

# Muestreo de Señal PWM

Oscar Ivan Moreno Gutierrez

May 10, 2024

## 1 Descripción

Este proyecto se centra en realizar medidas sobre dos señales que se obtienen al filtrar una señal PWM generada con el dsPIC. Las medidas que se realizan incluyen el valor máximo, el valor medio y el valor eficaz de las señales. La información obtenida de ambas señales se muestra en una pantalla LCD. Además, se crea un filtro digital utilizando la herramienta de MATLAB para procesar las señales.

## 2 Archivos y Directorios

- `main.c`: Este es el archivo principal de mi proyecto.
- `pintar_lcd.c`: Este archivo se utiliza para funciones relacionadas con la visualización LCD.
- `funciones.c`: Este archivo contiene funciones básicas y necesarias para el proyecto, como la aplicación de filtros.

## 3 Procedimiento

Empezamos con la creación de las funciones básicas del dsPIC utilizando un SSCP para una salida PWM, Cumpliendo el con el esquema proporcionado (Figura 1).

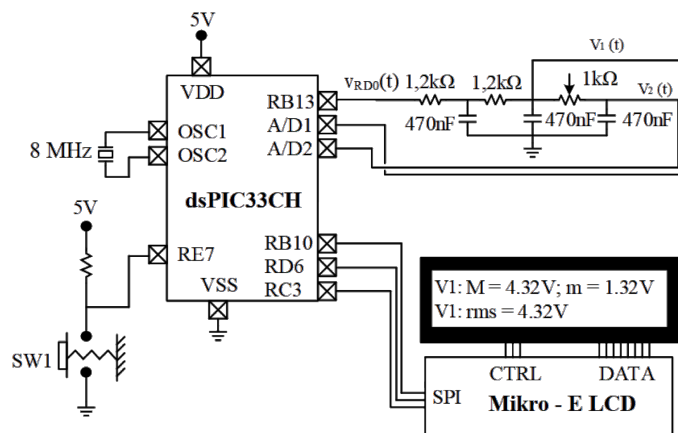


Figure 1: Esquema del proyecto

La salida PWM sale por el RB13 y se conecta con un filtro proporcionado por el profesor. Donde se obtienen las señales senoidal y triangular. Además de otros cambios:

- El pin RE9 (Switch 3) sale de la pantalla de inicio.
- El pin RE8 (Switch 2) cambia el tipo de filtro aplicado.
- El pin RE7 (Switch 1) cambia el tipo de señal visualizada.
- El pin RA0 (potenciómetro) se utiliza para manipular el ciclo de trabajo de la señal PWM.
- El pin RC1 fue asignado para ADC1, que recibe la señal senoidal.
- El pin RC3 fue asignado para ADC2, que recibe la señal triangular.

Una mejora sería la implementación de manipular el ciclo de trabajo con un módulo ADC. Utilizando un potenciómetro para manipular el ciclo de trabajo de la señal PWM.

### 3.1 Señales

Posteriormente, realizo las medidas de las señales PWM, visualizado en un osciloscopio. Como se muestra en la Figura 2.

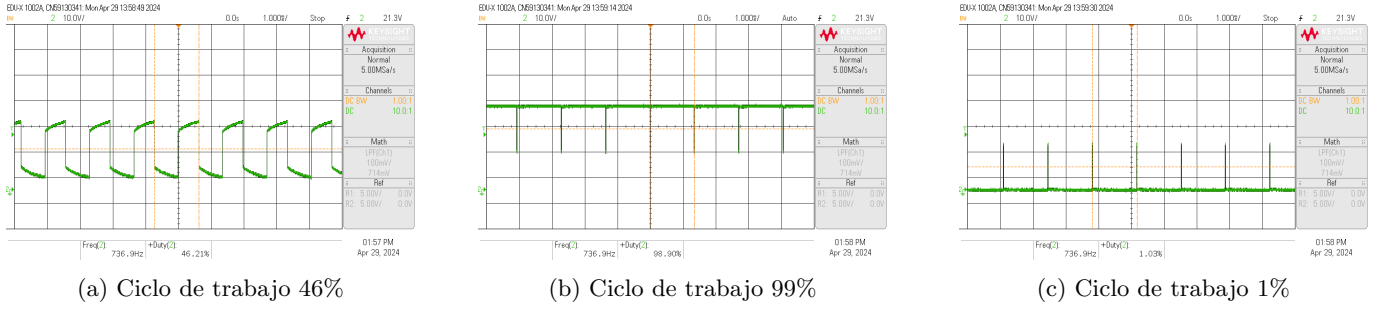


Figure 2: señal PWM en osciloscopio

Se diseñó que el límite de la señal PWM sea de 1% a 99% en su ciclo de trabajo. Se observa que la señal PWM se comporta de manera correcta en el osciloscopio.

