Instituto Tecnológico Superior De Atlixco.

Ingeniería Mecatrónica.

IM140744.

7-A.

Microcontroladores.

PRÁCTICA 2 Uso del TIMER, ADC y DAC.

CATEDRÁTICO: Dra. Mariana Natalia Ibarra Bonilla.

INTEGRANTE: José Ángel Balbuena Palma. IM140744

Fecha de realización:

30/10/2017

Fecha de entrega:

1/11/2017

Objetivo general.

El objetivo de estas prácticas es el uso de módulos como el temporalizador, el convertidor analógico-digital (ADC) y el convertidor digital-analógico (DAC) que trae incorporado nuestra tarjeta de desarrollo Ophyra los cuales utilizaremos para diversas aplicaciones prácticas.

Materiales.

- Protoboard.
- Módulo de LEDS.
- Tarjeta de desarrollo Ophyra.
- Servomotor.
- Jumpers.
- Osciloscopio.
- Computadora con el software Atollic TrueStudio, STM32CubeMX y STMFlashLoader .

MARCO TEÓRICO.

TEMPORALIZADOR.

Un temporalizador permite medir el tiempo transcurrido entre eventos basados en contar pulsos generados por el oscilador principal. La tarjeta de desarrollo Ophyra trae integrado 12 timers de 16 bits y 2 de 32 bits para el contador de autorecarga, el cual puede ser creciente o decreciente. Para las aplicaciones a desarrollar un timer con la resolución de 16 bits el cual puede presentar un número de hasta 65535 para su conteo y con una frecuencia de reloj de 16Mhz.

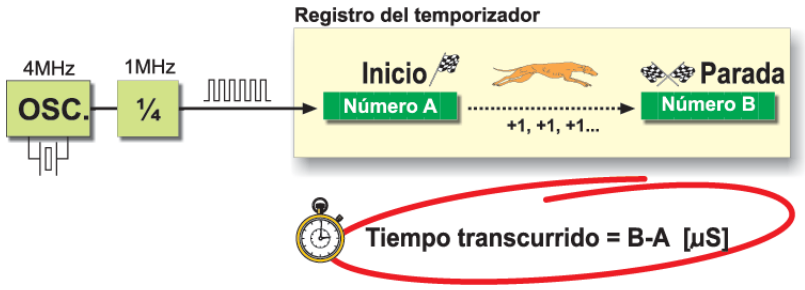


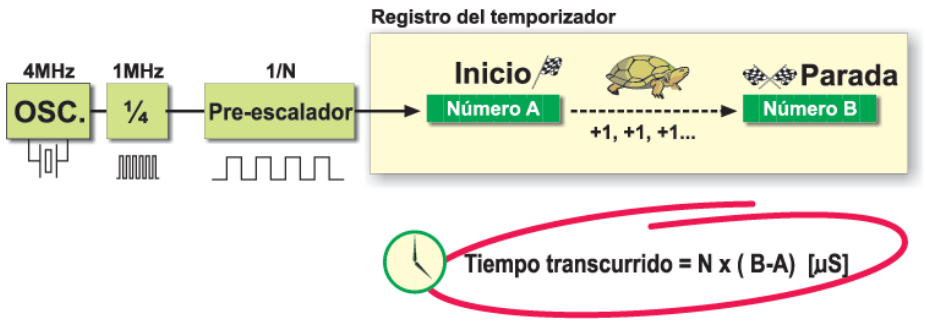
Diagrama de un Temporalizador

Prescaler.

Un prescaler es un divisor de frecuencia programable que permite cambiar la frecuencia nuestra señal de reloj a una deseable. Se tiene una frecuencia de reloj grande y al aplicar el prescaler se tiene una frecuencia de reloj menor.

Calculo del prescaler:

$$Prescaler = \frac{Frecuencia\ del\ Reloj}{Frecuencia\ deseada} - 1$$



Temporalizador con prescaler

Base de Tiempos.

Es un circuito que esta formado por tres bloques con un registro asociado:

- El contador: Es un contador ascendente o descendente de 16 bits. El cual tiene un Counter Register El contador se incrementa o decrementa una unidad cuando llega un flanco del siguiente bloque.
- El prescaler. Este también cuanta con un registro el Prescaler Register.
- El registro de autocarga (Auto-Reload Registrer). Es un registro que almacena el periodo que va a contar el Timer. Es conocido como el contador de periodos(Counter Period). Hay dos tipos ascendente y descendente:
  - Contador descendente: El valor del contador disminuye hasta llegar a 0 en el siguiente conteo escribe el contenido en el registro y empieza nuevamente.
  - Contador ascendente:El valor del contador crece hasta llegar alcanzar al valor en el registro, en el siguiente conteo se pone a cero y comienza de nuevo.

Calculo del contador de periodos.

$$Contador\ de\ periodos = (Frecuencia\ deseada)(Tiempo\ a\ contar) - 1$$

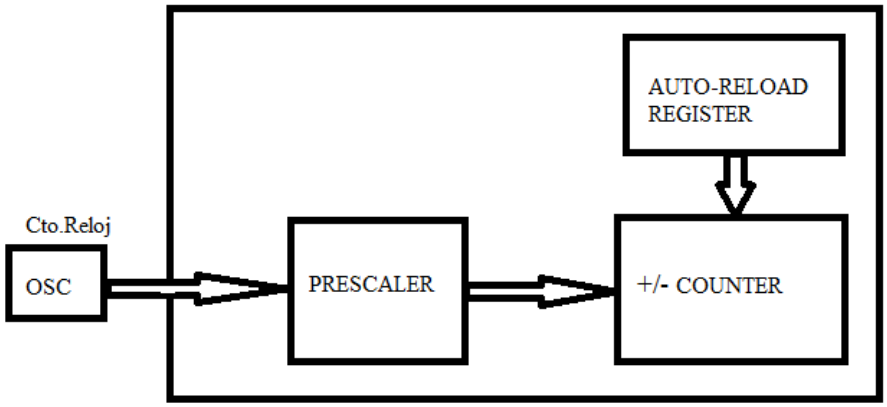


Diagrama de bloques del timer para generar la base del tiempo.

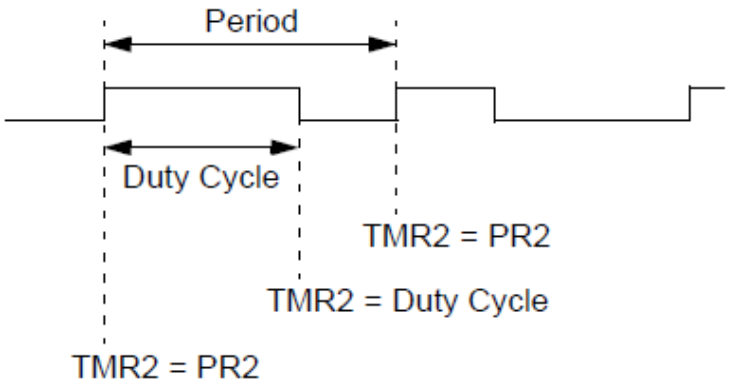
Interrupción.

Una interrupción es un evento que hace que el microcontrolador deje de ejecutar la tarea que está realizando para atender dicho acontecimiento, luego regrese y continúe la tarea que estaba llevando acabo antes de que se presentara la interrupción. Una interrupción por timer ocurre cuando se produzca un evento de actualización. En el cado de un contador ascendente esta ocurrirá cuando exista un desborde en el conteo, pues el contador habrá llegado a su valor máximo de conteo en el registro y este se reiniciará de nuevo produciendo una actualización.

Generación del PWM usando Timers.

¿Qué es PWM?

Es la modulación por ancho de pulso, que es la variación del ciclo de trabajo este es la duración de la señal en alto, esta no varía la frecuencia de la señal.



Modulación por ancho de pulso.

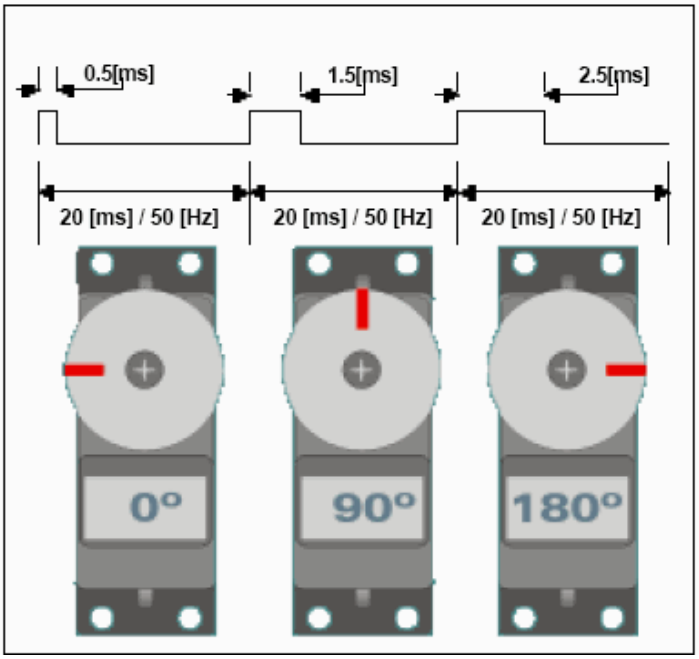
Para usar el PWM por timer, debemos definir un prescaler para reducir la frecuencia del timer de 16MHz a la frecuencia del pwm que emplearemos.

$$Prescaler\ = [\frac{Frecuencia\ del\ Reloj}{(Frecuencia\ del\ PWM)(Resolucion\ del\ PWM)}] - 1$$

La resolución del Pwm nos da el numero de incrementos o intervalos en el que se divide el periodo de la señal que su totalidad nos darían el 100% del ciclo de trabajo.

