FILTRO PASABAJO IMPLEMENTADO CON EL MICROCONTROLADOR ATMEGA 2560

Tobón Collazos, Daniel.

{danito\_bon@hotmail.com}

Pérez Arce, Juan Camilo

{kamilo199603@gmail.com}

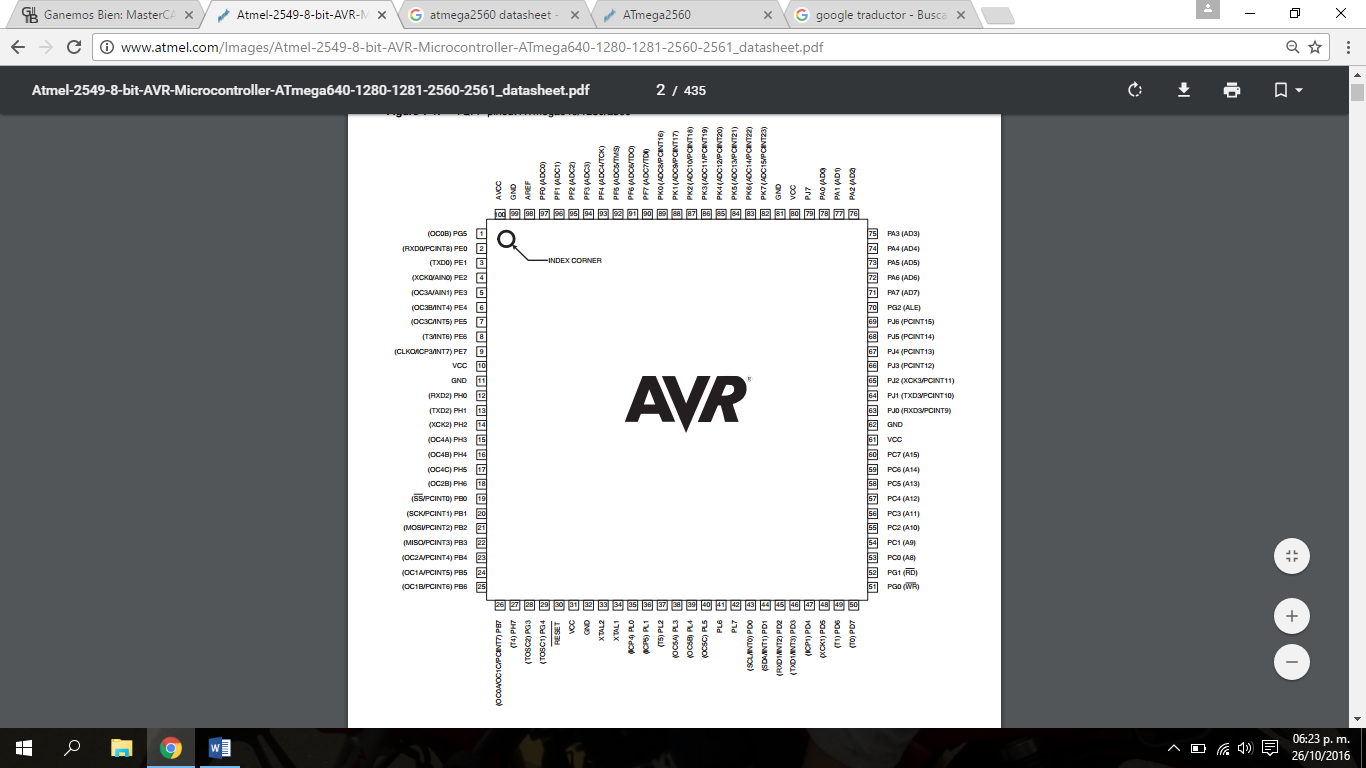
*Resumen*— El presente laboratorio es la implementación de un FITRO PASABAJAS utilizando el microcontrolador Atmega . Con base en la teoría de los microprocesadores, se pretende muestrear los valores de tensión de una señal seinusoidal proveniente de un generador de funciones cada 1 ms, a través de un conversor análogo-digital integrado en el propio microcontrolador; luego, discretizar, cuantizar y codificar los datos de entrada en su representación binaria en dónde se procesa la señal de forma digital aplicando un filtro pasabajo con frecuencia de corte de . Después, la señal filtrada es sacada hacia un conversor digital-análogo en dónde se observa la señal continua en un osciloscopio UTD2102CEL. Lo que se pretende es desarrollar el entendimiento en los conversores de señal utilizando los microcontroladores de la familia AVR y el software de simulación Atmel Studio . La simulación del circuito se realizó con el programa Proteus . Por último, se graficó la respuesta en frecuencia del filtro pasabajas en el software matemático Matlab.

*Índice de Términos.* Filtro; AVR; Pasabajas.

# Introducción

Este documento introduce el tema al manejo de un filtro pasabajas implementado con un microcontrolador atmega de la familia avr. Con base en la teoría de los microprocesadores, se pretende desarrollar el entendimiento en el manejo de un microcontrolador cómo un filtro pasabajas; el Atmega [[1]](#footnote-1) [1] es un dispositivo electrónico de bajo consumo que utiliza la arquitectura RISC y puede alcanzar rendimientos de hasta millones de instrucciones por segundo operando con voltajes de alimentación . La figura 1 muestra los 100 pines que conforman el microcontrolador Atmega 2560 indicando las funciones que pueden desempeñar, conversores de señal análogo a digital, PWM, temporizadores, contadores, etc.

*Figura 1. Microcontrolador Atmega 2560.*



Fernández López, David Fernando

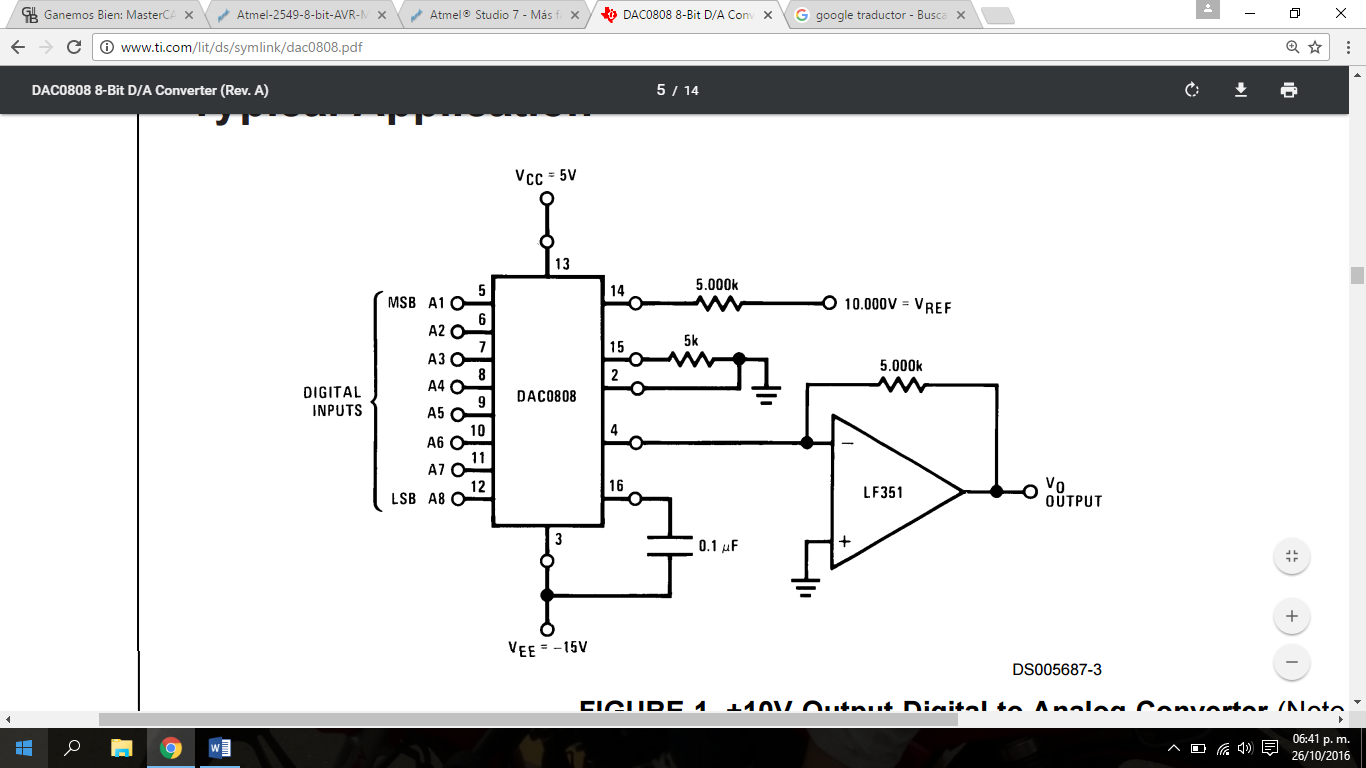
{fdez-david@hotmail.com}

**Fuente:** TQFP-pinout. ATmega640/1280/25608-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash Datasheet.

El microprocesador cuenta con su set de instrucciones o códigos de programación para el lenguaje ensamblador; el software Atmel Studio 7.0 [2] permite simular la programación del código en un entorno de desarrollo integrado que involucra cada registro de control y status.

