Proyecto M3: Medición y Síntesis de frecuencia
Arquitectura de Computadores

Ezequiel Matsuo 69804/3

Febrero 2023

Índice

1. Introducción

El proyecto elegido es 'Medición de frecuencia para señales menores a 200 Hz y síntesis de senoidal de la misma frecuencia'. La idea es poder analizar la frecuencia de diferentes formas de onda (sinusoidal, triangular, cuadrada) y en base al cálculo de la frecuencia medida poder luego generar una señal sinusoidal de la misma frecuencia. El proyecto se dividió en dos partes, la medición de la frecuencia de entrada y la síntesis de la sinusoide. Para lograr esto se utilizaron los módulos Timer0 y Timer2 (para generar un PWM), el módulo UART para poder pasar los datos por el puerto serie, el módulo del conversor ADC para analizar los datos de la señal de entrada, e interrupciones.

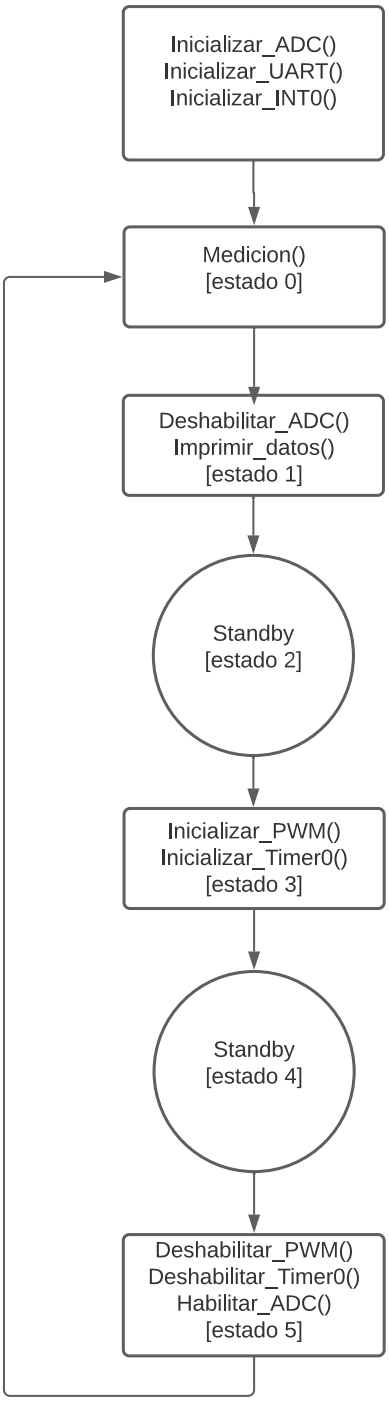
2. Resolución

Para la resolución, el proyecto se dividió el programa en 2 partes: medición y síntesis.

2.1. Estructura del programa

El esquema que se utilizó para conmutar entre la medición de frecuencia y síntesis de frecuencia se muestra en el diagrama de flujos de la figura 1. Estos se conmutarán a través de un pulsador.
Primero se inicializan el conversor ADC, el UART que será el encargado luego de imprimir los datos por el

puerto serie y la interrupción externa INT0, el cual a través de un pulsador se conmutará entre la medición y la síntesis.
Se definieron estados del 0 al 5 de forma de que el programa sea secuencial, el programa entrará a cada uno de los estados a través de condicionales *if* dentro del *while(1)* del main.
El estado 0 corresponde al bloque donde se realiza la medición, este estado no se encuentra explícito en el main del programa sino que se ubica dentro de una interrupción. Una vez que los parámetros necesarios para calcular la frecuencia de la señal de entrada fueron realizados, el programa inmeditamente pasa al estado 1, donde se deshabilita el ADC para que deje de medir y se imprime por pantalla la frecuencia obtenida a través del puerto serie. De este estado, se pasa inmediatamente al estado 2. En este estado 2 el sistema no hace nada, se queda en standby. Se decidió implementarlo de esta forma, ya que si dejamos que el programa se quede el estado 1, en cada iteración del bucle *while(1)* el programa estaría deshabilitando el ADC e imprimiendo pantalla. Durante este estado 2 el microcontrolador no hace nada, queda en espera de que el pulsador sea presionado para poder conmutar hacia la síntesis de frecuencia. En tanto no sea así, el programa quedará en el estado 2 indefinidamente.
Una vez que el pulsador sea presionado, el programa pasa al estado 3 para realizar la síntesis de frecuencia. Aquí se inicializan el Timer2, para el PWM, y el Timer0, para generar la base de tiempo. Una vez que se genera la señal con la dada frecuencia medida, el programa pasa al estado 4.