Control de servomotores atreves de PWM.

Para emplear el centro de servomotores atreves del pwm se deben considerar aspectos de operación en del ciclo de trabajo , así como el periodo la señal el cual es de 20ms, para el correcto funcionamiento y posicionamiento de los servomotores.

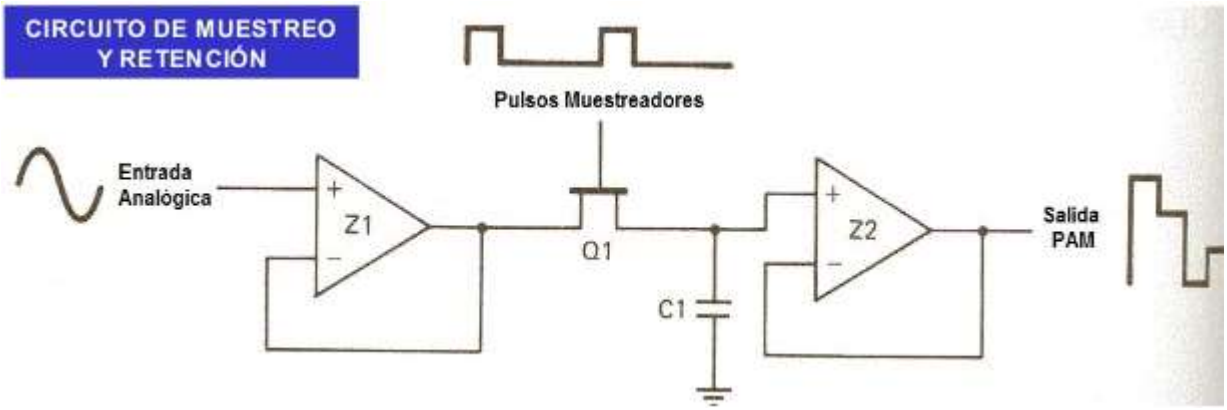


Posiciones de servomotores a diferentes ciclos de trabajo.

Como se muestra en la anterior imagen para un ciclo de trabajo del 2.5% corresponde una posición de 0 grados, para el 7.5% corresponde la posición de 90 grados y para el 12.5% tendremos la posición de 180 grados.

ACD (Convertidor Analógico -Digital).

Es el que se encarga de convertir un valor analógico de voltaje en su correspondiente combinación en binario. El módulo ADC utiliza muestreo y retención (sample and hold) con un condensador un capacitor.



Muestreo y Retención

El convertidor A/D convierte una señal de entrada en un resultado de 12 bits atreves de aproximaciones sucesivas.

Las aproximaciones sucesivas se basan en realizar una contabilización de forma ascendente o descendente hasta encontrar un valor digital que iguale a la tensión entregada por el conversor D/A (Digital analógico) y la tensión de la entrada.

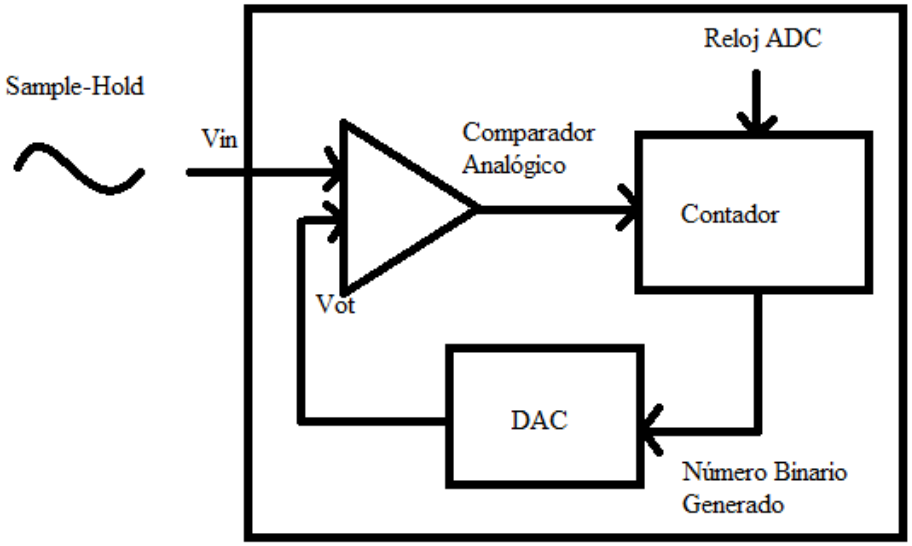
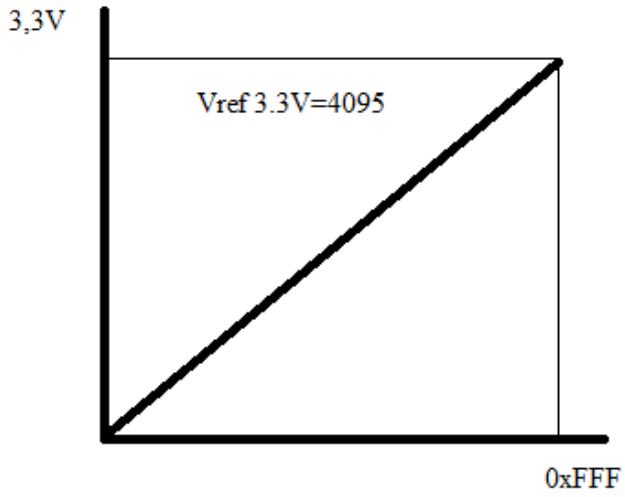


Diagrama de aproximaciones sucesivas

Resolución mínima o cantidad de conversión se puede ajustar a diferentes necesidades al seleccionar voltajes de referencia. El ADC que posee nuestro microcontrolador es de 12 bits por lo que el valor máximo a presentar en el convertidor es de 4095, el voltaje máximo que se puede introducir al pin del ADC es de 3.3V .



Por lo que la resolución de ADC es de 0.805mV.

DMA.

El acceso de memoria directo es un sistema de transferencia de datos especial, el cual permite la transferencia de datos en segundo plano sin el uso de CPU , el procesador puede ejecutar otras tareas y solo interrumpirá cuando esta disponible un bloque datos completos para su procesamiento.

DMA que con él cuenta nuestro microcontrolador posee hasta 8 canales para la transferencia de datos. Se puede configurar la transferencia de datos en un flujo continuo, a través de la configuración del DMA modo circular.

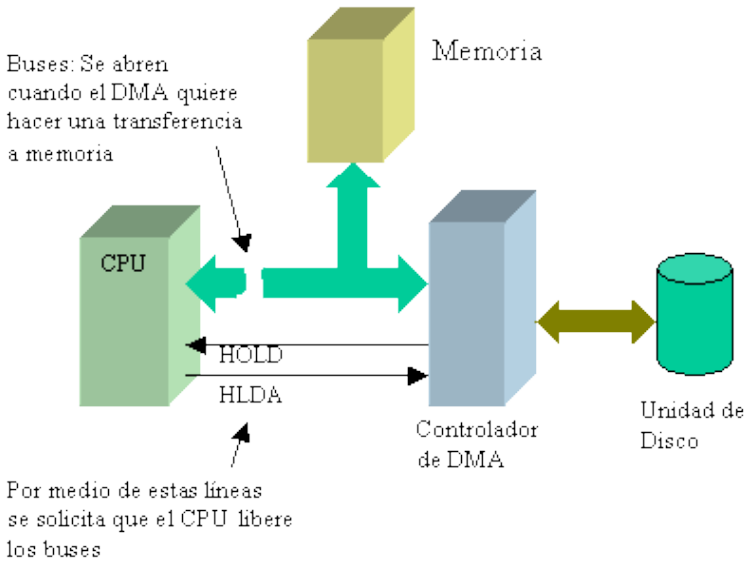


Diagrama de bloques de un DMA.

DAC.

Es un convertidor digital analógico, puede convertir señales digitales con datos binarios en señales voltaje analógico. El microcontrolador STM32F407VGT6 cuenta con dos DAC que se encuentran en los timers 6 y 7 y la resolución de estos DAC es de 12 bits, el voltaje máximo que pueden proporcionar es de 3.3V.

