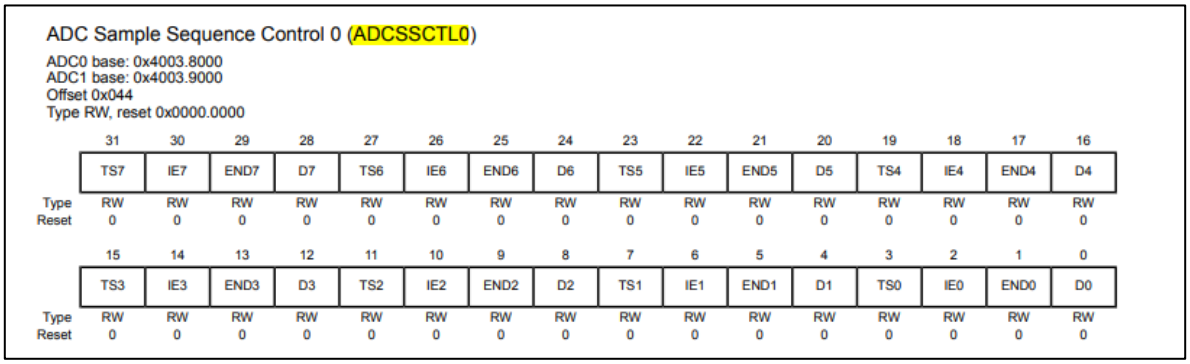
Figura 85. Registro ADCSSMUX0.

La secuencia de muestreo va a ser, primero señal 0, en segundo lugar, la señal 1 y por último la señal 2.

```
ADC0->SSMUX0 |= 0x00000210;
```

Figura 86. Asignación del registro ACDCSSMUX0 en Visual Studio.

Por otro lado, para indicarle cuando comienza y termina el muest utilizamos el registro ADCSSCTL0, que se encuentra en la página 853.


ADC Sample Sequence Control 0 (ADCSSCTL0)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x044
Type RW, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 87. Registro ADCSSCTL0.

Para indicar la interrupción del muestreo es necesario escribir un 1 en el bit 1, el cual indica que la muestra es el final de secuencia, y un 1 en el bit 2, para definir la interrupción de muestreo, lo que nos da en hexadecimal un 6, por lo que la última señal leída es la señal analógica #2.

```
ADC0->SSCTL0 |= 0x00000644;
```

Figura 88. Asignación del registro ACDCSSCTL0 en Visual Studio.

El registro ADCIM muestra el estado de la señal de interrupción sin procesar de cada secuenciador. Este se encuentra en la página 823.

```
ADC0->IM |= (0xf<<16) | (0xf<<0);
```

Figura 89. Asignación del registro ADCIM en Visual Studio.

El registro ADCPC establece el número de conversiones de ADC por segundo , se encuentra en la página 891.

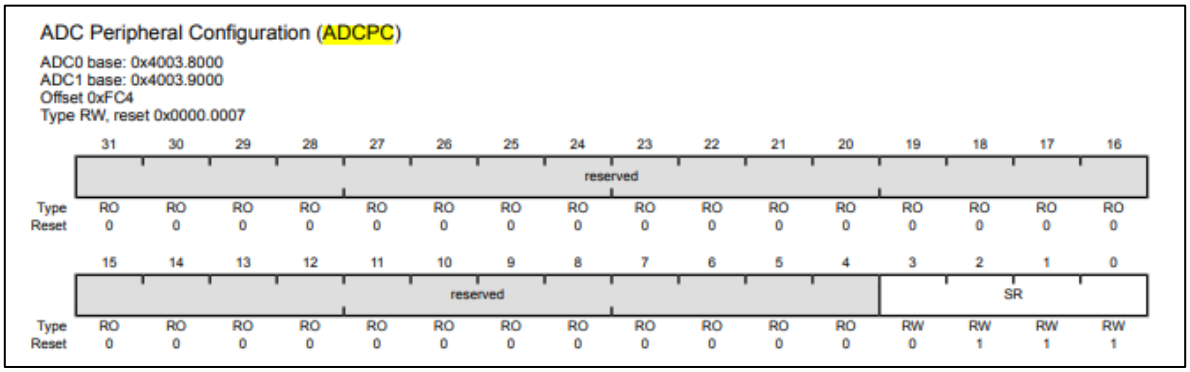


Figura 90. Registro ADCPC.