Inicializar_ADC()
Inicializar_UART()
Inicializar_INT0()
Medicion()
[estado 0]
Deshabilitar_ADC()
Imprimir_datos()
[estado 1]
Standby
[estado 2]
Inicializar_PWM()
Inicializar_Timer0()
[estado 3]
Standby
[estado 4]
Deshabilitar_PWM()
Deshabilitar_Timer0()
Habilitar_ADC()
[estado 5]

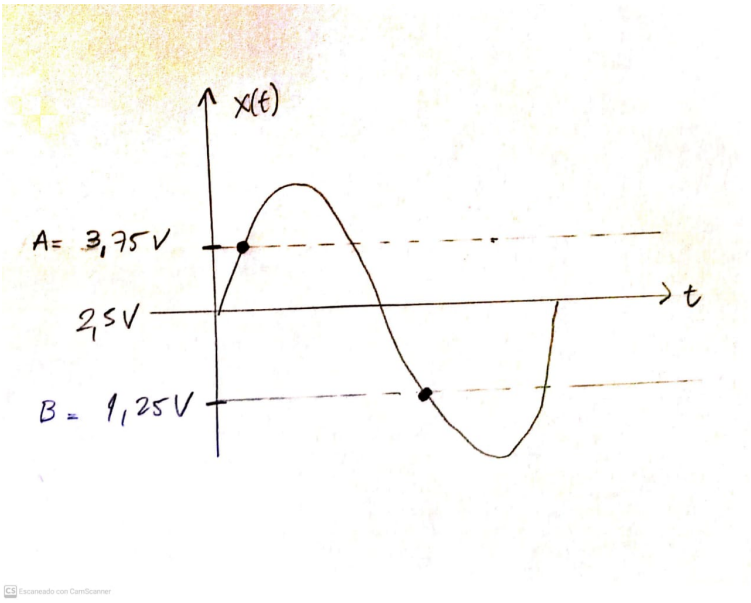


Figura 2: Señal de entrada

Este estado 4 también es un standby, no se realiza ninguna acción hasta que el pulsador sea nuevamente presionado. Si el pulsador es presionado, el programa pasa al estado 5 y luego al estado 0, para volver a comenzar el programa, desde la medición.
Es decir, el programa estando en los estados 2 y 4 (standby) podrán pasar al siguiente estado siempre que el pulsador sea presionado. En cambio, los otros estados pasarán al siguiente de forma inmediata.

2.2. Medición de frecuencia

Para medir la frecuencia de la señal de entrada se utilizó el conversor ADC del microcontrolador. Como se considera que la señal de entrada es de hasta 200Hz, el ADC podrá realizar una cantidad suficiente de conversiones dentro de un periodo mayor o igual a $\frac{1}{200Hz} = 5ms$.
Para obtener la frecuencia de la señal se cuentan la cantidad de máximos y mínimos de la señal, sabiendo que entre dos máximos (considerando que la señal es periódica) existe un periodo de la señal. Sabiendo la cantidad de conversiones que realiza el ADC desde el primer máximo hasta el próximo y el tiempo que tarda cada conversión podemos determinar la frecuencia de la señal.
Se habilitó la interrupción de conversión del ADC, de forma que cada vez que se completa una conversión el código salte a dicha subrutina. Aquí es donde se realiza la medición de la frecuencia.
Se evitó contar los cruces por cero de la señal (cruces por $2{,}5V$ en el caso en que la señal vaya de $0 - 5V$) debido al ruido que se pueda presentar sobre ella. En una señal limpia con seguridad sabremos que la señal cruzará por cero 2 veces en un periodo completo (si consideramos que la señal empieza desde un máximo o un mínimo), sin embargo si la señal es ruidosa puede cruzar más de 2 veces en un periodo generando que la medición sea errónea.

2.2.1. Ruido y medición de los máximos y mínimos

Para poder contar los máximos y mínimos se analizan las veces que la señal pasa por encima de un dado umbral y por debajo de un dado umbral.
Consideramos que la señal de entrada tendrá amplitudes entre 0 y $5V$, cuyo valor medio es de $2{,}5V$, esto debido a los valores de tensión que puede aceptar el ADC del microcontrolador como se muestra en la figura 2. Se definieron dos umbrales para poder diferenciar claramente un máximo de un mínimo y evitar mediciones erróneas.