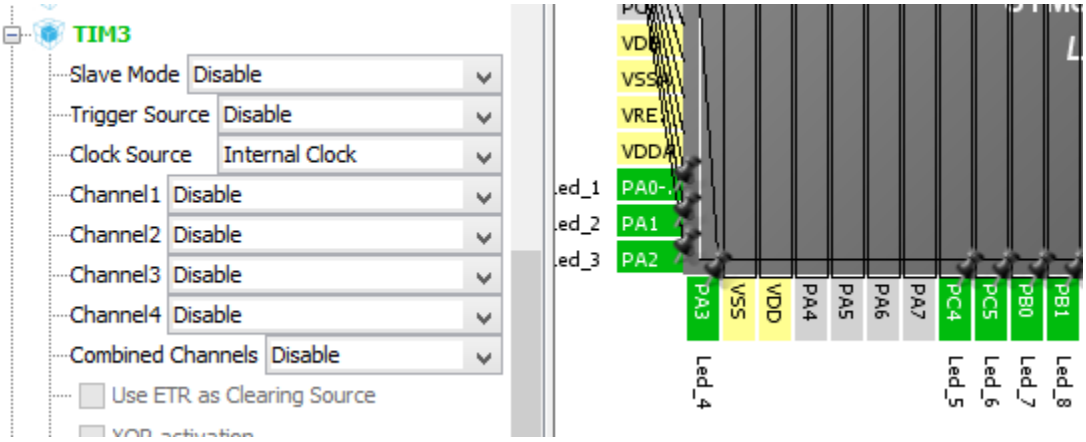
Desarrollo.

Practica1. Secuencia Kit con el uso de la interrupción por Timer.

La actividad que se desarrollo fue una secuencia para el control del encendido y apagado de un módulo de 8 leds, la cual consiste en el encender el primer led pasado un segundo apagar el led encendido para encender el led 2, después de otro segundo se apaga el segundo led y se enciende el tercer led repitiendo el patrón para los ocho leds, una vez llegado al led numero 8 la secuencia se realiza en orden descendente el led ocho se apaga para encender el led 7 , así hasta llegar al led 1. Para lograr el conteo de un segundo se recurrió a una interrupción por timer.

Configuración del timer e interrupción en el software ST32cubeMx.

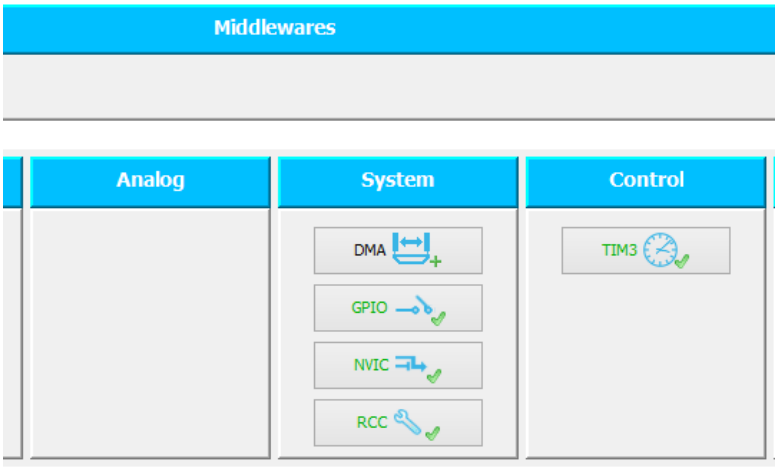
En el software creamos un nuevo proyecto, seleccionamos el micro controlador stm32f407vg, para después configurar los recursos a emplear.



Selección de pines y timer a emplear con fuente de reloj interna.

Como se puede observar en la imagen anterior se seleccionan 8 pines de micro controlador como salidas, también seleccionamos el timer 3 el cual es un timer con una señal de reloj de 16 MHz la cual emplearemos fuente de reloj este temporizador es de 16 bits.

Se debe pasar a la configuración del timer y pines GPIO.



Ventana para la configuración del timer y pines GPIO.

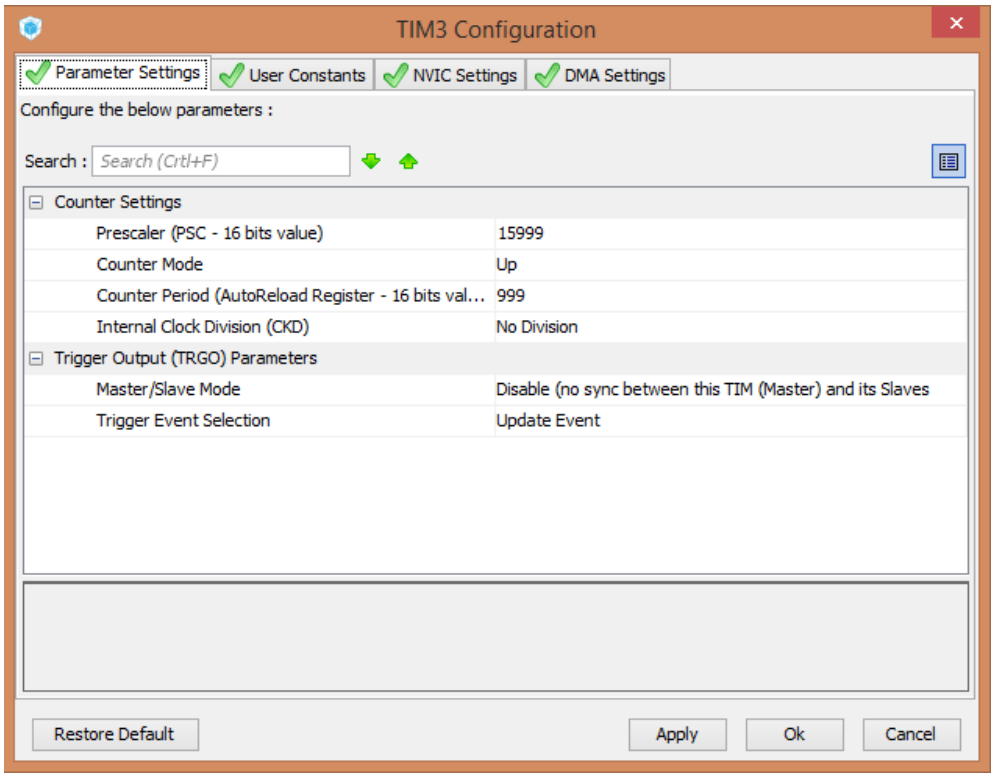
Configuración del timer.

Par poder realizar la confirmación del timer lo primero que se debe realizar es el cálculo del prescaler para reducir la frecuencia de reloj a una frecuencia deseada como de 1KHz.

$$Prescaler\ = \frac{16MHz}{1KHz} - 1 = 15999$$

También se debe calcular la el contador de periodos que emplearemos para un tiempo de 1segundo con la frecuencia configurada a un 1KHz.

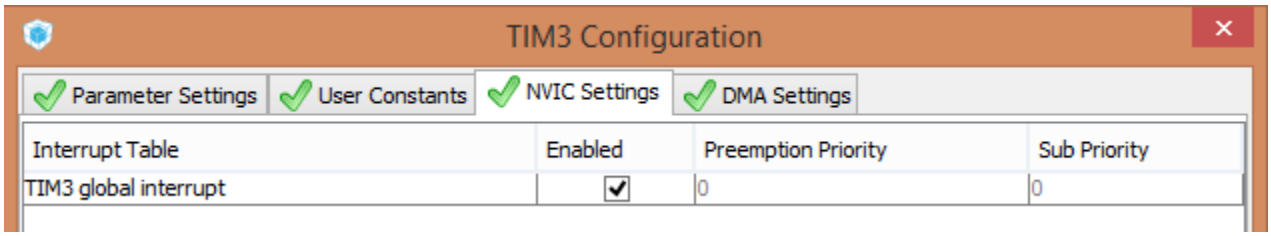
$$Contador\ de\ periodos = (1KHz)(1s) - 1 = 999$$



