

PROCESAMIENTO DIGITAL DE SEÑALES

Laboratorio N°1: Muestreo con FRDM-K64

DOCENTES:

Parlanti, Gustavo.

Rossi, Roberto.

Molina, German.

ALUMNOS:

Gomez Augusto Rodrigo, matricula: 39807998

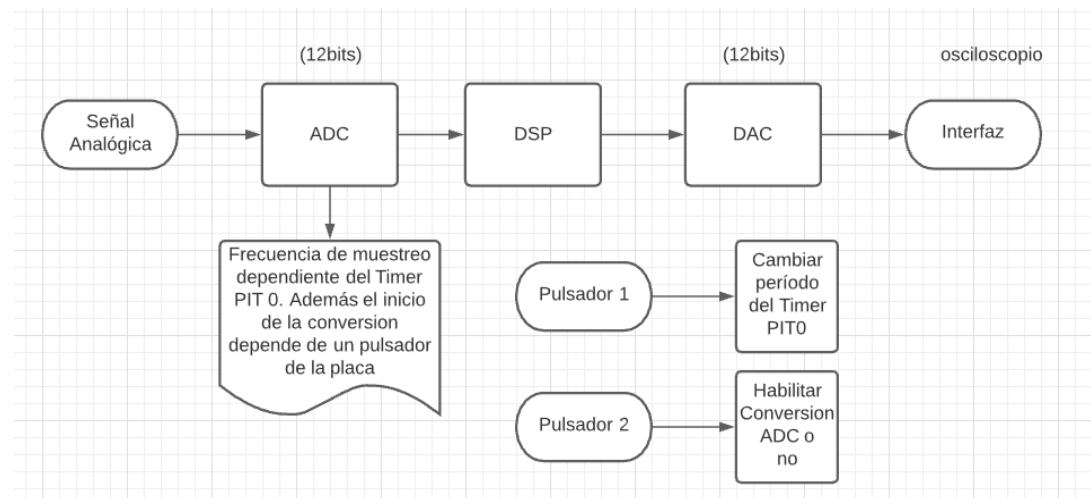
Ferraris, Domingo Jesús, matricula: 36656566

Alaniz Franco, matricula: 37234066

Facultad de Ciencia Exactas, Físicas y Naturales
Procesamiento digital de señales

Diagrama de bloques simplificado de funcionamiento:

El sistema se encarga de muestrear y convertir una señal ingresada para luego redireccionar al módulo DAC de la placa y poder observar el efecto de muestrear la señal a distintas velocidades.



El sistema consta del ADC, trabajando en modo 12 bits SE, al que le ingresamos la señal a muestrear, que en nuestro caso generamos con la placa de sonido de la PC. Además consta con un pulsador selector de velocidades e indicador de las mismas por medio de los colores de un led RGB, y de un pulsador con función de start/stop of conversión. Finalmente por medio del DAC de 12 bits reconstruimos la señal analogica.