Sin embargo, aún podemos teneer errores al medir los picos de la señal como se muestra en la figura 3 donde podemos llegar a medir dos máximos consecutivos o dos mínimos consecutivos erróneamente debido al ruido.

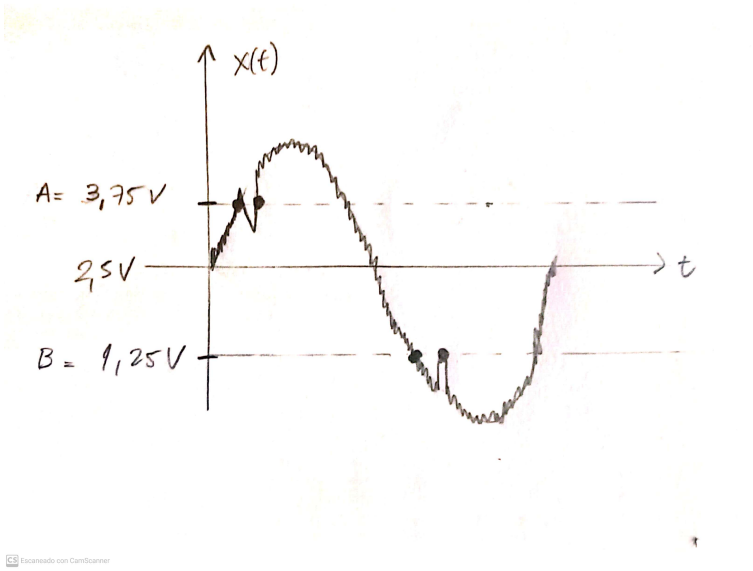


Figura 3: Señal de entrada con ruido

Para evitar esto, se decidió que una vez que se midió un máximo, el siguiente pico que se mida sí o sí debe ser un mínimo, lo mismo al revés, una vez que se midió un mínimo el siguiente pico sea un máximo, esto para evitar medir dos máximos o dos mínimos seguidos. Para lograr esto, se definió una variable 'valor', el cual se va incrementado cada vez que pase por un máximo o un mínimo.
Para que se cuente un máximo, se deben satisfacer dos condiciones: que la señal sea mayor al umbral A y que 'valor' sea un número par. De forma que si la señal pasó por encima del umbral A la primera vez como se muestra en la figura 3 y si 'valor' es par entonces sí se contará ese máximo. Luego del conteo 'valor' pasa a incrementarse, tomando un cifra impar.
Sin embargo, si la señal vuelve a pasar por un máximo este no se contará ya que 'valor' es impar. Esto evita entonces contar máximos consecutivos erróneamente.
Para el caso de los mínimos se contarán si la señal se encuentra por debajo del umbral B y 'valor' es impar.

2.2.2. Cálculo de la frecuencia

Primero es necesario determinar el periodo de cada conversión del ADC. Como se utilizó un prescaler de 128, el clock del ADC posee una frecuencia de $125kHz$. Sabiendo además que cada conversión del ADC tarda 13 ciclos de reloj, tenemos entonces que:

$$T_{ADC} = 13 * \frac{1}{125kHz} = 104\mu s \tag{1}$$

La primera conversión tarda 25 ciclos de reloj, sin embargo esta no se tiene en cuenta para el cálculo.
Cada vez que se completa una conversión, el código salta a una subrutina. Aquí se verifica si el valor de la señal pasa por encima del umbral de $3{,}75V$ (se considera un máximo) o si pasa por debajo del umbral de $1{,}25V$ (se considera un mínimo). Para obtener un mejor resultado, se consideró medir 4 periodos de la señal de entrada, es decir, medir hasta 5 máximos de la señal. Al mismo tiempo una variable 'conteo' cuenta la cantidad total de conversiones realizadas en esos 4 periodos.
Una vez que se completaron 4 periodos el ADC se deshabilita junto con la interrupción. Por tanto, el tiempo transcurrido en estos 4 periodos será:

$$4T_{senal} = conteo \cdot T_{ADC} \tag{2}$$

Finalmente, la frecuencia de la señal será entonces:

$$f_{senal} = \frac{4}{conteo \cdot T_{ADC}} = \frac{4 \cdot 10^6}{conteo \cdot 104} \tag{3}$$

2.3. UART

Para mostrar por pantalla el valor de la frecuencia medida se utilizó el módulo UART del microcontrolador de forma de poder transmitir los datos por el puerto serie. La trama se definió asíncrona, con paridad y 8 bits de largo. Se definieron funciones en el archivo 'UART.c' de forma de poder enviar tanto un caracter como una cadena de caracteres.
Luego con la función *Imprimir_medicion(uint32_t conteo)* se imprimen las frecuencias medidas a través del UART.

