

## TPN°1- Último Ejercicio

**Autor:** Pablo D. Folino

**Link del repositorio:** [https://github.com/PabloFolino/MSE\\_PSF\\_TP1.git](https://github.com/PabloFolino/MSE_PSF_TP1.git)

Enunciado:

Genere con un tono de LA-440. Digitalice con 10, 8, 4 y 2 bits con el ADC, envíe los datos a la PC y grafique: 1) Señal original con su máximo, mínimo y RMS 2) Señal adquirida con su máximo, mínimo y RMS. Hay diferencias? a que se deben?. Suba un pdf con los códigos, los gráficos y algunas fotos/video del hardware utilizado.

Para generar la señal desde la PC se utiliza el siguiente programa ([visualize folino.py](#)), el cual mediante un menú se puede acceder a enviar distintos tipos de señales (senoidal, cuadrada, triangular, suma de dos senoidales, una senoidal que barre en frecuencia) y se tiene una opción para configurar las amplitud, las frecuencias, la fase entre las frecuencias, etc.

El menú principal es:

```
Programas de la transformada Discreta de Fourier
Elija una opción:

[1] Señal senoidal (fs,f,amp,muestras,fase)
[2] Señal cuadrada (fs,f,amp,muestras)
[3] Señal triangular (fs,f,amp,muestras)
[4] Suma de frecuencias senoidales  $0.7*f+0.3*B$ 
[5] Barrido de frecuencias senoidales de f a B

[6] TBD
[7] TBD
[8] Seteo de frecuencia de la señal de entrada, número de muestras, frecuencia
de sampling
[9] Salir

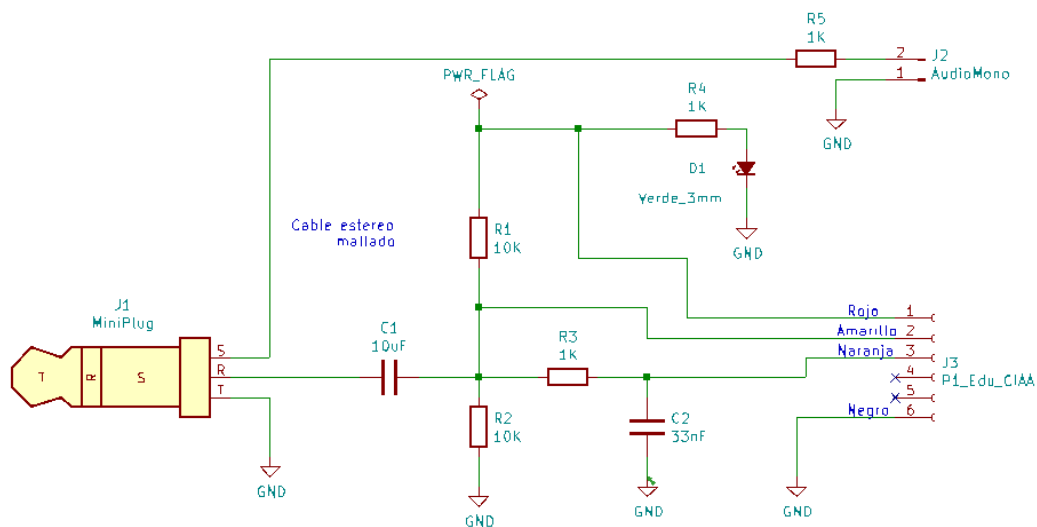
Elija una opción: 
```

Y al acceder a la opción [8] se puede configurar:

```
-----
Los valores actuales son:

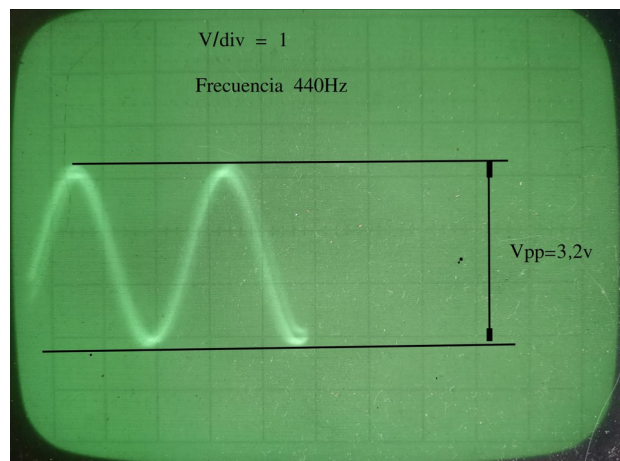
La frecuencia de la señal es=1000
La amplitud de la señal es=1.0
La cantidad de períodos es N=10000
La fase de la señal en radianes es=0*Pi radianes
-----
La frecuencia de fs=44100Hz, y el ts=2.2675736961451248e-05seg
La cantidad de veces que se desea repetir es=1
La frecuencia de B=2500Hz
Desea cambiar los valores S o N[Enter] :□
```

Para la adquisición de la señal se utiliza el siguiente circuito:



En la salida J2 se conecta un osciloscopio, y en J3 se conecta la EDU-CIAA. La señal proviene de la PC desde la entrada J1.

La señal medida en el osciloscopio tiene una frecuencia de 440Z y una tensi{on pico a pico de 3,2v, con lo cual la  $V_p=1,6v$  y la tensión eficaz es de  $1,13v$ .