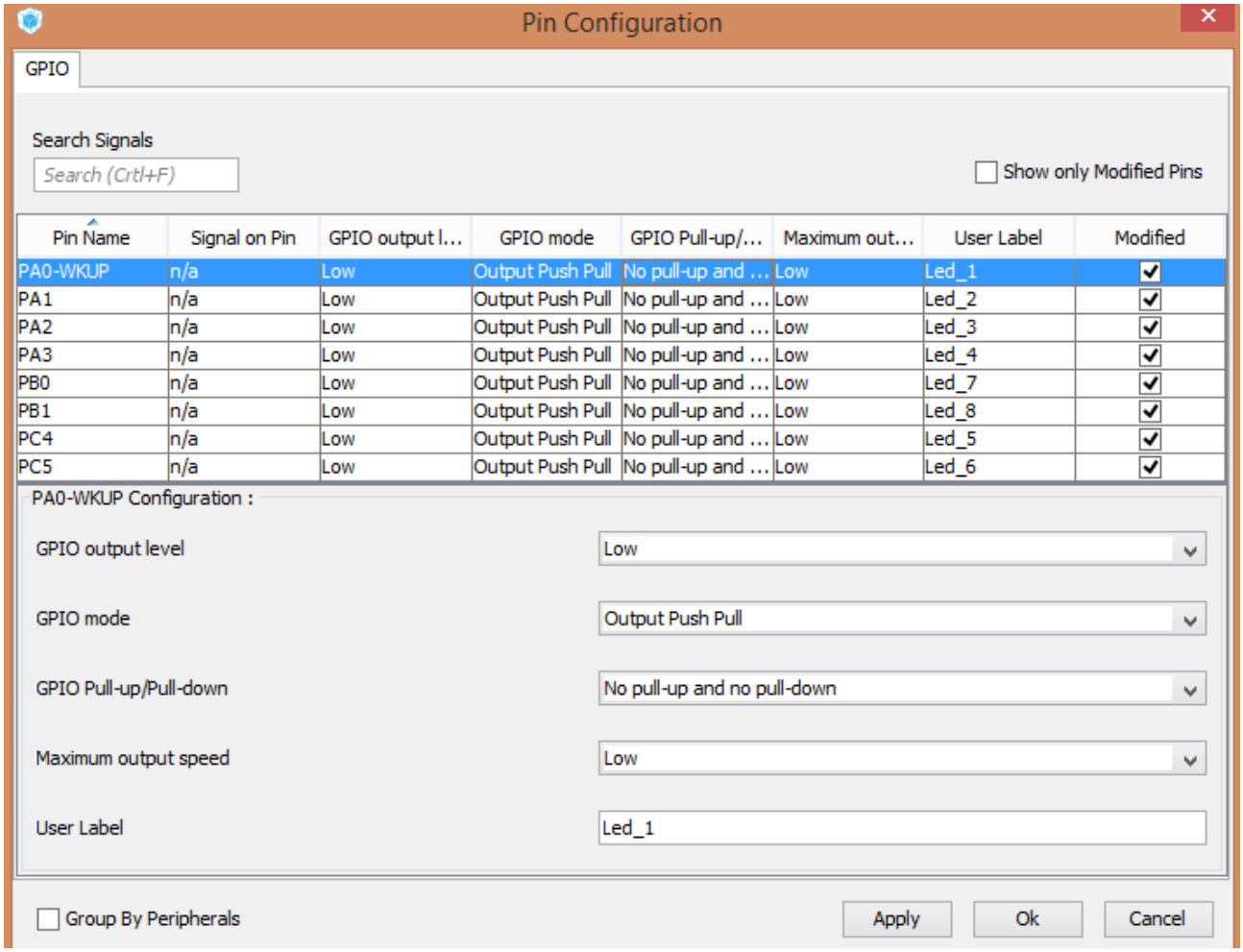
Parámetros del temporalizador.

En la pestaña Parameter Settings configuramos el prescaler a 15999 el valor obtenido, En Counter Mode seleccionamos Up esto se realiza para configurar el contador de manera ascendente. También se debe colocar el Counter Period que calculamos de 999.

Debemos configurar la interrupción por timer y esta se hace en la pestaña NVIC seleccionando la opción TIM3 global interrupt.



Se deben configurar los pines de salida en modo Push Pull , asignar un nivel lógico bajo a los pines para que los pines comiencen apagados.



Configuración de los pines GPIO para los leds a emplear.

Por ultimo de debe generar el código que emplearemos en software Atollic true Studio para la programación de la aplicación.

En Atollic true Studio buscamos el archivo main.c dentro de la carpeta Src del proyecto, se debe habilitar la interrupción en la función principal int main(void){},dentro de esta funcion se encuentra la instrucción MX_TIM3_Init()que inicializar el timer3, para inicializar el modo de la interrupción se emplea la función HAL_TIM_Base_Start_IT(&htim3); la cual recibe como parámetro el puntero de timer3.Esta instrucción se puede colocar afuera del ciclo infinito ya que el conteo se reinicia cada que hay una actualización y comienza de nuevo el conteo.

Para programar el código que se ejecutara en la interrupción es necesario situarnos en el archivo stm32xx_it.c donde se contiene las definiciones de las funciones de las interrupciones que son invocadas al requerirse en el archivo main.c, buscamos la función void TIM3_IRQHandler(void){} es la funciones que es llamada cada vez que termina el conteo del timer, en esta función encontramos la instrucción HAL_TIM_IRQHandler(&htim3); la cual vuelve a iniciar el conteo automáticamente.

Función de la interrupción del timer3 para implementar este código debemos declara una variable int=contador; donde me registre el caso en el que nos encontramos.

```
void TIM3_IRQHandler(void)
{
  HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_RESET);
  HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_RESET);
   switch(contador){
     case 1:
         HAL_GPIO_WritePin(GPIOA, Led_1_Pin,GPIO_PIN_SET);
         break;
     case 2:
         HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_SET);
         break;
     case 3:
         HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_SET);
         break;
     case 4:
         HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_SET);
         break;
     case 5:
         HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_SET);
         break;
     case 6:
         HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_SET);
         break;
     case 7:
         HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_SET);
         break;
     case 8:
         HAL_GPIO_WritePin(GPIOB, Led_8_Pin,GPIO_PIN_SET);
         break;
     case 9:
        HAL_GPIO_WritePin(GPIOB, Led_7_Pin,GPIO_PIN_SET);
         break;
     case 10:
         HAL_GPIO_WritePin(GPIOC, Led_6_Pin,GPIO_PIN_SET);
         break;
     case 11:
         HAL_GPIO_WritePin(GPIOC, Led_5_Pin,GPIO_PIN_SET);
         break;
     case 12:
         HAL_GPIO_WritePin(GPIOA, Led_4_Pin,GPIO_PIN_SET);
         break;
     case 13:
         HAL_GPIO_WritePin(GPIOA, Led_3_Pin,GPIO_PIN_SET);
         break;
     case 14:
     HAL_GPIO_WritePin(GPIOA, Led_2_Pin,GPIO_PIN_SET);
         contador=0;
         break;
  }
  HAL_TIM_IRQHandler(&htim3);
  contador++;
}
```

Apagamos los led que se encuentran encendidos

Para el la variable contador que hemos generado creamos una estructura switch la cual contiene 14 casos en los cuales enciende un led en la posición que se encuentre el conteo para el contador igual a 1 se enciende el led 1 y así hasta el contador igual a 9 en esta posición el led encendido es el 7 ya que la secuencia de los leds iría de regreso hasta el led 1.

En el caso 14 la variable contador es reiniciada para que en la siguiente actualización comience el conteo en uno.

Es una función que reinicia el conteo del timer

La variable contador incrementa en uno cada vez que existe un desborde y el contador del timer es reiniciado.

Practica 2. Control de un servomotor atreves de un potenciómetro con el uso de ADC y el PWM del timer3.

Como hemos visto para poder configurar el timer para el control de un servomotor atravez de PWM requiriendo dos parámetros el prescaler y el contador de periodos que es la resolución del PWM.

Calculo de prescaler para el PWM.

Se debe hacer a una frecuencia de 50Hz.Pero la resolución se calcula de la siguiente manera:

Rango de operación del servo = 2.5ms-0.5ms=2ms

Dividimos el rango de operación del servo entre las 180 posiciones angulares que pude ofrecernos.

2ms/180=1.11*10^-5segundos

Tenemos 20ms por los dividimos entre 1.11*10^-5segundos

20ms/1.11*10^-5s=1800

Por lo que tenemos una resolución de 1800 para el pwm para representar el ciclo trabajo a 100%.

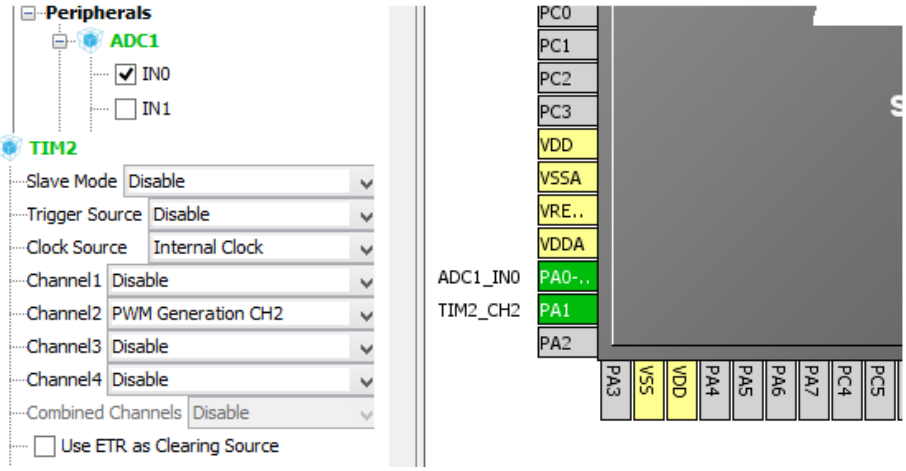
Con los datos que obtuvimos es posible calcular el prescaler.

$$Prescaler = \left[\frac{16MHz}{(50Hz)(1800)}\right] - 1 = 177$$

Configuración de StmCubeMx.