2.4. Síntesis de frecuencia

Para sintetizar la señal de salida se utilizó como criterio que en un dado periodo al menos se logren 100 puntos de la señal. Como el microcontrolador no posee un DAC, se utilizó el PWM del Timer2 y un filtro externo para establecer los valores analógicos en cada punto de la señal. Para variar la frecuencia de la señal de salida lo que se fue variando fue el tiempo entre cada punto sintetizado.
En la figura 4 se muestra como debería ser la señal sintetizada después del filtro.

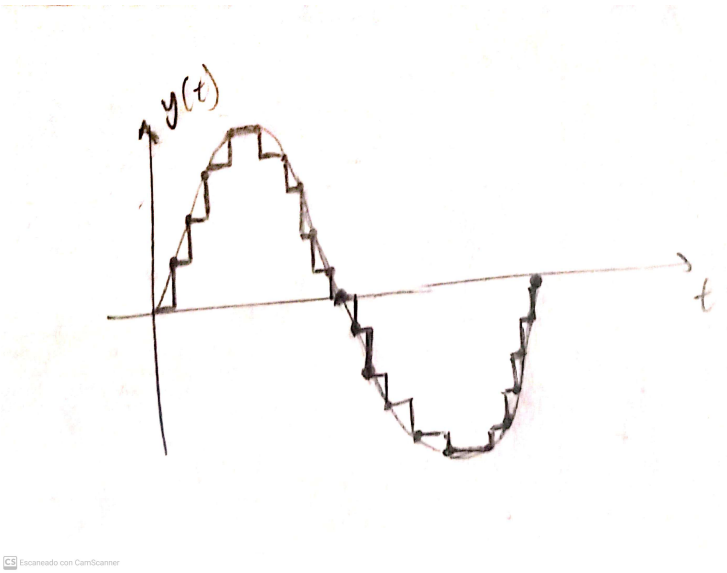


Figura 4: Señal de salida

2.4.1. Base de tiempo

Para variar la frecuencia de la señal de salida lo que se fue variando fue el tiempo entre cada punto sintetizado, para ello se utilizó la interrupción del Timer0 como base de tiempo el cual se modificaba según la frecuencia a sintentizar.
Como criterio se consideró que dentro de un periodo de la señal se reproduzcan 100 puntos. Es decir:

$$T_{senal} = 100 \cdot T_{base\,de\,tiempo} \tag{4}$$

Esto es equivalente a decir que la frecuencia de la base de tiempo debe ser 100 veces más rápida que la frecuencia de la señal. Como la frecuencia de la señal de entrada tiene un máximo de $200Hz$, la frecuencia máxima de la base de tiempo es de $20kHz$.
El Timer0 se interrumpe cada vez que el valor contador TCNT0 llega al valor tope. Este valor tope es el

registro OCR0A.
Este contador se irá incrementando cada $T_{contador} = \frac{1}{f_{clock}/prescaler}$ donde $f_{clock} = 16MHz$ y $prescaler = 256$ para nuestro caso. Por tanto el periodo de la base de tiempo queda definida como:

$$T_{base\,de\,tiempo} = T_{contador} \cdot OCR0A \tag{5}$$

Por tanto variando OCR0A variamos el tiempo con el que se ejecuta la interrupción. Si por ejemplo, la base de tiempo debe ser grande porque es necesario sintetizar una señal lenta, OCR0A debe ser un número grande para que TCNT0 cuente por más tiempo.

2.4.2. Variación del PWM

Para variar el valor analógico de la señal en cada punto lo que se fue variando fue el ciclo de trabajo del PWM. Por tanto fue necesario crear un vector de los valores de tensión del seno en un periodo completo.
Se utilizó el Timer2 de 8 bits para generar el PWM. Por tanto los valores almacenados de tensión debieron ser convertidos a valores digitales para poder variar el ciclo de trabajo del PWM.