De forma general, se definieron los puertos de entrada y de salida del microcontrolador; el puerto F recibe la señal seno del generador de funciones y el puerto B saca la señal digital a un conversor digital-análogo. Hay que recalcar que este microcontrolador no cuenta con un conversor DAC integrado, por lo que externamente se utilizó el DAC0808 [3]. La figura 2 muestra la conexión típica del conversor digital-análogo DAC0808 tomado del datasheet del dispositivo.

*Figura 2. Conexión típica del DAC0808.*

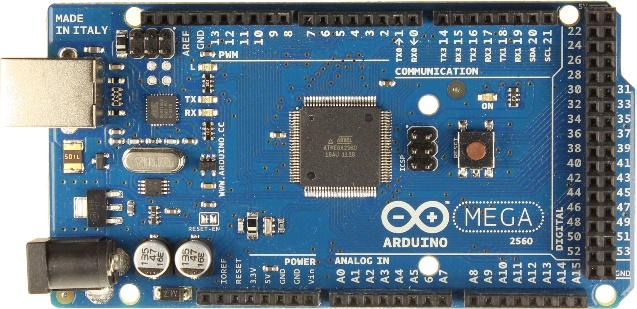


**Fuente:** DAC0808 8-bit D/A converter datasheet. National Semiconductor.

Como se observa, cuenta con 8 entradas digitales, por lo que una resolución de 8 bits del ADC es suficiente para representar la señal de forma digital.

Ahora bien, la placa Atmega 2560, (figura 3), tiene integrado el microprocesador AVR, puertos de I/O, alimentación VCC=5V, cristal de cuarzo de 16 MHz y todos los componentes básicos necesarios para la implementación del circuito.

*Figura 3. Atmega 2560 de arduino.*

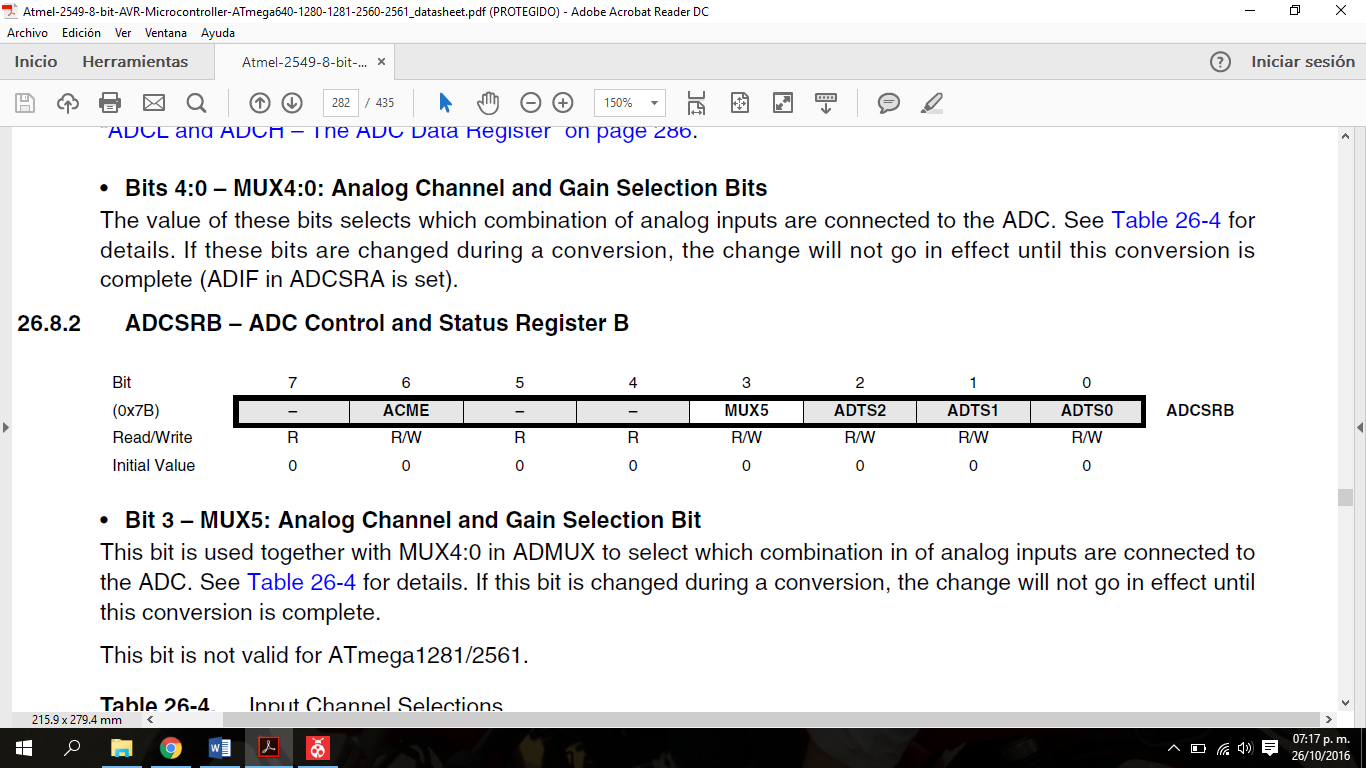


**Fuente:** Arduino mega 2560 Rev3. Paruro.pe, tu tienda online de electrónica.

# Configuración del ADC del Atmega 2560

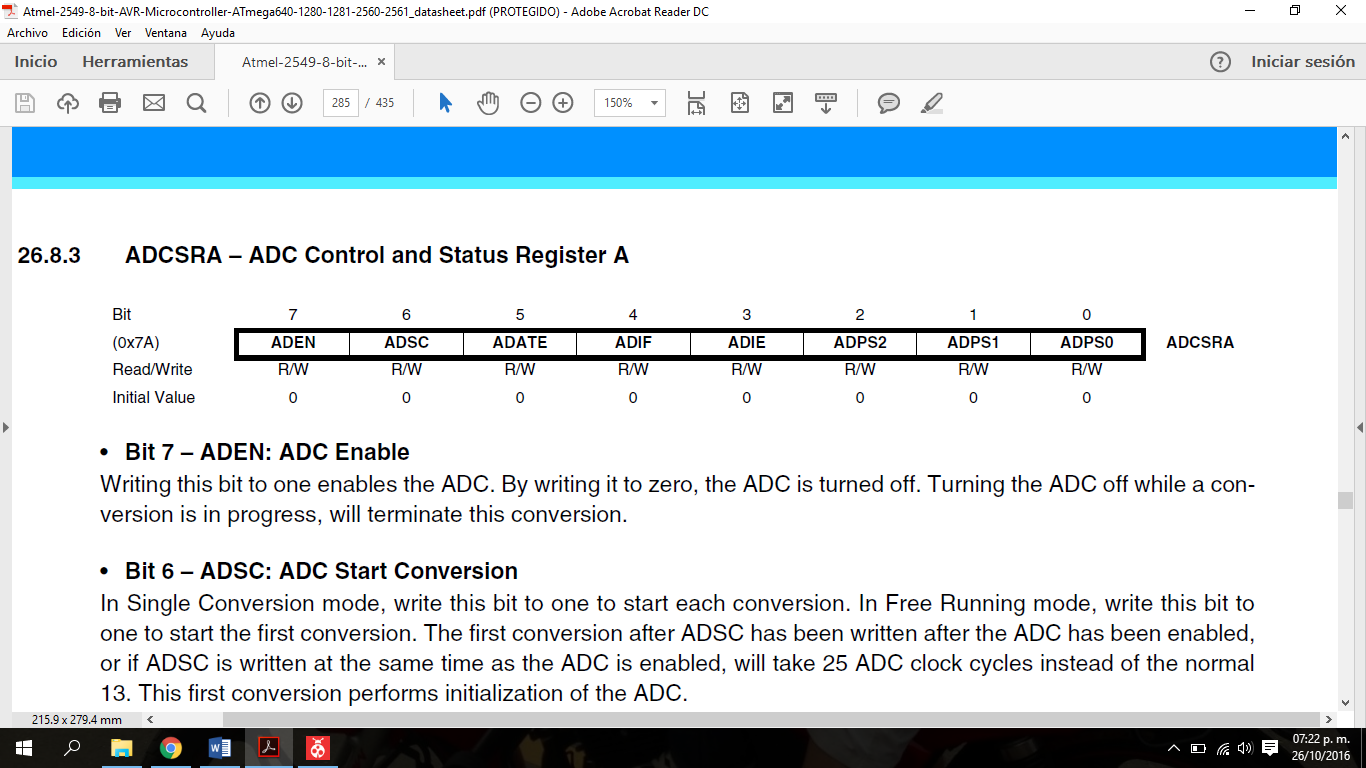
El primer paso consiste en configurar el micro para que reciba los valores de tensión de entrada y los convierta a una señal digital en dónde luego pueda ser procesada o transmitida. El datasheet muestra en la sección del ADC los registros de control y status. Las imágenes que se van a mostrar a continuación son tomadas del datasheet del dispositivo AVR.

El registro ADCSRB permite seleccionar el ADC a utilizar en el micro. Para este caso, se va a utilizar el ADC0, por lo que se ponen en cero los bits , y . Por defecto, los registros se inicializan con cero, por lo que no es necesario realizar una carga de datos a este registro. Solo se quiere dar a entender que se va a trabajar con el ADC0.