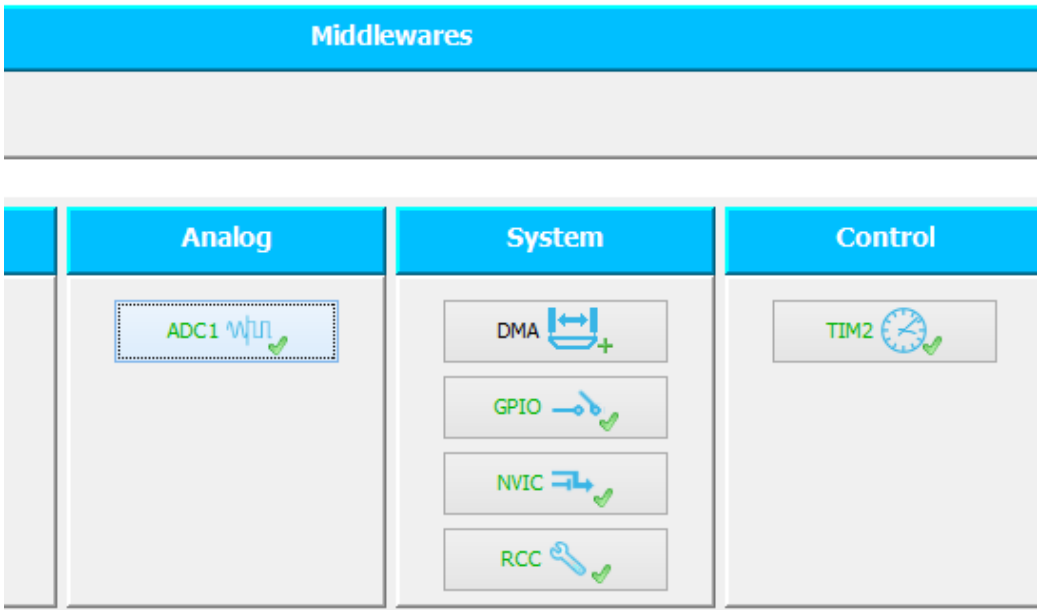
En el software creamos un nuevo proyecto, seleccionamos el micro controlador stm32f407vg, para después configurar los recursos a emplear.

Primero seleccionamos el ADC a utilizar el cual es el ADC1, seleccionamos el canal1 del ADC que se encuentra en el pin PA0 , seleccionamos el Timer que se va utilizar el cual es el Timer2 , seleccionamos el reloj interno como señal de reloj y el canal 2 la opción PWM Generation CH2 que se encuentra en el pin PA1. Como se muestra en la imagen a continuación.



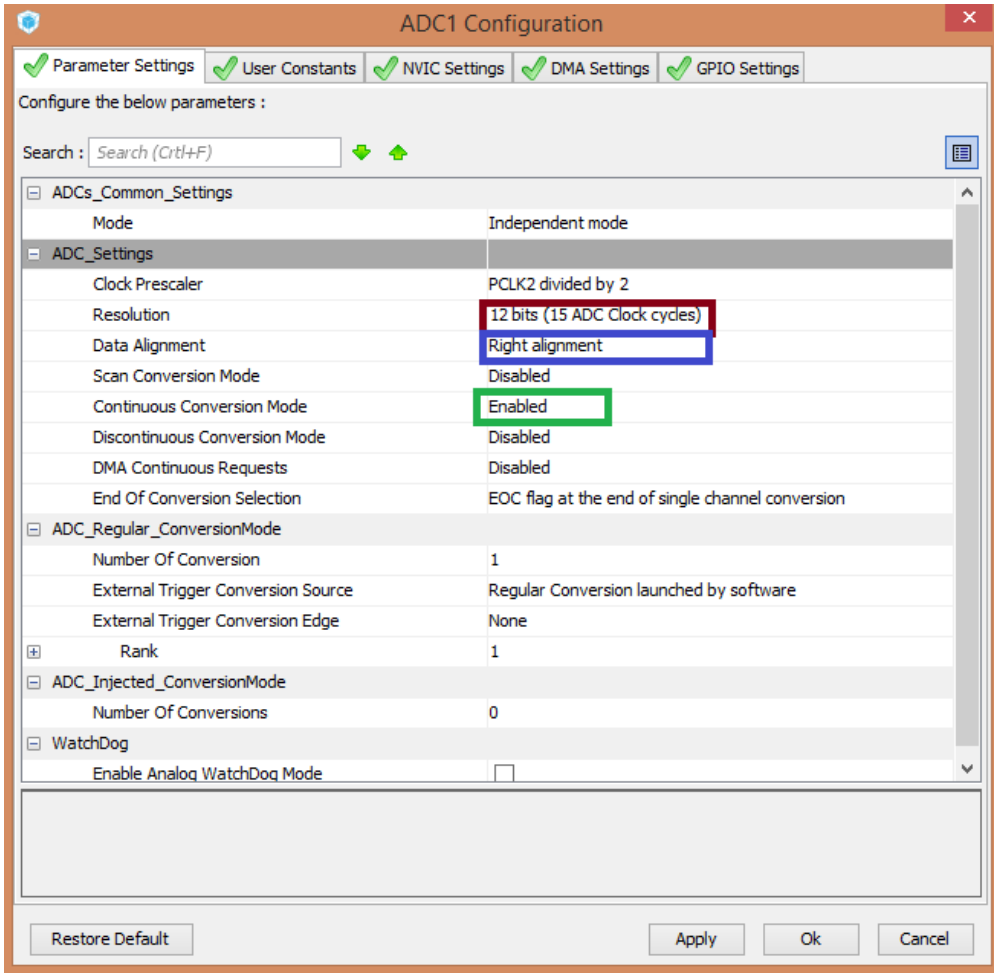
Selección del Timer y ADC a utilizar.

Configuración del ADC Y TIMER2.



Seleccionamos la opción de ADC1.

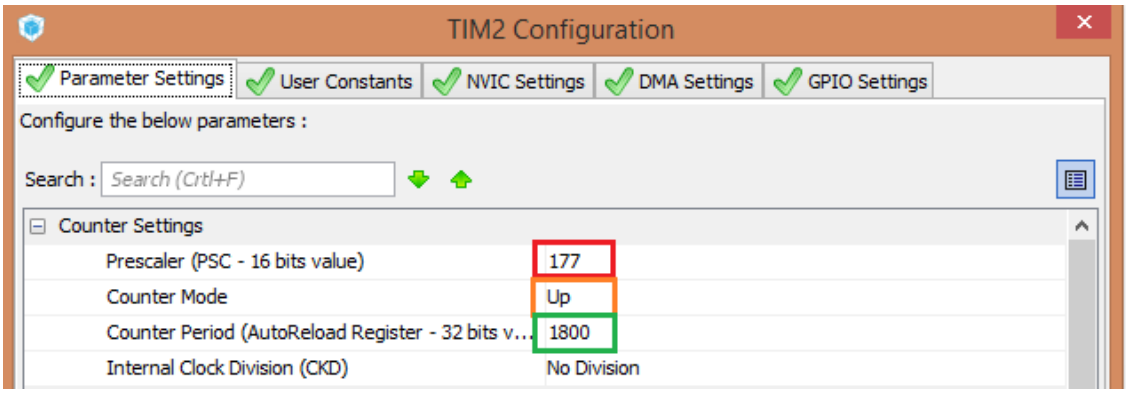
Donde configuraremos la resolución de ADC a 12 bits, la alineación de los datos a la derecha ubicando el bit menos significativo en la parte derecha, también activamos el modo de conversión continuo.



Configuración del ADC.

Seleccionamos el Timer2.

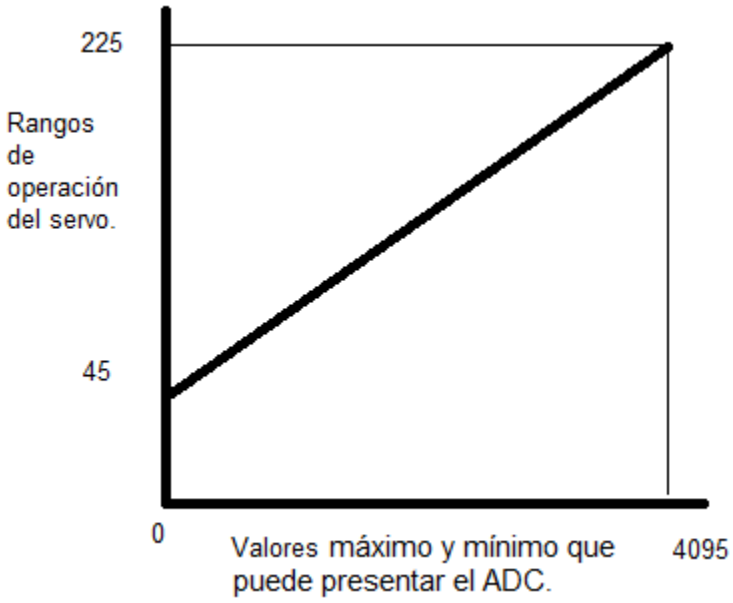
En la opción de establecer parámetros se debe colocar el prescaler que se ha calculado de 177 así como el contador de periodos que obtuvimos hay que recordar que este valores el mismo de la resolución de nuestro PWM de 1800, se debe activar el modo de el conteo ascendente. Como se muestra a continuación.



Configuración del Timer2.

Generamos el archivo para programar en Atollic True Studio.

Para poder relacionar los valores que proporcionados por el potenciómetro a través del ADC de 12 bits que van de 0 a 4095 para lecturas de entrada de 0 a 3.3v, con los rangos de operación del PWM para un servomotor de 2.5% del ciclo trabajo al 12.5% que corresponden a los valores numéricos de 45 a 225 para una resolución de 1800 se recurre a una relación como la que se muestra a continuación.