$$Duty\,cicle_{PWM} = \frac{V \cdot 255}{5V} \tag{6}$$

Donde V es el valor de tensión almacenado en un vector.
Para evitar tener que representar los valores de tensión en su forma decimal y evitar tener que usar el formato float, se decidió multiplicar los valores de tensión por 100. De esta forma una tensión almacenada de $2{,}5V$ queda representada en el vector como 250 que luego se vuelve a dividir por 100 para poder realizar la conversión usando la ecuacion de arriba.
Considerando que la frecuencia máxima a sintetizar es de $200Hz$ y por tanto la frecuencia máxima de la interrupción del Timer0 (base de tiempo) es de $20kHz$, la frecuencia de la señal PWM debe ser mayor que $20kHz$. Esto para garantizar que el valor medio del ciclo de trabajo del PWM se haga efectivo. Por tanto se utiliza como criterio que al menos la frecuencia del PWM sea 3 veces mayor que la frecuencia de la base de tiempo. Por tanto:

$$f_{PWM} > 3f_{base\,de\,tiempo} \tag{7}$$

De esta forma se definió una frecuencia del PWM de $62{,}5kHz$. Se utilizó el modo Fast PWM.

3. Resultados

Para probar el código y ver los resultados se utilizó el programa Proteus.

3.1. En la medición

Todas las formas de ondas de entrada consideradas poseen amplitud de $0 - 5V$.
Como no se obtuvieron buenos resultados en la medición, se utilizó un factor de corrección para el periodo del ADC, es decir, en lugar de considerar para el cálculo el valor de $104\mu s$ (el cual corresponde a una frecuencia del ADC de $125kHz$ y prescaler de 128), se utilizó un valor de $100\mu s$. Los resultados luego de esta modificación se muestran en la tabla siguiente.

	Onda sinusoidal	Onda triangular	Onda cuadrada
$f = 1Hz$	$0{,}9Hz(-10\,\%)$	$0{,}9Hz(-10\,\%)$	$0{,}9Hz(-10\,\%)$
$f = 10Hz$	$10{,}5Hz(+5\,\%)$	$10{,}4Hz(+4\,\%)$	$10{,}8Hz(+8\,\%)$
$f = 100Hz$	$102{,}5Hz(+2{,}5\,\%)$	$101{,}5Hz(+1{,}5\,\%)$	$104{,}7Hz(+4{,}7\,\%)$
$f = 200Hz$	$183{,}4Hz(-8{,}3\,\%)$	$181{,}8Hz(-9{,}1\,\%)$	$186{,}9Hz(-6{,}55\,\%)$
$f = 1kHz$	$975{,}6Hz(-2{,}44\,\%)$	$952{,}3Hz(-4{,}77\,\%)$	$975{,}6Hz(-2{,}44\,\%)$
$f = 5kHz$	$5000{,}0Hz(0\,\%)$	$5000{,}0Hz(0\,\%)$	$5000{,}0Hz(0\,\%)$

Para las frecuencias de interés (0-200Hz) se observan que se obtienen resultados aceptables incluso buenos para frecuencias entre $10Hz - 100Hz$. Vemos que para frecuencias muy bajas como $1Hz$ y frecuencias en $200Hz$ los resultados poseen poca precisión.
Luego se comprobó el funcionamiento del programa para frecuencias del orden de los kHz, el cual como se ve en la tabla se obtienen buenos resultados.

3.2. En la síntesis

Los resultados obtenidos de la señal sintetizada dependen de la medición anterior, por tanto el error debido a la medición se arrastra luego para la síntesis. A pesar de todo ello, se logran algunos resultados aceptables.

	Onda sinusoidal	Onda triangular	Onda cuadrada
$f = 1Hz$	$2{,}43Hz(+143\,\%)$	$2{,}46Hz(+146\,\%)$	$2{,}5Hz(+150\,\%)$
$f = 10Hz$	$10{,}1Hz(+1{,}01\,\%)$	$10{,}63Hz(+6{,}38\,\%)$	$9{,}90Hz(-1{,}00\,\%)$
$f = 100Hz$	$89{,}28Hz(-10{,}71\,\%)$	$88{,}49Hz(-11{,}5\,\%)$	$89{,}28Hz(-10{,}71\,\%)$
$f = 200Hz$	$156{,}25Hz(-21{,}8\,\%)$	$156{,}25Hz(-21{,}8\,\%)$	$156{,}25Hz(-21{,}8\,\%)$

Como la señal que sale del microcontrolador es una señal PWM cuya fundamental es de $62{,}5kHz$ se implementó un filtro pasabajos activo con un operacional cuya frecuencia de corte es de $500Hz$. Esto para atenuar la fundamental y los armónicos del PWM y dejar pasar el valor medio del PWM y las frecuencias de interés (frecuencias hasta $200Hz$).
Las figuras 5 y , muestran las formas de ondas visualizadas en el osciloscopio digital dentro del entorno de Proteus. Para determinar la frecuencia de la sinusoide sintetizada se utilizaron cursores.