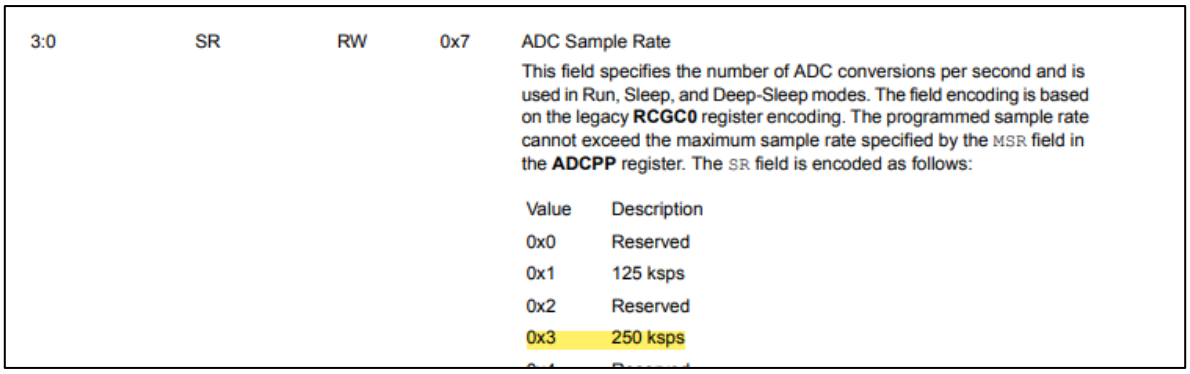


Figura 91. Bits 0-3 del registro ADCPC.

En este caso establecemos una velocidad de conversión de 250ksps

```
ADC0->PC |= 0x3;
```

Figura 92. Asignación del registro ADCPC en Visual Studio.

Finalmente habilitamos los secuenciadores de las muestras mediante el registro ADCACTSS, escribiendo un q en el bit 0 del registro.

```
ADC0->ACTSS |= (1<<0);
```

Figura 93. Asignación del registro ACTSS en Visual Studio.

Ahora establecemos la configuración de lectura de valores del adc como se muestra en la figura 94.

```
extern void ADC0_Leer(uint16_t *adc_data){

    //ADC Processor Sample Sequence Initiate (ADCPSSI)
    ADC0->PSSI |= (1<<0);
    delay_ms(1);

    while (ADC0->RIS & 0x01 == 0);
    delay_ms(1);

    while(ADC0->SSOP0 & (1<<0) == (1<<0));
    adc_data[0] = ADC0->SSFIFO0 & 0xfff;
    delay_ms(1);
    while(ADC0->SSOP0 & (1<<4) == (1<<4));
    adc_data[1] = ADC0->SSFIFO0 & 0xfff;
    delay_ms(1);
    while(ADC0->SSOP0 & (1<<8) == (1<<8));
    adc_data[2] = ADC0->SSFIFO0 & 0xfff;
    delay_ms(1);

    ADC0->ISC |= (1<<0);
    delay_ms(1);
}
```

Figura 94. Configuración para leer ADC0.

Ya configuradas las funciones PWM y ADC, dentro del programa principal, invocamos estas configuraciones y definimos el algoritmo a implementar para la lectura de las señales analógicas obtenidas mediante los potenciómetros y la modificación de la señal PWM para los servomotores.