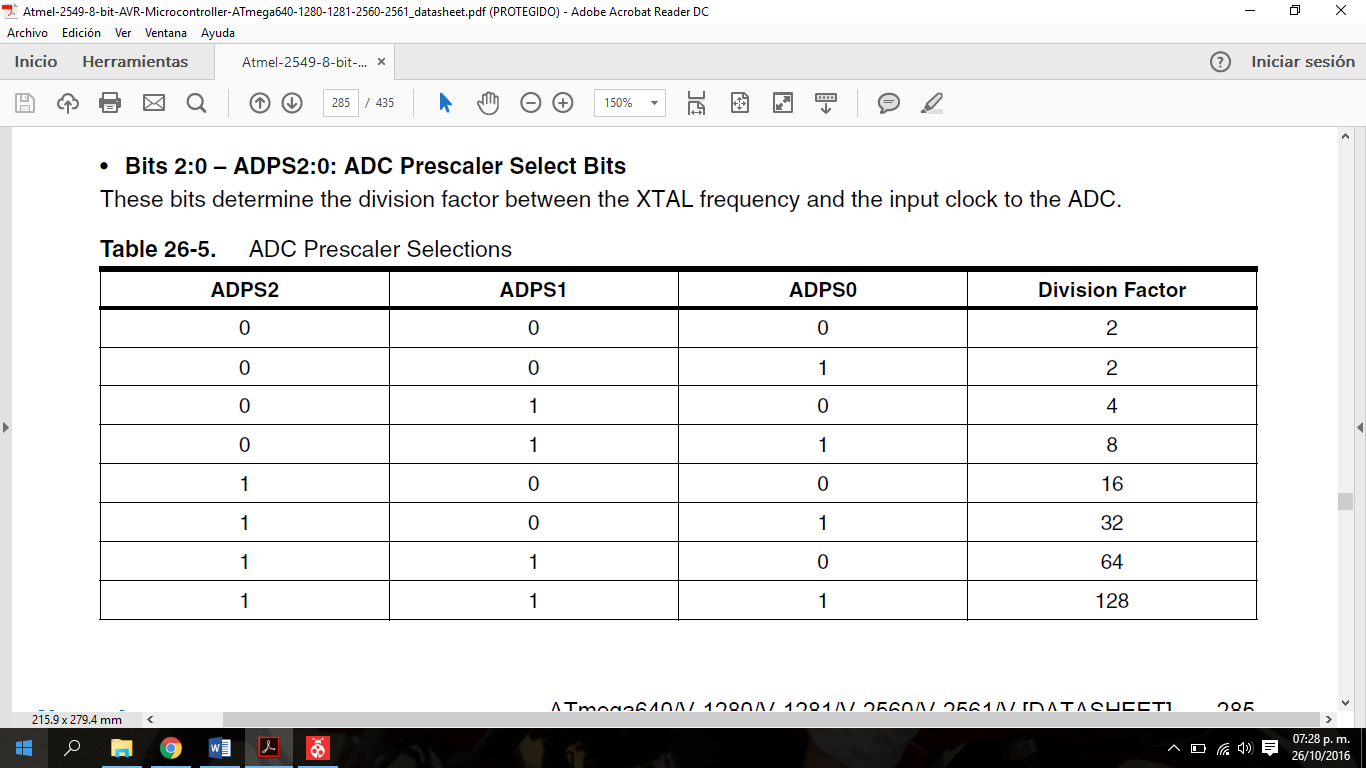


Ahora bien, ya seleccionado el ADC que se va a utilizar, se procede a manejar el registro que habilita el uso del ADC, ya que por defecto y por consumo energético, viene deshabilitado.

El registro ADCSRA controla la habilitación del ADC, el inicio de conversión del dato, el prescaler, contiene la bandera de final de conversión, interrupción por final de conversión y un bit de control para la auto conversión.



Los bits , y corresponden al prescaler del ADC, es decir, el factor de división que divide la frecuencia de trabajo del micro. La tabla de la siguiente figura muestra la combinación de bits, para seleccionar un factor de 2, 4, 8, 16, 32, 64 y 128.



El micro tiene un cristal de cuarzo de , es decir, que la frecuencia o la velocidad con la que ejecuta las instrucciones es de 16 millones por segundo. El ADC necesita una frecuencia o velocidad de trabajo adecuada por dos motivos; una frecuencia alta para el ADC implica que la conversión de datos es muy rápida, pero el valor digital puede no ser muy confiable. Ahora bien, una frecuencia baja implica que el valor digital es confiable, pero el tiempo de conversión del dato es demorado. El datasheet del fabricante indica el rango del ADC entre y , esto quiere decir que poner a trabajar el ADC en una frecuencia intermedia entre ese rango es lo óptimo, ya que el dato es confiable y la conversión no es lenta. Es en este punto, en donde entra la importancia del prescaler, ya que es el factor que determina la velocidad de trabajo del ADC. La frecuencia o velocidad de trabajo del ADC se define cómo:

Dónde:

🡪 Frecuencia del ADC

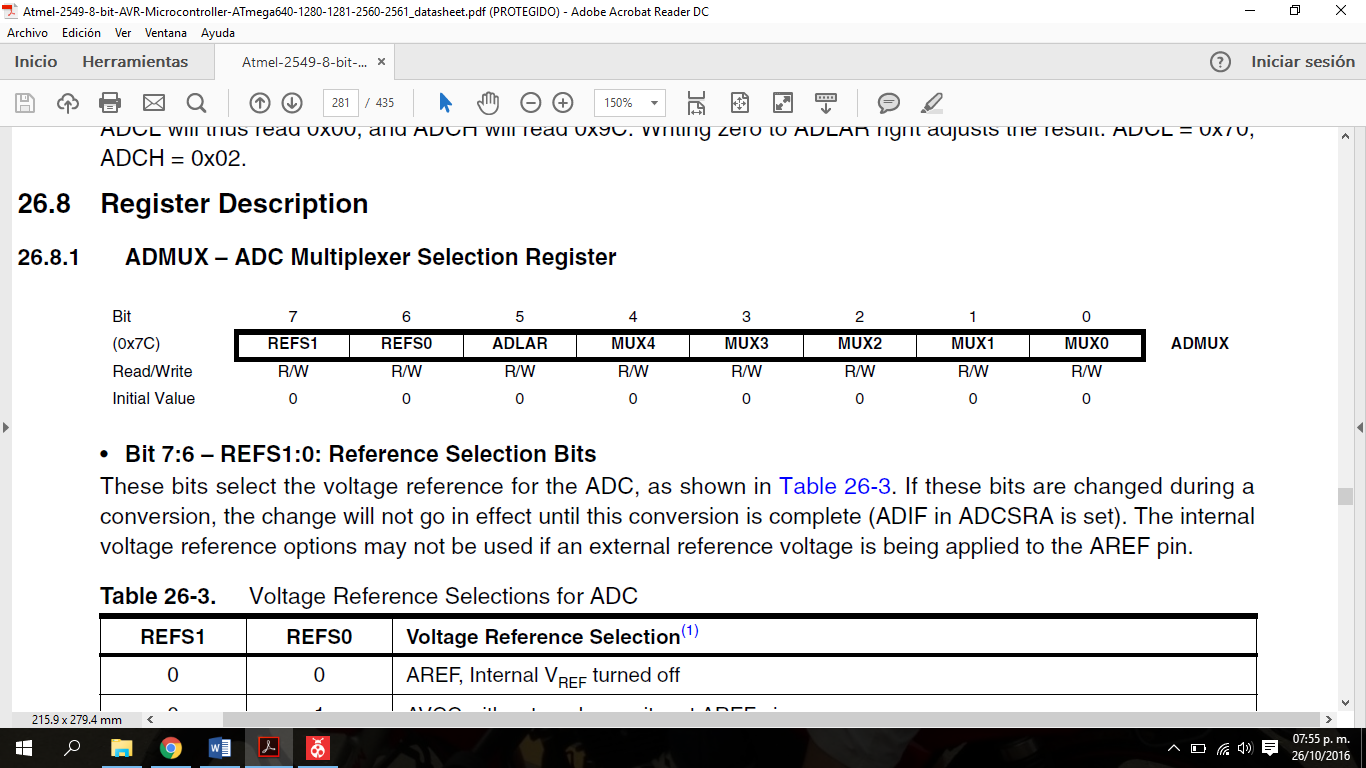
🡪 Frecuencia del microprocesador

Para este caso, la frecuencia del microprocesador es de . La frecuencia de trabajo del ADC debe ser aproximadamente la mitad del rango, es decir, . Reemplazando estos valores en la ecuación (1) y despejando el prescaler daría:

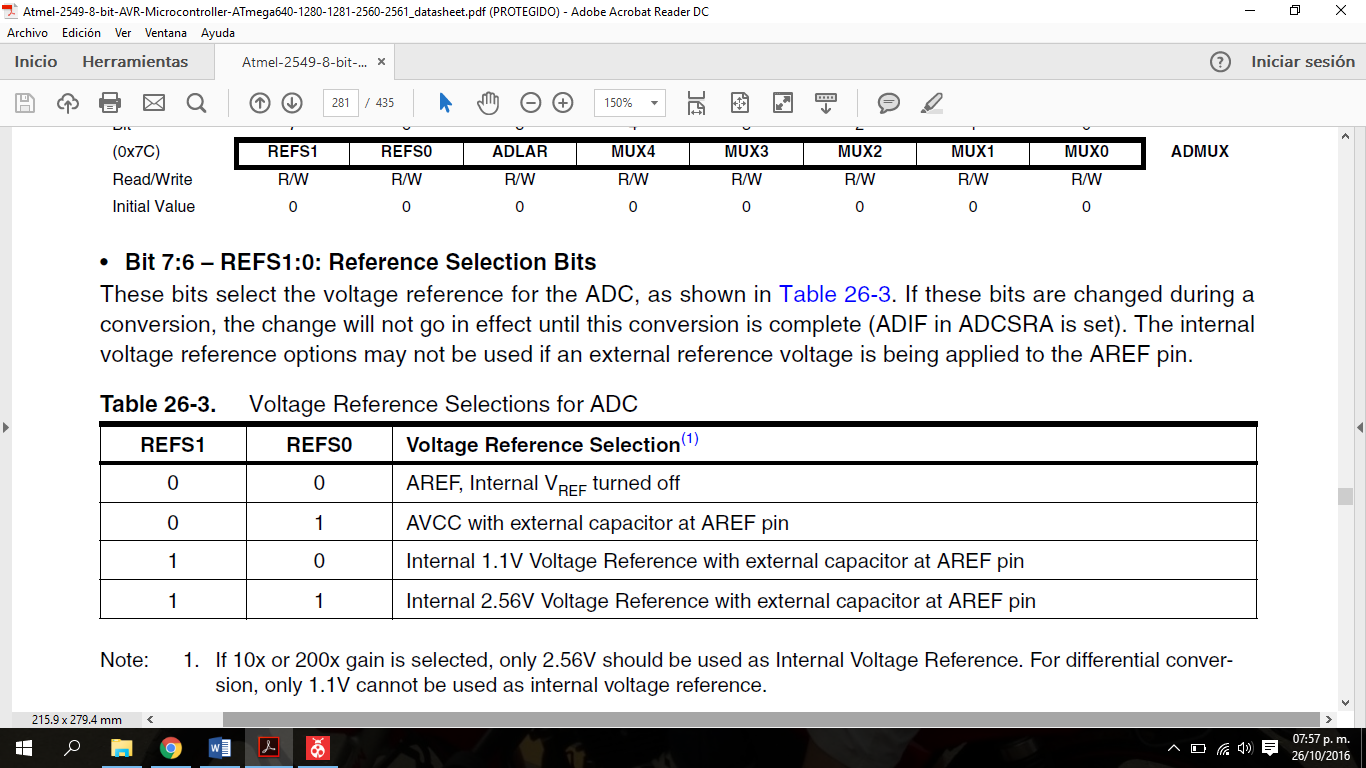
Esto indica que, de la tabla anterior, utilizando un cristal de y seleccionando un prescaler de se va a garantizar que la velocidad de conversión del ADC va a ser de , un valor intermedio del rango de velocidad, lo cual, es lo óptimo. Teniendo esto en cuenta, se cargan los tres bits menos significativos del registro ADCSRA con 1 para seleccionar el prescaler de 128.

Por otra parte, en esta aplicación no se necesita la interrupción de final de conversión del ADC, por lo que, el bit ADIE se mantiene en cero. De la misma manera, el bit ADATE que es el autotrigger no se utiliza, también se mantiene en cero.

Los tres bits restantes si son importantes; el bit 7, ADEN habilita el ADC. Este bit se setea en 1, luego el bit ADIF es el bit de bandera que indica el final de conversión. Cuándo este bit se pone en 1 indica que la conversión ha terminado. Por último, el bit ADSC es el bit de inicio de conversión. En otras palabras, para este caso en particular, el registro se carga con el número binario (). El último registro de control para el ADC es el ADMUX; en este registro se selecciona el voltaje de referencia del ADC, si va a ser de una fuente externa, AVCC de 5V o las fuentes internas del micro de 1.1V o 2.5V.



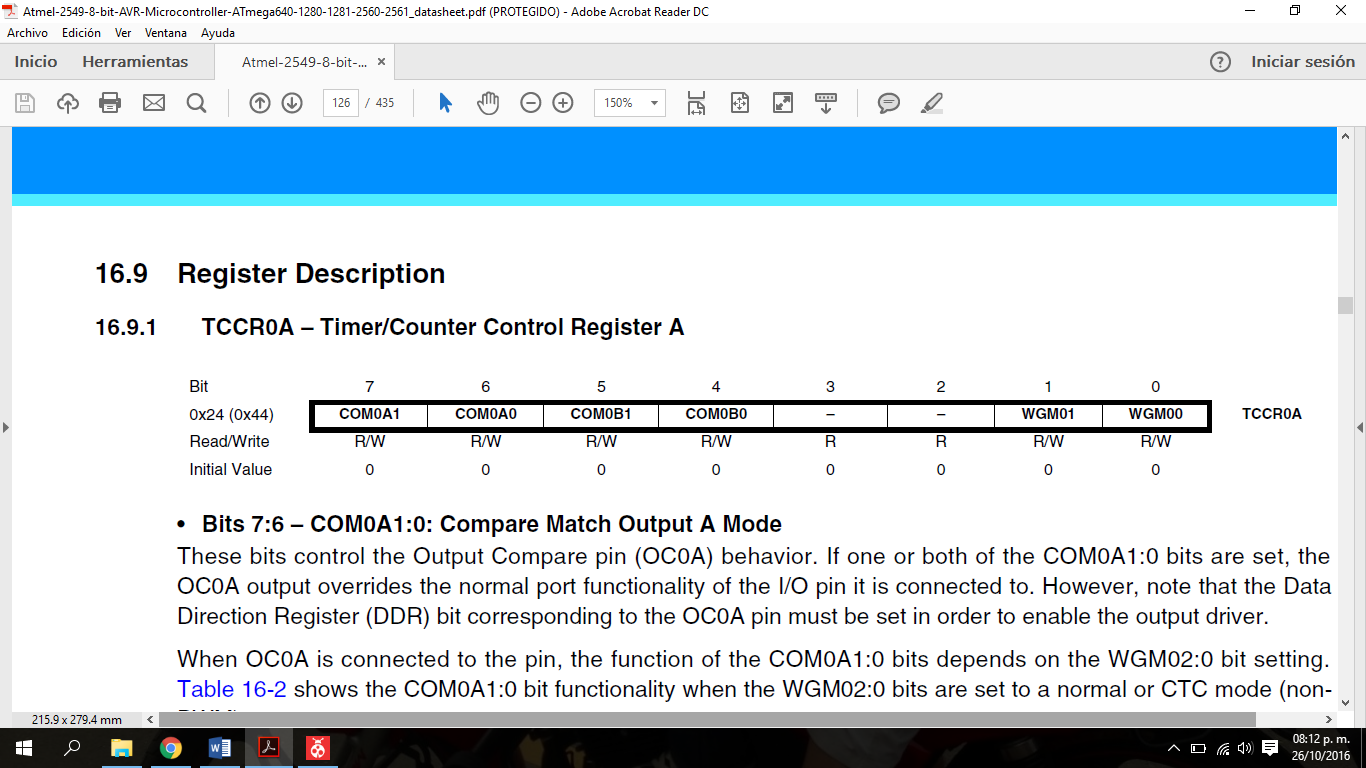
La combinación de bits que se muestran en la siguiente tabla describen la selección del voltaje de referencia. Para esta aplicación se seleccionó AVCC.



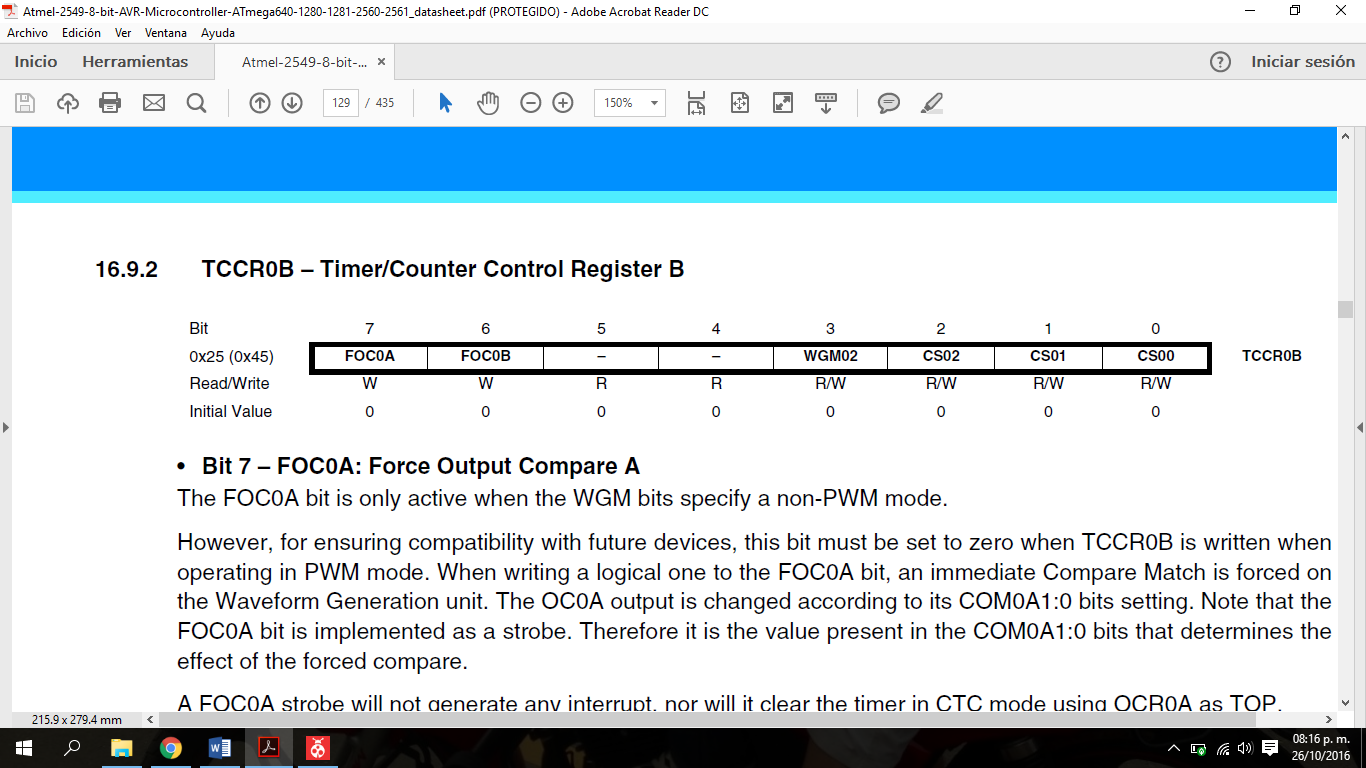
Por último, el bit ADLAR se cargó en 1 para seleccionar la justificación izquierda del dato convertido y guardarlo en ADCH.