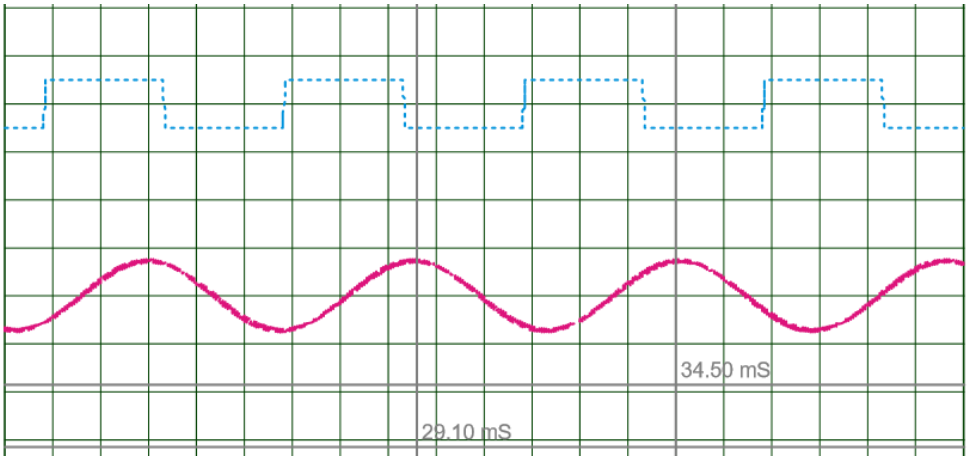


Figura 5: Azul: Señal de entrada cuadrada 200Hz - Rosa: Señal sinusoidal sintetizada

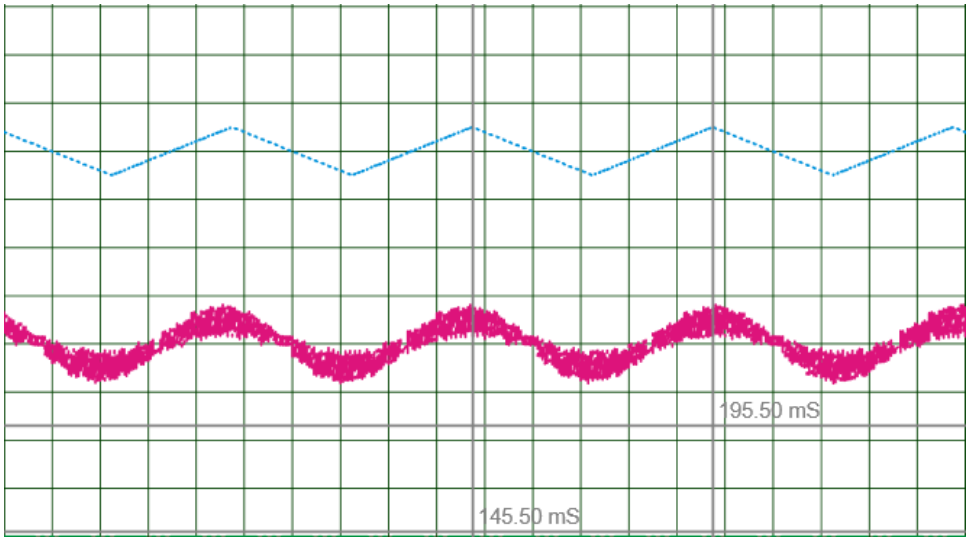


Figura 6: Azul: Señal de entrada triangular 20Hz - Rosa: Señal sinusoidal sintetizada

4. Conclusiones

A partir de los resultados obtenidos vemos que la medición de frecuencia no es del todo satisfactoria ya que se tuvo que aplicar una corrección para obtener mejores resultados. Sin embargo es un método aceptable y que considera el posible ruido que pueda interferir sobre la señal.
En cuanto a la síntesis, el método para la sintetización está encaminada. Evidentemente, en general,no se obtuvieron resultados aceptables pero se podrían realizar mejoras para conseguir mayor precisión. Quizás utilizando el Timer1 de 16bits se obtengan tiempos espaciados más precisos y la utilización de más de 100 puntos generados mejoren el programa implementado.
A pesar de lo anterior, se consiguió que el programa funcionara incorporando al mismo tiempo conocimiento sobre los módulos utilizados y la forma de organizar el proyecto.

5. Referencias

-ATmega328P Datasheet.
-PWM Sine Wave Generation.