Los servomotores utilizados en este experimento son los sg90, la datasheet nos indica que su ciclo de trabajo es de 1 a 2ms como se muestra en la figura 95:

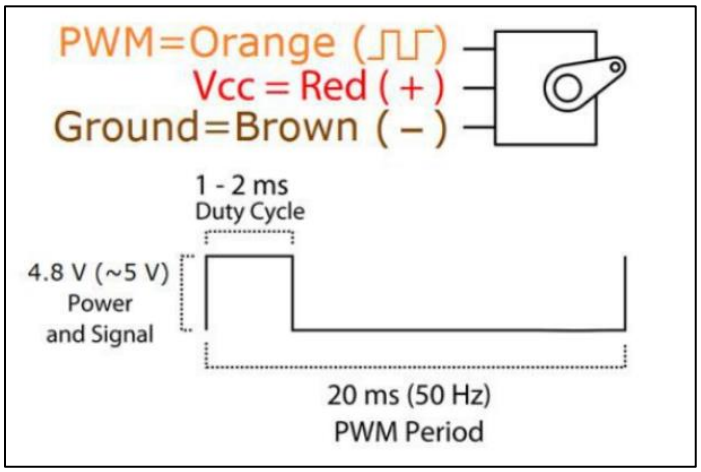


Figura 95. Duty cycle servomotres sg90.

El reloj se establece a 20Mhz y para el PWM se solicita una frecuencia de 50Hz.

Para calcular la frecuencia del PWM hacemos las siguientes operaciones.

Inicialmente, se deben dividir los 20Mhz entre 8, porque así se estableció en la configuración del PWM:

$$\frac{20{,}000{,}000Hz}{8} = 2{,}500{,}000Hz$$

Ahora hacemos la división de los inversos de ambas frecuencias:

$$\frac{\frac{1}{50Hz}}{\frac{1}{2{,}500{,}000Hz}} = 50{,}000Hz$$

El periodo de esta frecuencia es de:

$$\frac{1}{50{,}000Hz} = 20ms$$

Para conocer el valor en frecuencia del ciclo de trabajo del servomotor basta con hacer una regla de 3:

$$\boldsymbol{20ms \rightarrow 50,000Hz}$$
$$\boldsymbol{2ms} \rightarrow 5000Hz$$
$$\boldsymbol{1ms} \rightarrow 2500Hz$$

Ya que conocemos los valores, definimos el algoritmo correspondiente a la lectura de los valores de las señales analógicas provenientes de los potenciómetros de la siguiente manera:

```
delay_ms(50);
ADC0_Leer(adc_data);
dcycle_S1= (uint16_t) (2500+(adc_data[0]*(2500/4095.0)));//Ciclo de trabajo Servo 0
dcycle_S2=(uint16_t)(2500+(adc_data[1]*(2500/4095.0)));// Ciclo de trabajo  Servo 1
dcycle_S3=(uint16_t)(2500+(adc_data[2]*(2500/4095.0))); //Ciclo de trabajo  Servo 2
```

Figura 96. Algoritmo de lectura.

Inicialmente se establece un tiempo de espera de 50ms, para permitir la adquisición correcta de la señal analógica.

Donde adc_data[n] es la variable donde se guardan los valores que envía la señal analógica de los potenciómetros, esta variable se multiplica por el factor $\frac{2500}{4095}$ y al resultado se le suma 2500, ya que el ciclo de trabajo de los servomotores es de 1 ms a 2 ms, para que cuando se obtenga el valor mínimo (0) en la señal analógica, se obtenga el valor de frecuencia correspondiente a 1 ms:

$$2500 + 0 * \left(\frac{2500}{4095}\right) = 2500$$

Y cuando se reciba el valor máximo (4095) de la señal analógica, se obtenga el valor de frecuencia correspondiente a 2ms:

$$2500 + 4095 * \left(\frac{2500}{4095}\right) = 5000$$

El valor obtenido de esta operación se guarda en las variables dcycle_Sn. Para poder modificar el ciclo de trabajo del PWM correctamente se hace la resta del valor total de la carga (50,000Hz) menos el valor obtenido en dcycle_Sn, este valor se estalece en el registro PWMnCMPA como se muestra en la figura*.

```
PWM0->_0_CMPA= PWM0->_0_LOAD- dcycle_S1;
PWM0->_1_CMPA= PWM0->_1_LOAD- dcycle_S2;
PWM0->_2_CMPA= PWM0->_2_LOAD- dcycle_S3;
```

Figura 97. Modificación del porcentaje del ciclo de trabajo.

RESULTADOS.