# Configuración del Timer del Atmega 2560

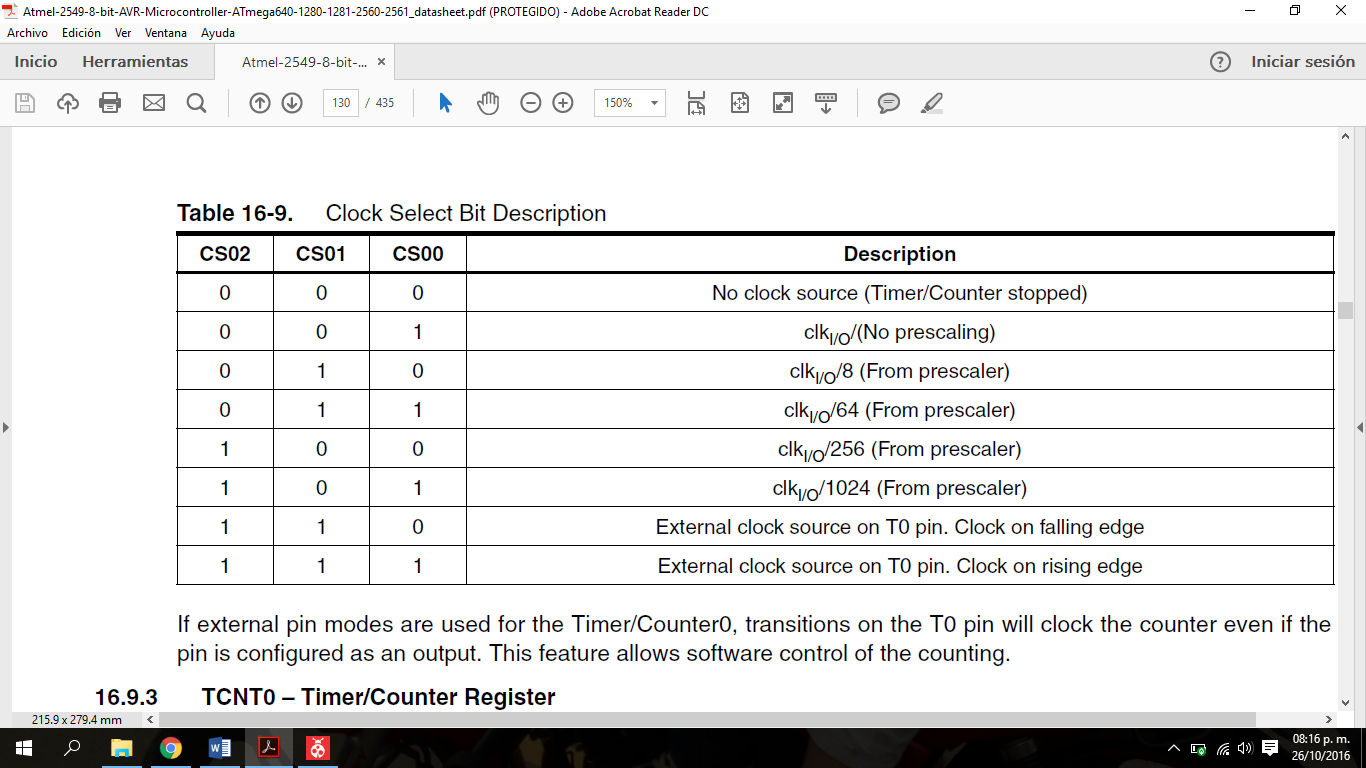
Cómo se requiere muestrear los datos de entrada al ADC cada , es necesario utilizar un timer que temporice el tiempo de muestreo de la señal. El timer 0 del AVR es un temporizador de 8 bits, permite temporizar máximo , utilizando un cristal de en el micro. Al igual que se hizo con el ADC, el timer se va cargando los registros de control y status con los bits que se van a utilizar para habilitarlo. El registro TCCR0A permite seleccionar el modo de trabajo del timer. Para esta aplicación se requiere una operación normal por lo que este registro se deja en cero, no se modifica.



Ahora bien, al igual que se hizo con el ADC, el timer también tiene su propio factor de división o prescaler. Su función es demorar la temporización del timer. El registro TCCR0B permite seleccionar el prescaler a utilizar, siendo los bits CS00, CS01 y CS02 los bits del prescaler.

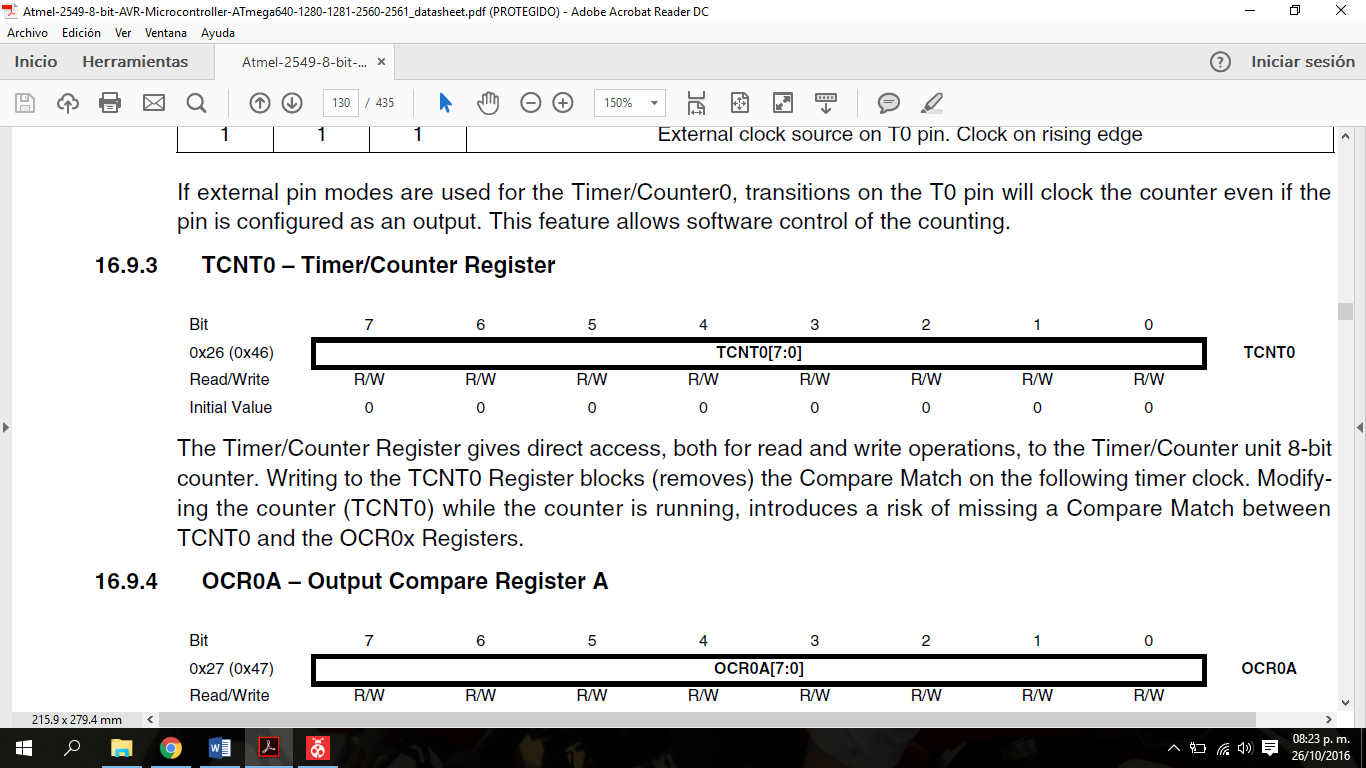


La siguiente tabla muestra factores de división de 1, 8, 64, 256 y 1024. En esta aplicación se seleccionó el prescaler más grande, de 1024.