De esta recta podemos obtener una ecuación que nos relacione el valor del ciclo de trabajo a emplear en pwm con el valor obtenido por el ADC dado por el potenciómetro.

Obtención de los valores para la ecuación y=mx+b donde m es la pendiente de la recta.

$$45=m\,(0)+b \rightarrow b=45$$

$$225=m4095+45 \rightarrow m=(225-45)/4095$$

$$m=4/91$$

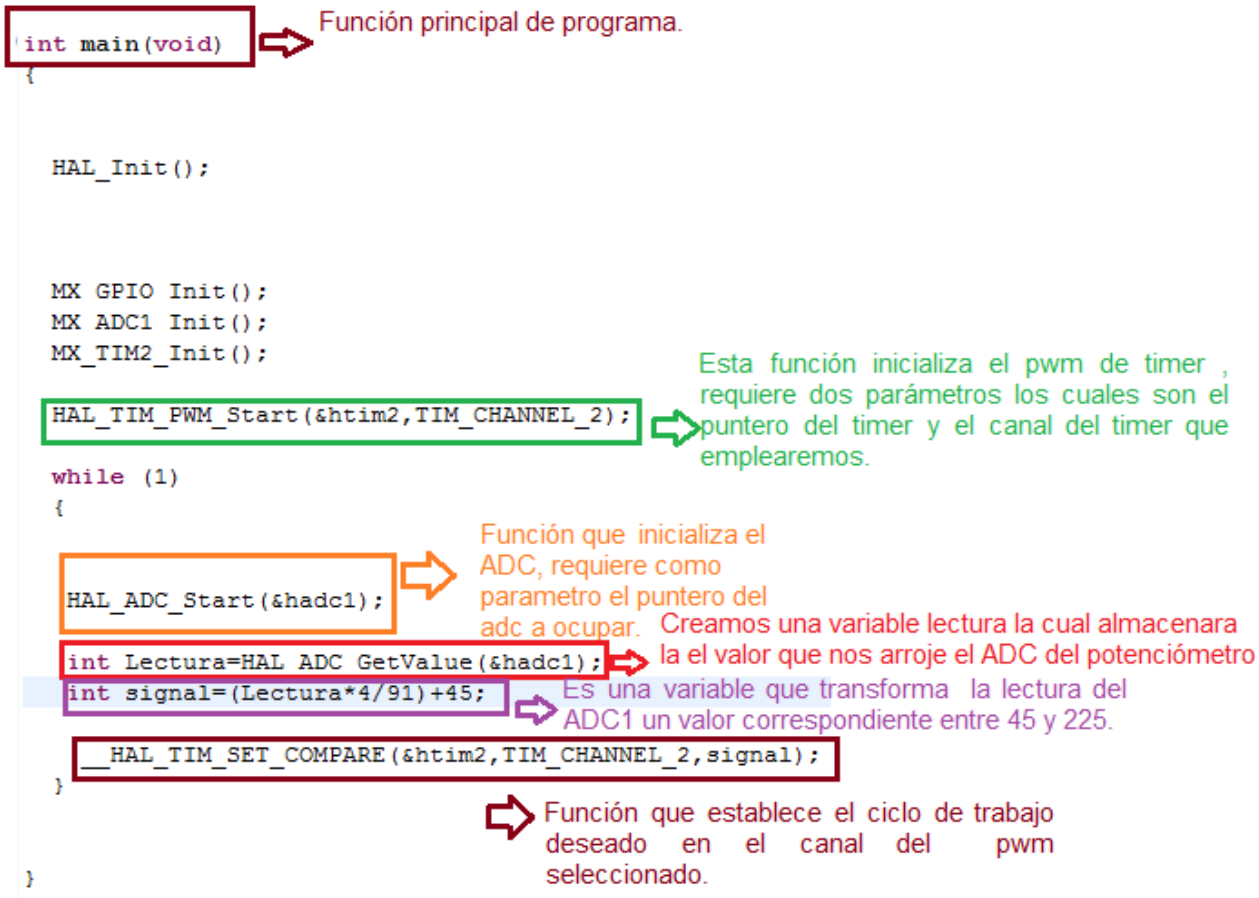
$$y=(4/91)*x+45$$

ciclo de trabajo a emplear en pwm=(4/91)*(valor obtenido por el ADC del potenciómetro)+45

Nos dirigimos a la carpeta del proyecto buscamos la capeta Src la cual contiene el archivo main.c el cual se muestra a continuación.

```
39 #include "main.h"
40 #include "stm32f4xx_hal.h"
41 #include "adc.h"
42 #include "tim.h"
43 #include "gpio.h"
44
```

Librerías incluidas podemos observar que se incluyen las librerías del adc y del timer

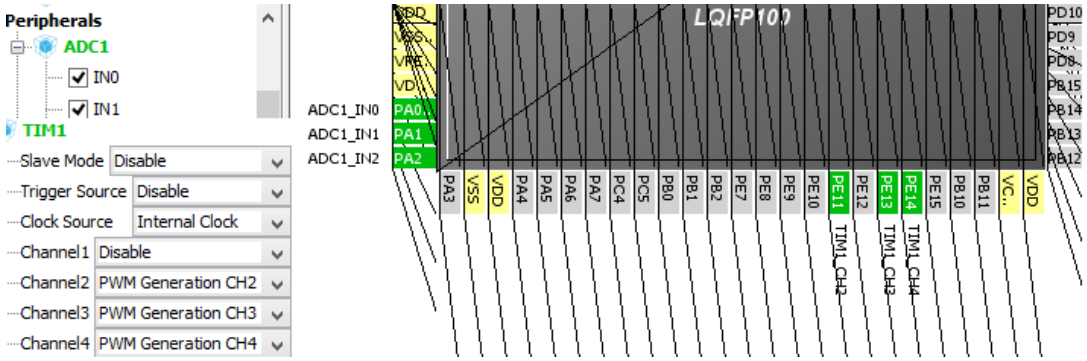


Practica3. Control de 3 servomotores a través de tres potenciómetros, utilizando el DMA para la transferencia de los datos obtenidos en el ADC.

Se debe emplear el uso del DMA para la transferencia de datos en la lecturas de los valores obtenido de ADC por cada uno de los potenciómetros a utilizar para agilizar este proceso se hace uso de esta poderosa herramienta, y obtener los valores de una manera más fluida.

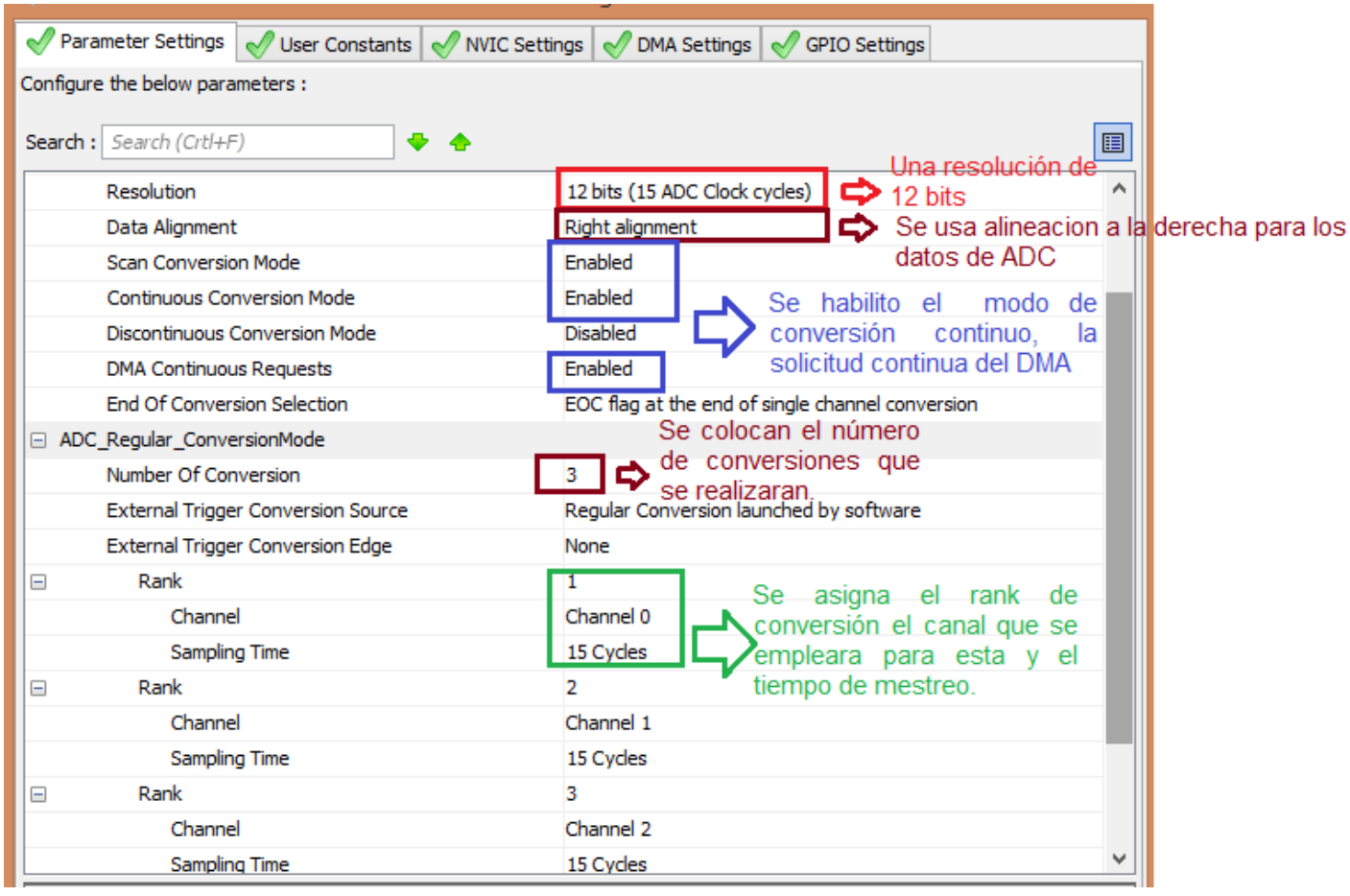
Configuración de StmCubeMx.