Resultados-Experimento 1

El duty establecido en los registros descritos en la metodología, corresponde a un 20% y una frecuencia de 10KHz, valores que se muestran en el osciloscopio en la figura 36.

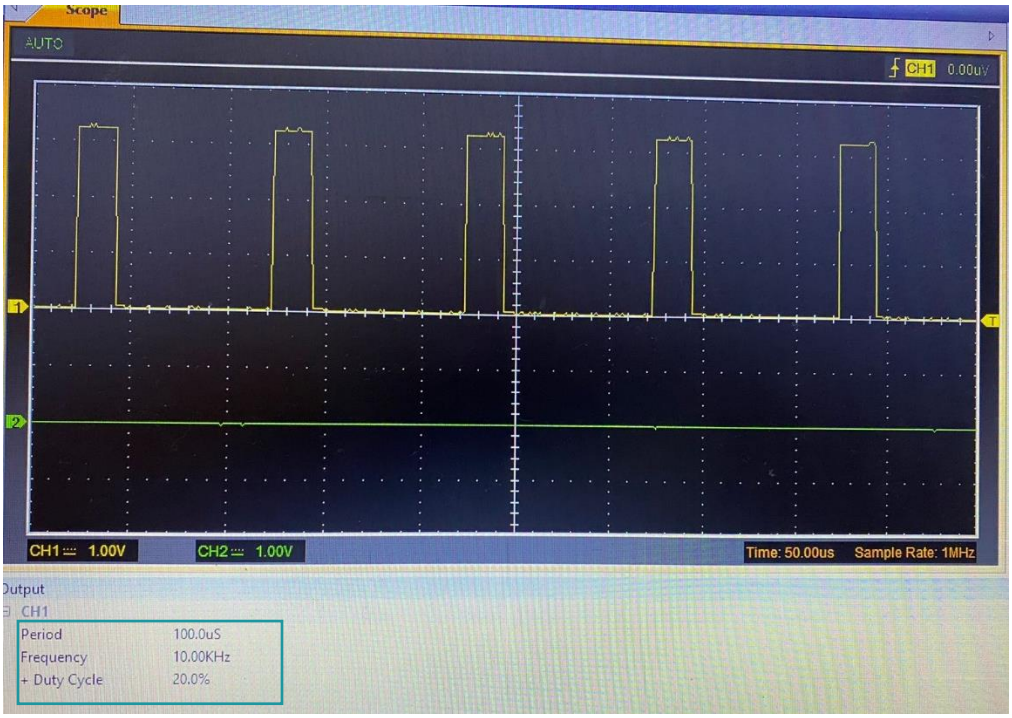


Figura 98. Señal en Osciloscopio

Resultados-Experimento 2

El duty cycle correspondiente a 1 ms es del 10% del periodo de 20ms de la señal, este valor, así como la frecuencia solicitada de 50Hz se obtuvieron de manera correcta, como se muestran en el osciloscopio en la figura **.

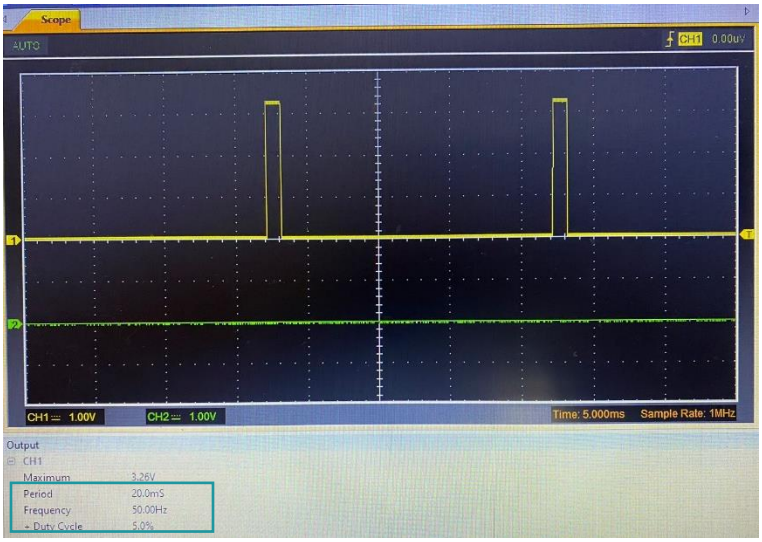


Figura 99. Señal en Osciloscopio

El duty cycle correspondiente a 2 ms es del 20% del periodo de 20ms de la señal, este valor, así como la frecuencia solicitada de 50Hz se obtuvieron de manera correcta, como se muestran en el osciloscopio en la figura **.