El registro que almacena la cuenta de la temporización es el registro TCNT0; cómo el timer 0 es de 8 bits, este registro puede temporizar un total de ciclos de reloj. Cómo el Atmega tiene un cristal de 16 MHz y se seleccionó un prescaler de 1024, entonces el tiempo máximo que puede almacenar este registro es:

Con un total de 255 ciclos:



Para temporizar 1 ms se debe ubicar el punto de partida de ese registro de manera que cuando llegue a 255 haya transcurrido 1 ms.

El valor que se carga en el registro TCNT0 está dado por:

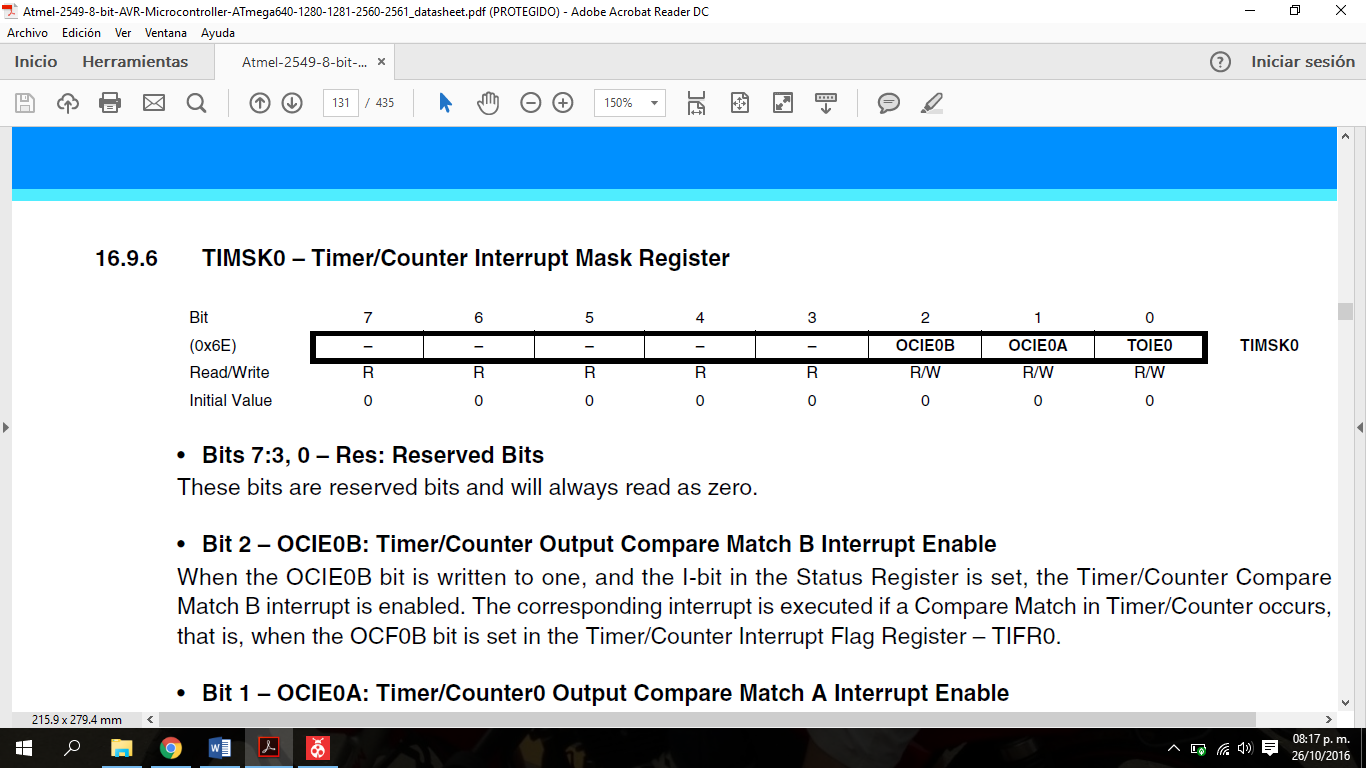
Dónde:

🡪 Tiempo de retardo (1 ms para esta aplicación)

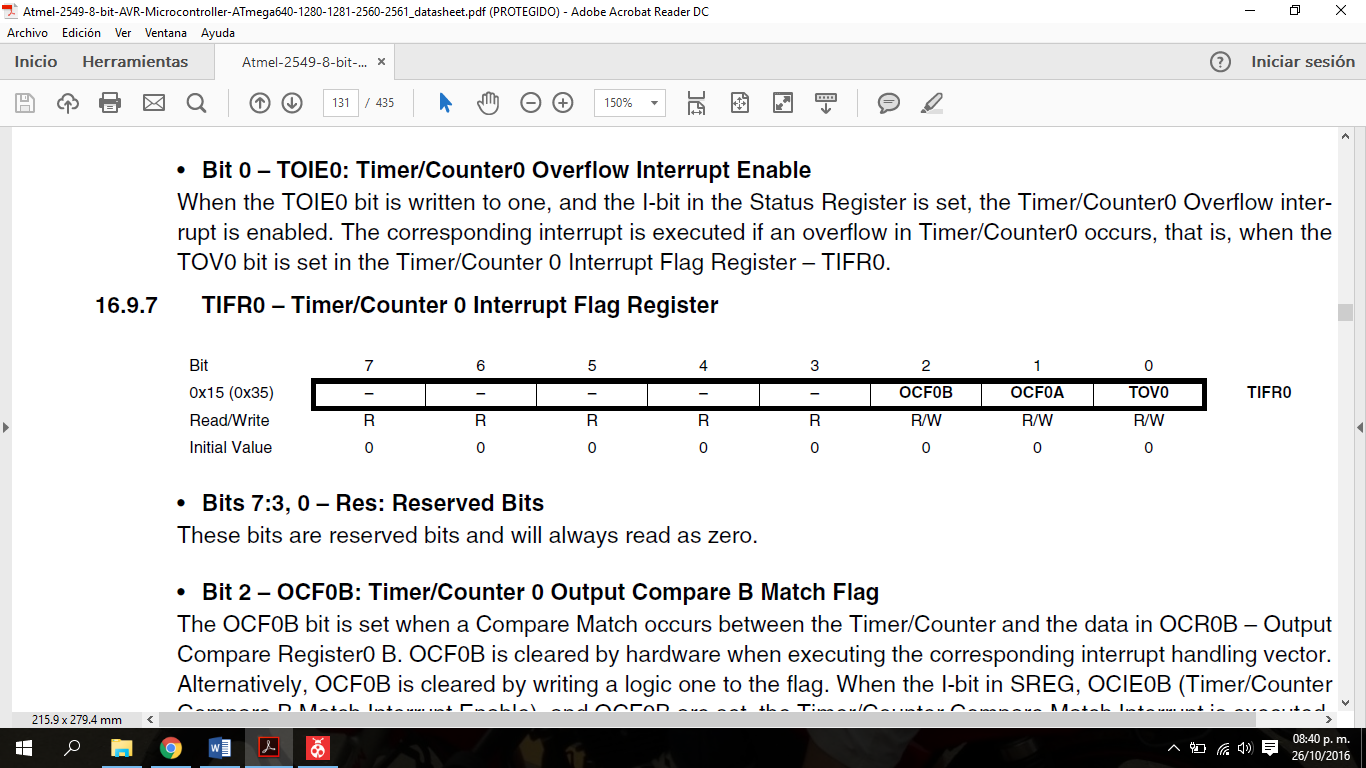
Reemplazando el prescaler de 1024 y una frecuencia de 16 MHz, quedaría:

Esto quiere decir, que cargando el número 239 en el registro TCNT0, al temporizar los ciclos de reloj hasta llegar a 255, habrá transcurrido 1 ms.

El registro final de control para el timer es el registro que habilita la interrupción. Se necesita que cuando transcurra 1 ms, el programa se detenga y atienda la rutina de atención a la interrupción por timer y que lea el dato del ADC, lo cargue, lo filtre y lo envíe al puerto de salida. Luego se recarga el temporizador y vuelva y lea el dato cada 1 ms. El registro TIMSK0 permite habilitar la interrupción por desbordamiento del timer (overflow), seteando el bit 0 del registro .



A diferencia del ADC, el temporizador tiene su bit de bandera en otro registro el cual, indica si ha ocurrido un desbordamiento. El registro TIFR0 es el registro que contiene el bit de bandera TOV0 el cual, indica si el timer se ha desbordado.



Con esto, se ha configurado todo lo necesario para el funcionamiento del filtro pasabajos en el microcontrolador Atmega 2560. En la siguiente sección se detalla el código en lenguaje ensamblador que se empleó para la adquisición y procesamiento de datos.