En el software creamos un nuevo proyecto, seleccionamos el micro controlador stm32f407vg, para después configurar los recursos a emplear.

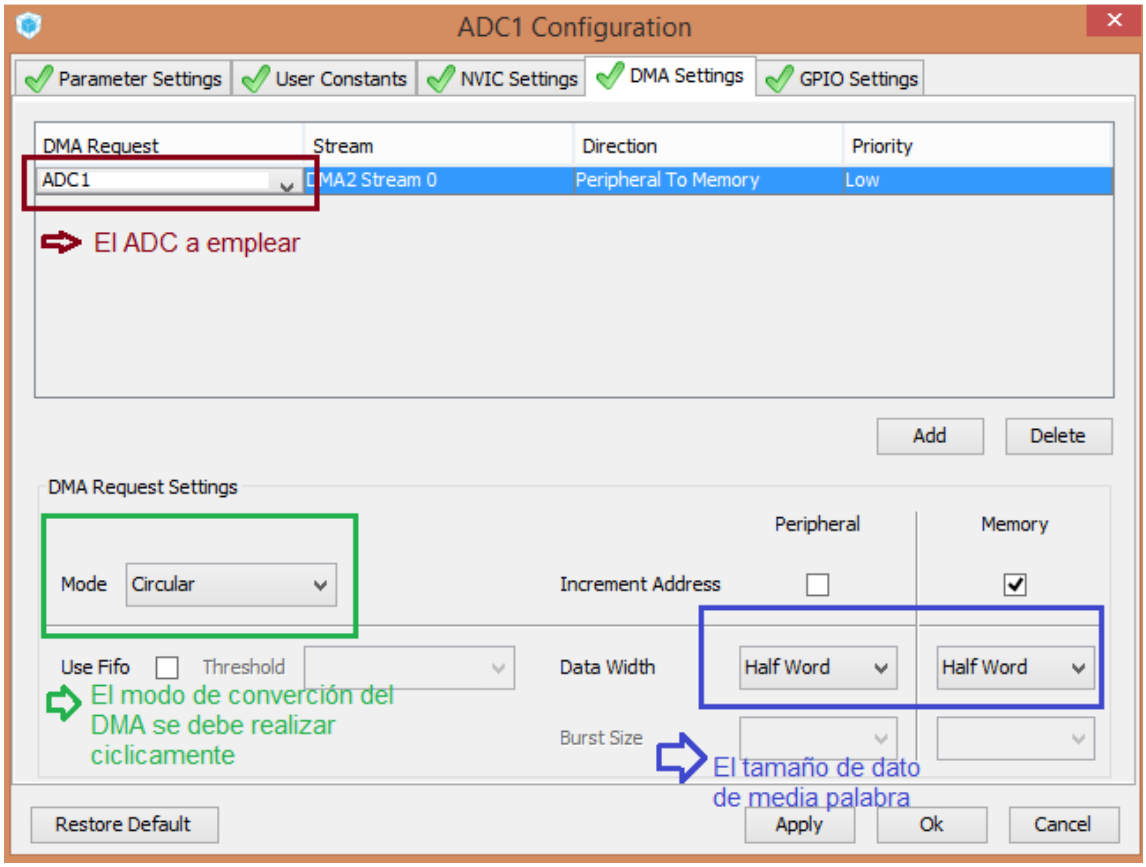


Como se muestra en la anterior imagen se seleccionó el ADC 1 con tres canales y el Timer 1 con tres canales para la generación del PWM el canal2, canal 3 y el canal 4.

Para el ADC en parameter settings se configuro.

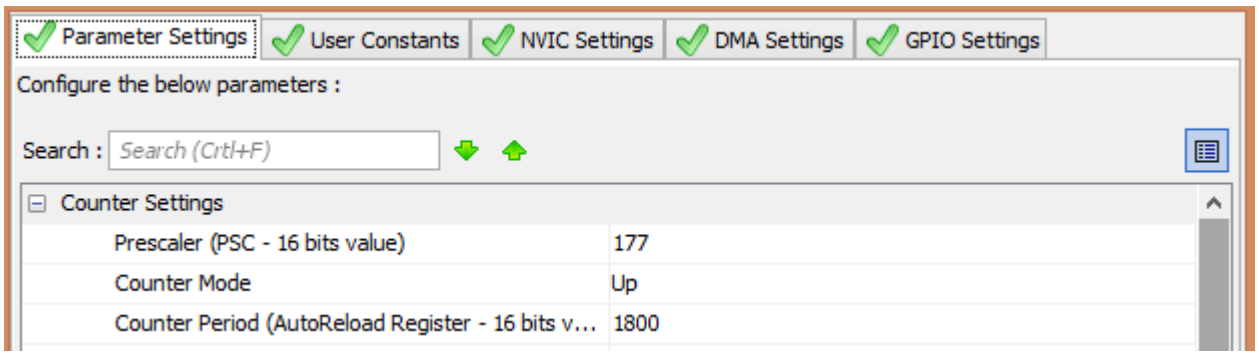


En DMA settings se configura.



Configuración del timer1 del PWM.

Se asigna el prescaler y contador de periodos calculados en la práctica anterior.



Se genera el código para la Programación en Atollic True Studio.

En el archivo main.c se declara una variable global uint16_t lectura_adc[3]; la cual es un buffer de 16 bits donde se almacenaran las tres lecturas de los canales del ADC.

Dentro de la función principal.

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_TIM1_Init();

  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim1,TIM_CHANNEL_4);
  HAL_ADC_Start_DMA(&hadc1,(uint32_t*)lectura_adc,3);

  while (1)
  {

    HAL_ADC_Start(&hadc1);
    int signal1=(lectura_adc[0]*4/91)+45;
    int signal2=(lectura_adc[1]*4/91)+45;
    int signal3=(lectura_adc[2]*4/91)+45;
      __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_2,signal1);
     __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3,signal2);
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_4,signal3);

  }
```

Se inicializan los tres canales de los pwm del timer1.

Esta función inicializa el DMA, requiriendo como parámetros el puntero del ADC a usar, y el buffer donde se almacena las lecturas pero modificando su tamaño a 32 bits.

Inicialización del ADC.

Creación de variables para la conversión de los valores obtenidos por el DMA a valores entre 45 y 225.

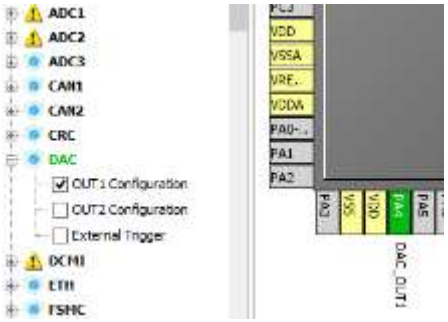
Establecemos el ciclo de trabajo en los PWM del timer dados por el DMA para el control de los servos.

Practica 4. Uso del DAC para creación de una señal diente de sierra.

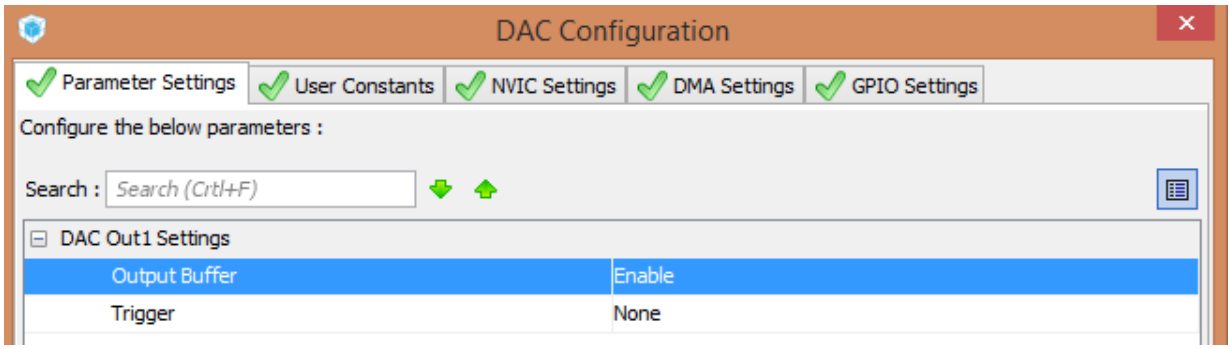
Configuración de StmCubeMx.

En el software creamos un nuevo proyecto, seleccionamos el micro controlador stm32f407vg, para después configurar los recursos a emplear.

Para esta práctica simplemente seleccionamos el DAC ya que Ophyra cuenta con dos de una resolución de 12 bits seleccionamos el canal 1.



En parameter settings lo único que se debe realizar es la habilitación de Output Buffer.



Generamos el código para Atollic true Studio.

En el archivo main.c en la función principal programamos.