La señal senoidal y triangular se obtiene al filtrar la señal PWM con un filtro analógico. El filtro suaviza la forma de onda cuadrada de la señal PWM, eliminando las componentes de alta frecuencia y conservando las componentes de baja frecuencia, lo que resulta en una forma de onda más suave. La visualización de la señal senoidal se muestra en la Figura 3a. La visualización de la señal triangular se muestra en la Figura 3b.

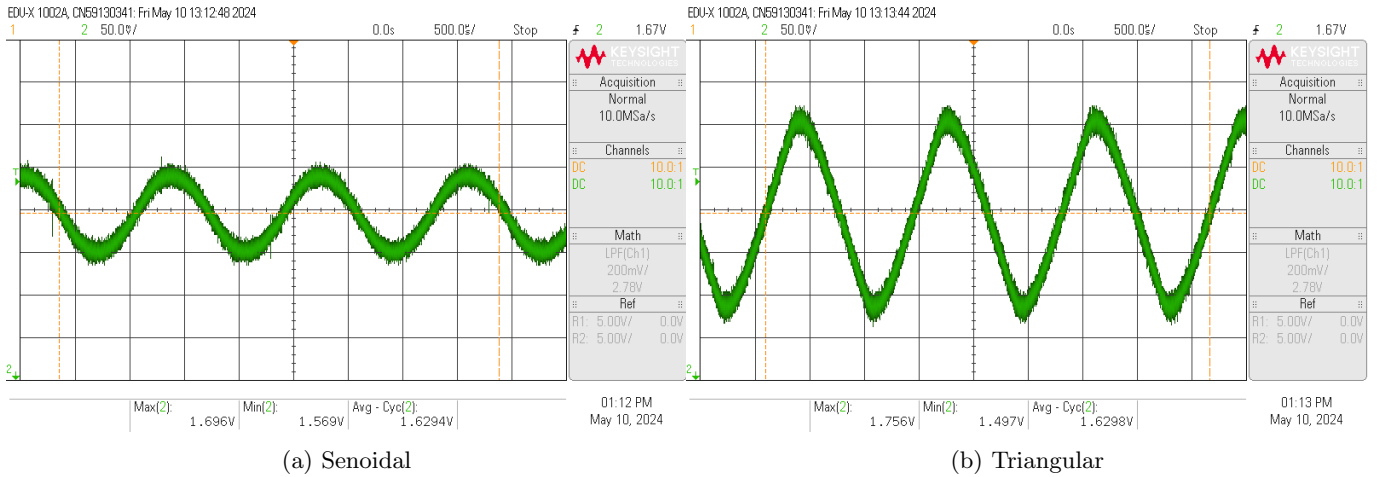


Figure 3: señal senoidal y triangular en osciloscopio

Se notó, aumentando el ciclo de trabajo aumenta el voltaje máximo de la señal, incluyendo su mínimo y máximo respectivamente. Cabe recalcar que el voltaje máximo y mínimo de las señales no exceden de los 0.8 volts.

### 3.2 Filtros

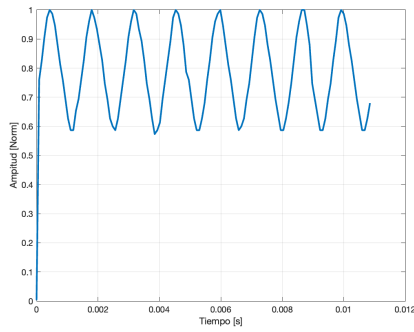
Se implementó un filtro digital a las señales senoidal y triangular. Se utilizó la herramienta de MATLAB Filter Designer para diseñar el filtro. Primero conociendo la escogiendo una frecuencia adecuada de muestreo por la siguiente fórmula:

$$f_s = 16 * 737hz = 11792Hz \quad (1)$$

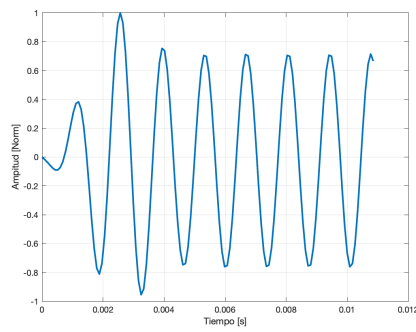
Donde 737 Hz es la frecuencia de nuestras señales. Se obtuvo el siguiente filtro FIR:

- **Filtro FIR:** Window
- **Frecuencia de muestreo:** 11792 Hz
- **Frecuencia de corte 1:** 710 Hz
- **Frecuencia de corte 2:** 750 Hz
- **Orden del filtro:** 50
- **Tipo de filtro:** Bandpass

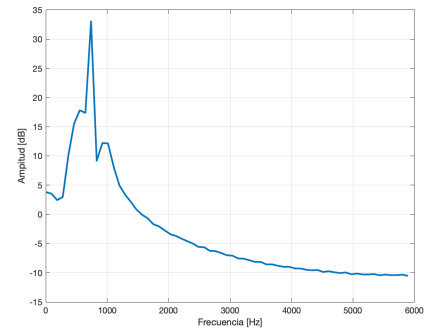
Aplicacion para el Señal senoidal:



(a) Señal de entrada con respecto al tiempo



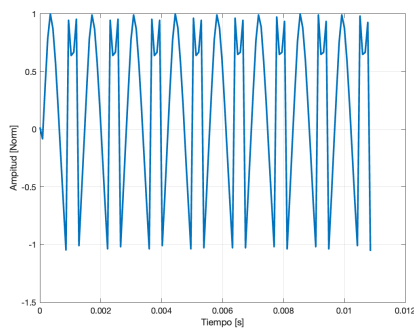
(b) Señal filtrada con respecto al tiempo



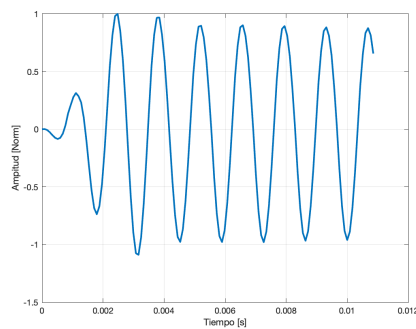
(c) Espectro de la señal filtrada

Figure 4: Filtro aplicado a la señal senoidal

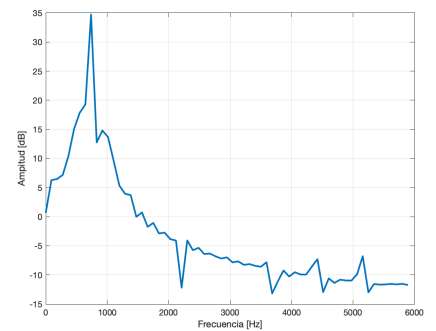
Aplicacion para la señal triangular:



(a) Señal de entrada con respecto al tiempo



(b) Señal filtrada con respecto al tiempo



(c) Espectro de la señal filtrada

Figure 5: Filtro aplicado a la señal triangular

Aplicando los filtros los voltajes toman valores de corriente alterna, donde los voltajes salen simétricos entre sí.

## 4 Código

Unos códigos importantes para el proyecto son los siguientes: Para calcular el RMS:

```
1  -----funciones.c-----
2  float calculate_rms(int16_t* samples, uint8_t num_samples) {
3  float sum_squared = 0.0;
4  float mean_squared=0.0;
5  float rms=0.0;
6  for (uint8_t i = 0; i < num_samples; i++)
7      sum_squared += (float)samples[i] * (float)samples[i];
8  mean_squared = sum_squared / num_samples;
9  rms = sqrt(mean_squared);
10 return rms;}
```

Para aplicar el filtro FIR:

```
1  -----main.c-----
2  // multiplicamos por 0x0100 para convertir a fractional
3  buffer_fractional[buffer_move] = buffer[buffer_move] * 0x0100;
4  aplicarFiltroFIR(buffer_fractional,buffer_filter,NUM_SAMPLES);
5  // dividimos por 0x0100 para convertir a integer
6  buffer[buffer_move] = buffer_filter[0] / 0x0100;
7  -----funciones.c-----
8  void aplicarFiltroFIR (fractional* buffer,fractional* output, uint8_t num_samples){
9  extern FIRStruct FIR11792Filter;
10 FIRDelayInit (&FIR11792Filter);
11 FIR(num_samples, &output[0], &buffer[0], &FIR11792Filter);
12 return;}
```