A continuación se muestra el conexionado de los distintos equipos intervinientes:



El código utilizado en el EDI- CIAA y se encuentra en el directorio **/Programas/TP1/src** del repositorio, se observa que la constante **ADC\_BITS** define la cantidad de bits utilizados para la conversión.

```
#include "arm_math.h"
#include "sapi.h"
```

```
#define SCT_PWM_PIN_LED 2
#define SCT_PWM_LED 2
#define SCT_ADC_PIN_OUT 8
#define SCT_ADC_OUT 1
#define ADC_FRAME_SAMPLES 128
#define ADC_TRIGGER_VALUE 512
#define ADC_SAMPLE_RATE 10000
#define ADC_BITS 10
```



```
struct header_struct {
    char pre[8];
    uint32_t id;
    uint16_t N;
    uint16_t fs;
    uint32_t maxIndex;
    uint32_t minIndex;
    q15_t maxValue;
    q15_t minValue;
    q15_t rms;
    char pos[4];
} __attribute__((packed)) header = {
    "header", 0, ADC_FRAME_SAMPLES, ADC_SAMPLE_RATE, 0, 0, 0, 0, 0, "end*"};

int16_t adc[ADC_FRAME_SAMPLES];
volatile uint16_t index;

void ADC0_IRQHandler(void) {
```

```

static enum {
    ESPERANDO_MENOR,
    ESPERANDO_MAYOR,
    MUESTREANDO,
} estado = ESPERANDO_MENOR;
uint16_t sample;

Chip_ADC_ReadValue(LPC_ADC0, ADC_CH1, &sample);

if (index == 0) {
    if (estado == MUESTREANDO) estado = ESPERANDO_MENOR;
    switch (estado) {
        case ESPERANDO_MENOR:
            if (sample < ADC_TRIGGER_VALUE) estado = ESPERANDO_MAYOR;
            break;
        case ESPERANDO_MAYOR:
            if (sample > ADC_TRIGGER_VALUE) estado = MUESTREANDO;
            break;
        default:
            break;
    }
}
if (estado == MUESTREANDO) {
    adc[index] = (((sample - 512)) >> (10 - ADC_BITS)) << (6 + (10 - ADC_BITS));
    index++;
}
gpioToggle(LED1); // este led blinkea a fs/2
}

void adc_init(void) {
    ADC_CLOCK_SETUP_T adc;

    Chip_ADC_Init(LPC_ADC0, &adc);
    Chip_ADC_EnableChannel(LPC_ADC0, ADC_CH1, ENABLE);

    Chip_ADC_SetStartMode(LPC_ADC0, ADC_START_ON_CTOUT8, ADC_TRIGGERMODE_FALLING);
    NVIC_EnableIRQ(ADC0_IRQn);
}

void sct_init(void) {
    uint16_t duty = Chip_SCTPWM_GetTicksPerCycle(LPC_SCT) / 2;

    Chip_SCTPWM_Init(LPC_SCT);
    Chip_SCTPWM_SetRate(LPC_SCT, ADC_SAMPLE_RATE);

    Chip_SCTPWM_SetOutPin(LPC_SCT, SCT_ADC_OUT, SCT_ADC_PIN_OUT);
    Chip_SCTPWM_SetDutyCycle(LPC_SCT, SCT_ADC_OUT, duty);

    Chip_SCTPWM_Start(LPC_SCT);
}

int main(void) {
    boardConfig();
    uartConfig(UART_USB, 460800);

    adc_init();
    sct_init();

    while (1) {
        index = 0;
        Chip_ADC_Int_SetChannelCmd(LPC_ADC0, ADC_CH1, ENABLE);
        while (index < header.N) {
            __WFI();

```

```

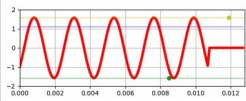
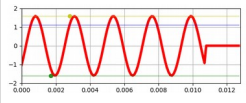
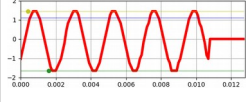
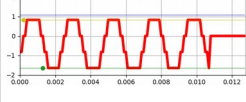
};
Chip_ADC_Int_SetChannelCmd(LPC_ADC0, ADC_CH1, DISABLE);

gpioToggle(LED2);
arm_max_q15 adc, header.N, &header.maxValue, &header.maxIndex);
arm_min_q15 adc, header.N, &header.minValue, &header.minIndex);
arm_rms_q15 adc, header.N, &header.rms);
header.id++;
uartWriteByteArray(UART_USB, (uint8_t *)&header, sizeof(header));
uartWriteByteArray(UART_USB, (uint8_t *)adc, sizeof(adc));
}
}

```

El programa de la PC se encuentra en el archivo **visualize.py** del directorio **/Programas/TP1/linux**.

Cuadro de resultados:

Condiciones — > f=440Hz    Vpp=3,2v    fs=44100Hz    Vrms=1,13v				
Cantidad de bits ADC	V máx.	V mín	V rms	
10	1,6v	-1,6v	1,13v	
8	1,58v	-1,59v	1,12v	
4	1,44v	-1,65v	1,11v	
2	0,825v	-1,65v	1,07v	

Se observa que a menor cantidad de niveles de conversión la señal se va deformando la tensión eficaz va disminuyendo, al igual que los valores máximos y mínimos de tensión.