```
int main(void)
{

  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_DAC_Init();

  uint16_t conta=0;
   HAL_DAC_Start(&hdac,DAC_CHANNEL_1);
   HAL_DAC_SetValue(&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,0);
  while (1)
  {
  if(conta==4095){conta=0;}
  else{conta++;}
  HAL_DAC_SetValue(&hdac,DAC_CHANNEL_1,DAC_ALIGN_12B_R,conta);
  }
```

Generamos una variable de 16 bits para el conteo

Inicializamos el DAC

Es ta funció establece canal 1 del DAC en cero

Preguntamos si el contador ha llegado a 4095 si es asi reiniciamos el contador si no incrementamos en una unidad al contador

En el canal 1 del DAC establecemos el valor del contador con una alineación a la derecha

La grafica obtenida es la siguiente.

Analizando la gráfica para obtener su pendiente tenemos

$$x = [2.5\ cuadros]\left[\frac{5ms}{1\ cuadro}\right] = 12.5ms$$

$$y = [3.3\ cuadros]\left[\frac{1V}{1\ cuadro}\right] = 3.3v$$

$$Pendiente = \left[\frac{y}{x}\right] = \left[\frac{3.3}{12.5 * 10\text{^} - 3}\right] = 264$$

Practica 5. Uso del DMA y el DAC para la graficar una señal seno.

Para poder obtener nuestra señal seno se debe hacer el uso de una ecuación la cual es

$$seno = \left[\sin\left(\frac{2\pi}{ns}\right)t + 1\right]\left[\frac{0xFFF + 1}{2}\right]$$

$$\left[\mathrm{Sin}\left(\frac{2\pi}{ns}\right)t + 1\right][2048]$$

Donde ns =el número de muestras en el periodo de una señal.

Para generar una señal a 100 Hz a un periodo de 100 muestras tendremos un tiempo de muestreo de 100 μs.

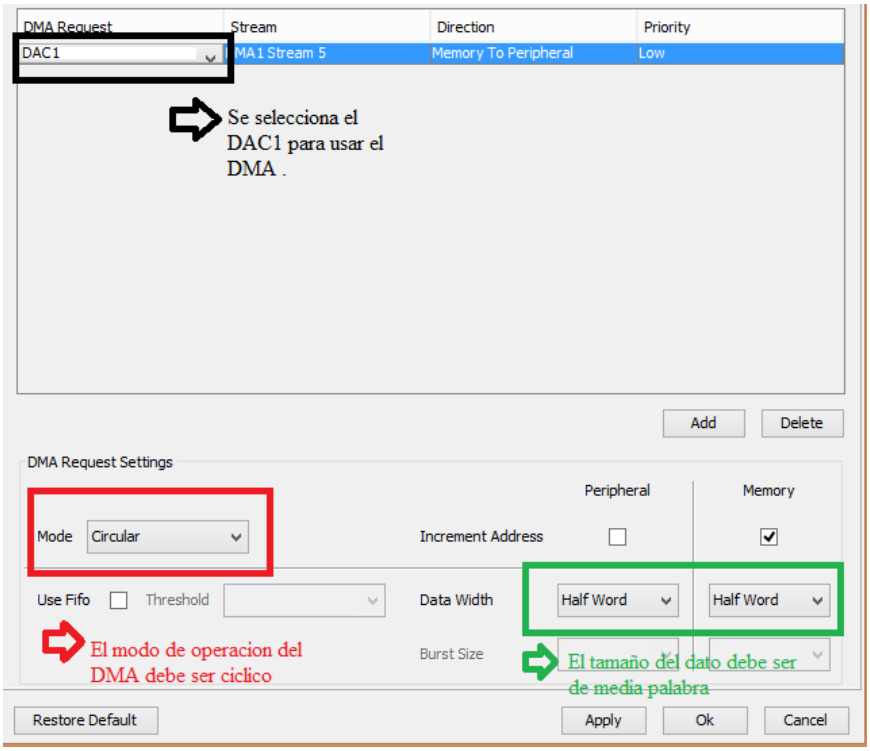
Establecemos un prescaler de 100

Por lo que tendremos incremento en el timer de 100/16MHz=6.25 μs

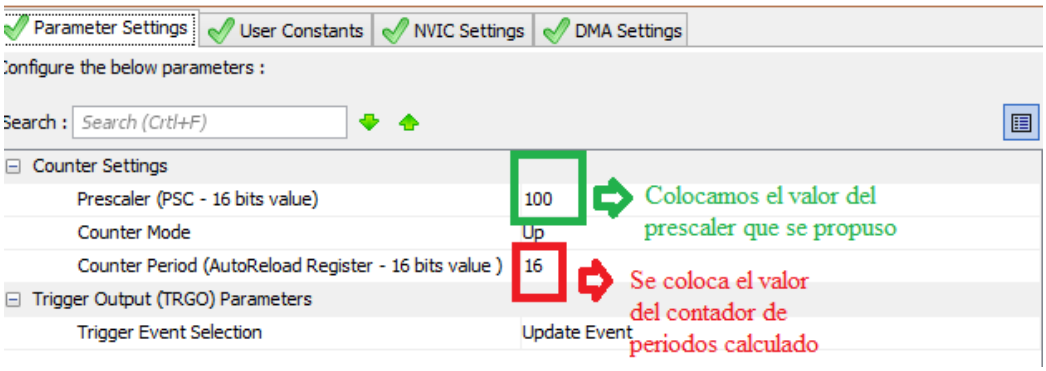
Y un contador de periodos de 100μs /6.25 μs =16

Configuraciones software StmCubeMx creamos un nuevo proyecto, seleccionamos el micro controlador stm32f407vg, para después configurar los recursos a emplear.

Se realizan las configuraciones similares a la práctica anterior con la configuración del DMA a utilizar.



Configuración del Timer 6 que está asociado al DAC.



Generación del código para Atollic True Studio.

Nos situamos en archivo main.c incluimos la librería matemática que se ocupara y la declaración de dos constantes muestras con valor de 100 y pi con valor de 3.14159

```
#include "math.h"
#define pi 3.14159
#define muestras 100
```

Debemos declarar un buffer de 16 bits llamado datos con el tamaño de muestras y la variable de16 bits result.

```
uint16_t datos[muestras],result;
```

En la función principal.

```
int main(void)
{

  HAL_Init();

  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_DAC_Init();
  MX_TIM6_Init();

  /* USER CODE BEGIN 2 */
  for(int k=0;k<muestras;k++)
  {
    result=rint((sinf(k*2*pi/muestras)+1)*2048);
    datos[k]=result<4095?result:4095;
  }
  //INICIALIZA EL TIMER 6
  HAL_TIM_Base_Start(&htim6);
  //INICIALIZA EL DMA
  HAL_DAC_Start_DMA(&hdac,DAC1_CHANNEL_1,(uint32_t*)datos,muestras,DAC_ALIGN_12B_R);

  while (1)
  {

  }
```

El ciclo for realiza 100 iteraciones una por por cada muestra que se toma

Aplica la ecuación para obtener la señal seno por cada iteración y lo guarda en la variable result

Inicializa el timer 6

Inicializa el DMA del DAC, pidiendo por parametro el puntero del dac, el cnal a emplear , los datos en tamaño de 32 bits, el numero de muestrars y la aliniación a la derecha

Esta instrucción verifica que la variable datos y result en cada iteracion no supere el valor de 4095 si pasa lo coloca a 4095

Grafica de la señal obtenida.

Conclusión.

Los recursos aprendidos de este micro controlador en esta unidad son varios y con diversas aplicaciones , desde el manejo de tiempos sin ocupar retardos , sino generando un sistema base tiempo, con el cual podemos configurar interrupciones por timer para generar aplicaciones que requieran llevar contadores o mediciones de tiempos, aprendimos a utilizar , el PWM generado por Timer el cual nos permite el control de servomotores los cuales son muy utilizados en aplicaciones mecatrónicas, se uso el ADC para el manejo señales de voltaje analógicas las cuales pueden ser utilizadas para tratar la información proveniente de un sensor ,se hizo uso del DMA el cual es un modulo de transferencia de datos de alto rendimiento con el cual transfieres datos de sensores al microcontrolador para su procesamiento y después enviar una señal al actuador como es el caso de los servomotores y los potenciómetros sin la necesidad de utilizar el CPU se realizó la transferencia de estos datos y se utilizo el DAC con el que cuenta el micro controlado el cual nos permitió generar señales analógicas desde microcontrolador como es la señal de diente de sierro a una senoidal.

Bibliografía.

Perles Àngel; ARM Cortex-M práctico. 1 - Introducción a los microcontroladores STM32 de St, Univercidad Politécnica de Valencia, Valencia 2017,206 págs.