# Lenguaje ensamblador: filtro pasabajos

El código en lenguaje ensamblador que se empleó para el desarrollo de este laboratorio se muestra a continuación:

.ORG 0x00

JMP START

.ORG 0x2E

JMP OVERFLOW

.DEF TEMP=R16

;===================================================

; INICIO PROGRAMA

;===================================================

START:

//REFERENCIA VOLTAJE DEL ADC (AVCC=VCC=5V)

LDI TEMP,(1<<REFS0)|(1<<ADLAR) ;CARGA 1 EN EL BIT REFS0 AL REGISTRO 16 (REFERENCIA EXTERNA AVCC DEL ADC)

STS ADMUX,TEMP ;ALMACENA EN EL REGISTRO ADMUX UN 1 LÓGICO EN EL BIT REFS0

LDI TEMP,(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)

STS 0x7A,TEMP ;HABILITA EL ADC

//HABILITAR PUERTO DE SALIDA

LDI TEMP,0xFF ;CARGA unos EN EL REGISTRO 16

OUT DDRB,TEMP ;DEFINE CÓMO SALIDA EL PUERTO B

//HABILITAR PUERTO DE ENTRADA

LDI TEMP,0x00 ;CARGA ceros EN EL REGISTRO 16

OUT DDRF,TEMP ;DEFINE CÓMO ENTRADA EL PUERTO F (PUERTO DEL ADC)

;===================================================

;CONFIGURACIÓN DEL TIMER

;===================================================

LDI R16,(1<<CS02)|(1<<CS00) ;PRESCALER DE 1024

OUT TCCR0B,R16 ;TCCR0B ES EL REGISTRO DONDE SE DEFINE CUÁL PRESCALER SE VA A UTILIZAR

LDI R16,(1<<TOIE0) ;HABILITACIÓN DE LA INTERRUPCIÓN POR TIMER (OVERFLOW)

STS TIMSK0,R16 ;TIMER INTERRUPT ENABLE (OVERFLOW)

SEI ;HABILITACIÓN GLOBAL DE INTERRUPCIONES

//INICIO DE CONVERSIÓN A/D

LDI R16,239 ;ES EL INICIO DEL TIMER, CUÁNDO LLEGUE A 255 HABRÁN TRANSCURRIDO 1ms

OUT TCNT0,R16 ;CUÁNDO LLEGUE A 255 OCURRE EL OVERFLOW

LDS TEMP,0x7A ; Registro ADCSRA

ORI TEMP,(1<<ADSC)

STS 0x7A,TEMP ; Genera el inicio de conversión

ESPERAR: ;RUTINA EN DÓNDE ESPERA QUE TERMINE EL TIEMPO

NOP

JMP ESPERAR ;DE UN 1ms

;===================================================

;RUTINA DE ATENCIÓN A LA INTERRUPCIÓN POR TIMER

;===================================================

OVERFLOW:

FinCONV:

LDS TEMP,0x7A

SBRS TEMP,4

JMP FinCONV

;===================================================

;DATO DE SALIDA DEL ADC

;===================================================

LDS R25,0x78 ;ADCL

LDS TEMP,0x79 ;ADCH

;===================================================

;FILTRO

;===================================================

MOV R18,TEMP ;X

LSR R18 ;X/2

LSR R18 ;X/4

LDS R19,0x04 ;X1

MOV R21,R19 ;R21=X1

LSR R19 ;X1/2

LDS R20,0x05 ;R20=X2

LSR R20 ;X2/2

LSR R20 ;X2/4

ADD R18,R19 ;SUMA: X\*0.25+(2\*0.25\*X1)

ADD R18,R20 ;SUMA: X\*0.25+(2\*0.25\*X1)+X2\*0.25

;===================================================

;ESCRITURA DEL FILTRADO EN EL PUERTO B

;===================================================

OUT PORTB,R18 ;ESCRIBE EN EL PUERTO B LA SALIDA DIGITAL FILTRADA

;===================================================

;RECARGA DEL TEMPORIZADOR/RE-INICIO DE CONVERSIÓN

;===================================================

LDI R17,239 ;ES EL INICIO DEL TIMER, CUÁNDO LLEGUE A 255 HABRÁ TRANSCURRIDO 1ms

OUT TCNT0,R17 ;CUÁNDO LLEGUE A 255 OCURRE EL OVERFLOW

LDS R17,0x7A ;ADCSRA

ORI R17,(1<<ADSC) ;