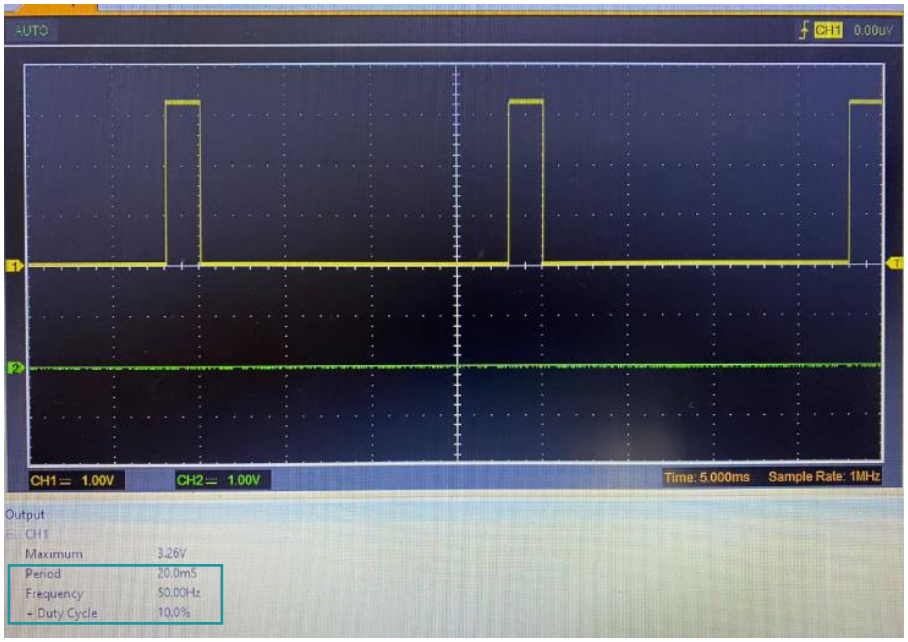


Figura 100. Señal en Osciloscopio

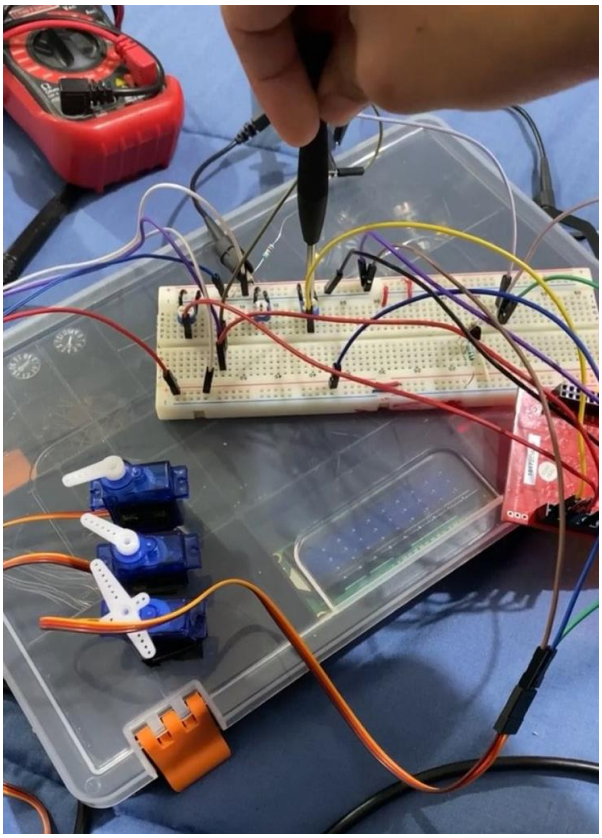


Figura 101. Conexión de los servomotores.