STS 0x7A,R17 ; Genera el inicio de conversión

;===================================================

;RETRASOS DE LA SEÑAL DE ENTRADA

;===================================================

STS 0x05,R21 ;X2=X1

STS 0x04,TEMP ;X1=X

RETI ;FIN DE LA RUTINA DE ATENCIÓN POR INTERRUPCIÓN POR TIMER

;===================================================

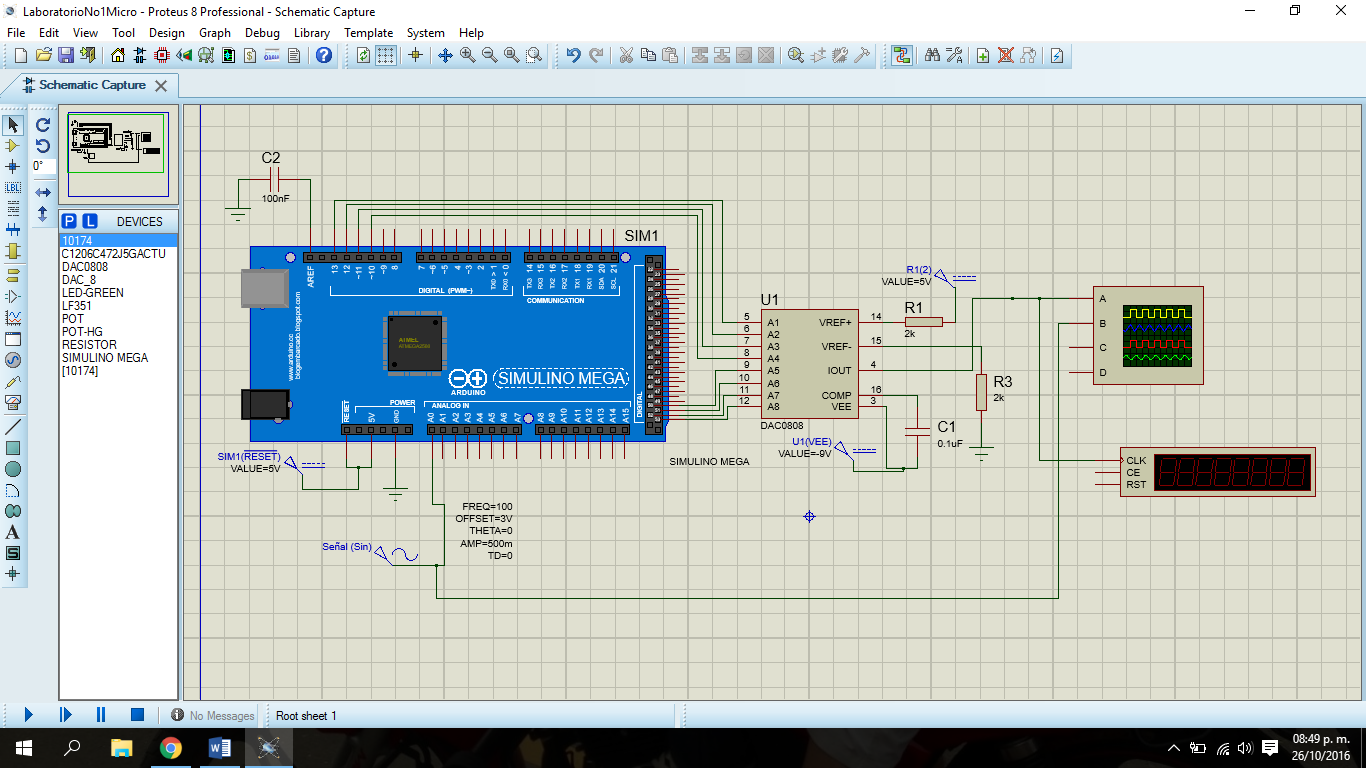
;FIN DE PROGRAMA

;===================================================

# Simulación: Proteus

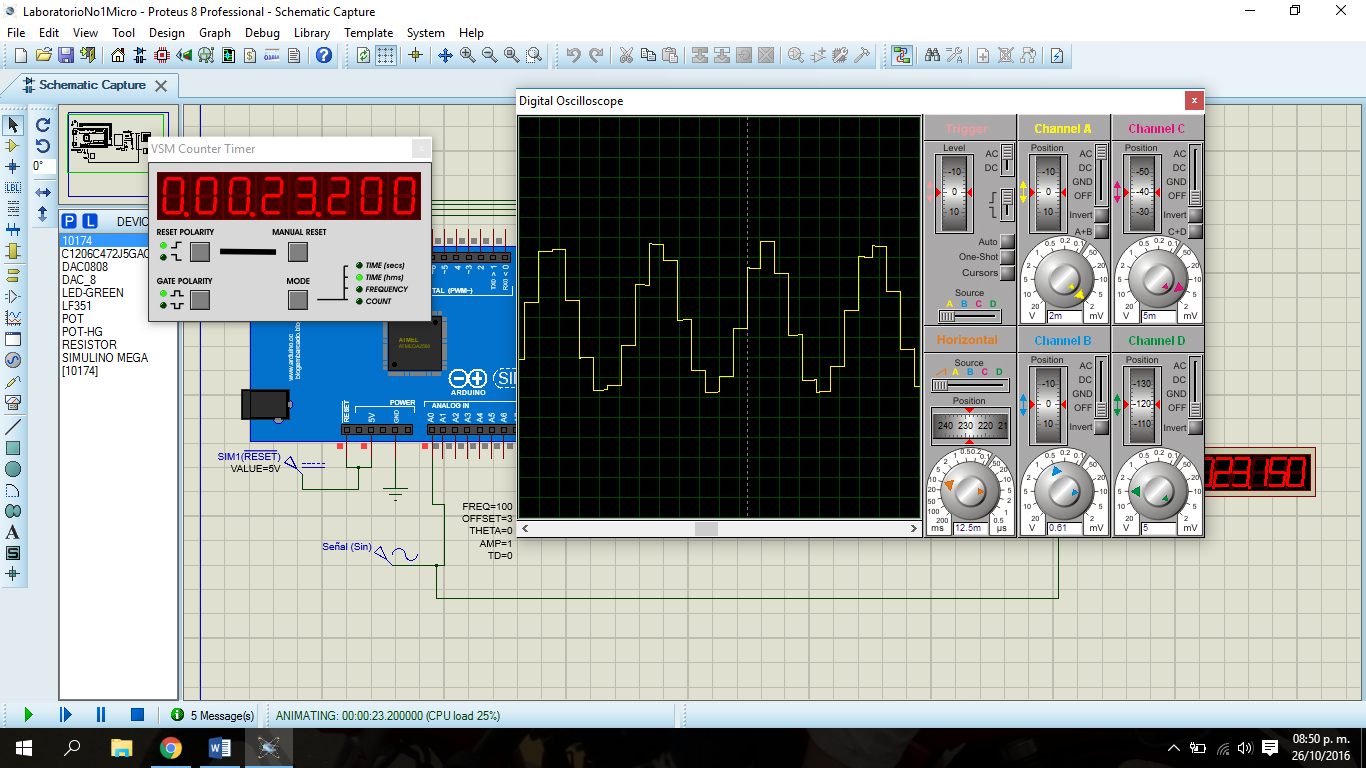
Con el objetivo de verificar el comportamiento del sistema se simuló en el software Proteus el funcionamiento del microcontrolador aplicando una señal sinusoidal de entrada de a . Proteus permite la integración de los microcontroladores, incluye la librería de arduino por lo que se escogió como programa de simulación, además permite cargar el programa de ejecución de código de Atmel Studio 7.0. La figura 4 muestra el esquema eléctrico del circuito en donde se observa las conexiones de los 8 bits digitales que van hacia el DAC0808 y la respectiva alimentación del micro.

*Figura 4. Esquema eléctrico del circuito.*



Por otra parte, la imagen del osciloscopio se aprecia en la figura 5, en dónde la retención de orden cero muestra la forma de onda digitalizada de forma cuadrada.

*Figura 5. Forma de onda digital.*



En la simulación en proteus no se incluyó el LF351 del datasheet del fabricante del DAC0808 debido a que no mostraba ningún comportamiento de la salida del sistema. En cuanto a la implementación del circuito si se incluyó el LF351.

En la siguiente sección se describe la respuesta en frecuencia del filtro utilizando el software matemático Matlab.

# Simulación: Matlab

Para realizar la simulación teórica del filtro digital pasa bajas se utilizó el software de simulación matemática MATLAB con el fin de realizar la una comparación entre la respuesta teórica y la respuesta práctica del filtro implementado. El análisis teórico del filtro digital implementado durante la práctica de laboratorio fue realizado a partir de la ecuación en diferencia que describe la respuesta del sistema, la cual se encuentra expresada como:

Al pasar por el proceso de conversión Análogo-Digital, la señal sinusoidal aplicada en la entrada del sistema pasa de ser una señal continua a ser una señal discreta. Esto quiere decir que el filtro digital implementado es un sistema de procesamiento de señales discreto. Para analizar el comportamiento del sistema y determinar sus principales características tales como la frecuencia de corte del filtro, su nivel de atenuación y su desfase, fue necesario graficar la función de trasferencia de dicho sistema obteniendo como resultado el diagrama de bode en el cual se puede observar la respuesta del filtro implementado. Para encontrar la función de transferencia del sistema fue se realizó la transformada Z de la ecuación en diferencia del filtro implementado obteniendo como resultado:

Ecuación en diferencia expresada en el dominio de Z:

Para encontrar la función de transferencia es necesario trabajar con términos de Z positivos por lo cual se realiza la siguiente operación:

*\**

Obtenido como resultado:

A partir de esta expresión se procedió a despejar la función de transferencia del sistema obteniendo como resultado:

Una vez obtenida la función de transferencia del sistema se procedió realizar el diagrama de bode en MATLAB implementando la siguiente serie de comandos:

num=[1 2 1];

den=[4 0 0];

Fs=1000;

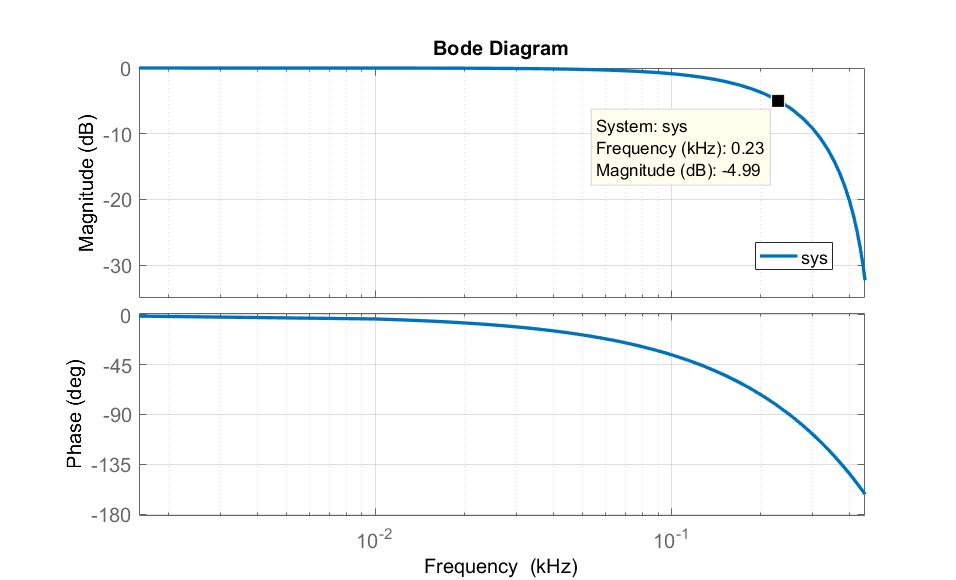
Ts=1/Fs;

dbode(num,den,Ts);

grid on

Obteniendo como resultado:

*Figura 6. Diagrama de bode filtro digital pasa bajas.*



En el diagrama de bode obtenido en la simulación es posible observar que la frecuencia de corte del filtro digital implementado es de , de igual forma se observa que el sistema presenta una respuesta de tipo butterworth, un desfase de …… y una atenuación de aproximadamente 20 Decibeles por década.

# Conclusión

Con este laboratorio se lograron aplicar todos los conceptos acerca de programación de sistemas embebidos en lenguaje de maquina adquiridos durante el semestre, logrando así realizar de manera exitosa la digitalización y el procesamiento de una señal analógica, permitiendo afianzar el conocimiento acerca del funcionamiento de los filtros digitales.

De igual forma es necesario tener claro el funcionamiento de cada uno de los comandos utilizados durante el proceso de programación, pues uno de los principales errores presentados durante el laboratorio radico en la mala interpretación del concepto de cada uno de los comandos

# Referencias

[1]. ATmega 8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash Datasheet. (en línea), disponible en: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561\_datasheet.pdf

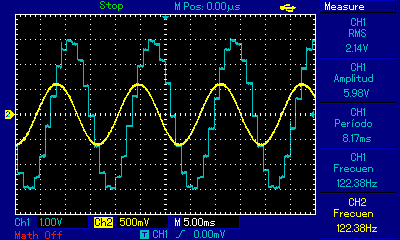
[2]. Atmel Studio 7.0 software. Microchip Atmel. (en línea), disponible en: http://www.atmel.com/Microsite/atmel-studio/

[3]. DAC0808 8-Bit D/A Converter datasheet. National semiconductor. (en línea), disponible en: http://www.ti.com/lit/ds/symlink/dac0808.pdf

# Anexos

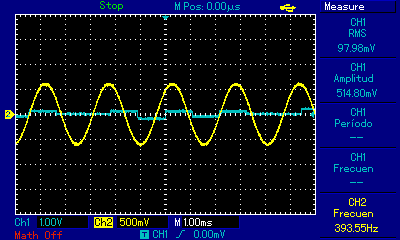
Como parte de la implementación del filtro, se montó en una protoboard el circuito y se cargó el programa al microcontrolador Atmega 2560. La señal digital se puede observar en la gráfica del anexo 1.

*Anexo 1. Gráfica de la señal de entrada y la señal de salida digital.*



La señal de salida (azul), tienen la misma frecuencia que la señal de entrada (amarilla), de frecuencia 122.38 Hz. Variando la frecuencia del generador de funciones se observó que, a frecuencias cercanas a los 300 Hz, la señal de salida se va atenuando. El anexo 2 muestra la atenuación de la señal de salida a 300 Hz.

*Anexo 2. Gráfica de la señal de salida atenuada.*



1. El Atmega 2560 es un microcontrolador de 8 bits con 16/32/64KB de memoria Flash programable. [↑](#footnote-ref-1)