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

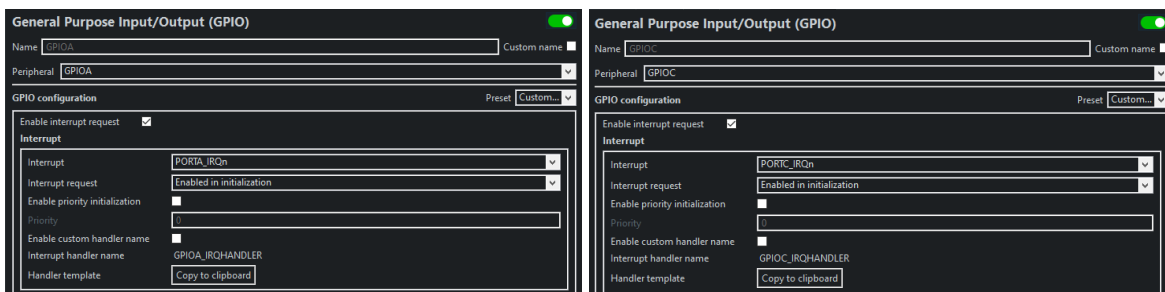
Configuración de periféricos

GPIO

Se utilizaron los 2 pulsadores SW2 y SW3 (GPIOC y GPIOA) de la placa en modo de interrupción por flanco descendente, con la opción de filtro de entrada habilitada para evitar posibles ruidos entre pulsaciones. Además se habilitó el led RGB para visualizar cada frecuencia de muestreo con un color distinto.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive streng
78	GPIOC	GPIO, 6	PTC6	U8[11]/SW2	SW2	Input	n/a	Interrupt on falling edge	Fast	Disabled	Low
38	GPIOA	GPIO, 4	PTA4	SW3	SW3	Input	n/a	Interrupt on falling edge	Fast	Disabled	High



General Purpose Input/Output (GPIO)

Name: Custom name:

Peripheral:

GPIO configuration Preset:

Enable interrupt request ☒

Interrupt:

Interrupt request:

Enable priority initialization ☐

Priority:

Enable custom handler name ☐

Interrupt handler name:

Handler template:

General Purpose Input/Output (GPIO)

Name: Custom name:

Peripheral:

GPIO configuration Preset:

Enable interrupt request ☒

Interrupt:

Interrupt request:

Enable priority initialization ☐

Priority:

Enable custom handler name ☐

Interrupt handler name:

Handler template:

GPIOA

Cuando SW3 interrumpe el handler de este puerto cambia el periodo de interrupción del PIT con valores almacenados en un buffer de estructura cíclica accedido por punteros de memoria.

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

GPIOC

La interrupción de SW2 llama al handler de este puerto y habilita el PIT que será la base de tiempos para el muestreo.

PIT

Utilizamos el ch0 del PIT desactivado en principio que se activa cuando interrumpa SW2. En este momento se encargará de generar interrupciones periódicas para que el ADC tome muestras de acuerdo a la velocidad de muestreo previamente seleccionada.

Periodic Interrupt Timer (PIT)

Name Custom name

Mode Peripheral

General PIT configuration Preset

Run PIT in debug mode ☐

Clock setting

Clock source

Clock source frequency

PIT channels + ×

CHANNEL_0	
Channel ID	<input type="text" value="CHANNEL_0"/>
Channel number	<input type="text" value="Channel 0"/>
Chained	<input type="checkbox"/>
Channel period/frequency	<input type="text" value="500ms"/>
Resulting period	500 ms (30000000 ticks)
Start channel	<input type="checkbox"/>
Interrupt	<input checked="" type="checkbox"/>
Interrupt	<input type="text" value="PIT0_IRQn"/>
Interrupt request	<input type="text" value="Enabled in initialization"/>
Enable priority initialization	<input type="checkbox"/>
Priority	<input type="text" value="0"/>

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

CHANNEL_0	
Channel ID	CHANNEL_0
Channel number	Channel 0
Chained	<input type="checkbox"/>
Channel period/frequency	500ms
Resulting period	500 ms (30000000 ticks)
Start channel	<input type="checkbox"/>
Interrupt	<input checked="" type="checkbox"/>
Interrupt	PIT0_IRQn
Interrupt request	Enabled in initialization
Enable priority initialization	<input type="checkbox"/>
Priority	0
Enable custom handler name	<input type="checkbox"/>
Interrupt handler name	PIT_CHANNEL_0_IRQHANDLER
Handler template	Copy to clipboard

Se utilizó el clock de la placa de 60 MHz con interrupción al desbordar, en un principio se probó con el clock de 40 MHz pero los retardos obtenidos no eran exactos. Inicialmente el PIT está deshabilitado Iniciado en 500ms, pero al iniciar el sistema se configura el periodo a 125us para la menor frecuencia de muestreo de 8KHz.

ADC

Utilizamos el ch18 en modo SE a una resolución de 12 bits y la referencia de tensión interna, la conversión está desactivada en el inicio en espera de que se active el PIT.

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

Analog-to-Digital Converter (ADC)

Name: Custom name: ☐

Mode: Peripheral:

ADC16 configuration Preset:

ADC16 basic configuration

Reference voltage source	<input type="text" value="VrefH / VrefL"/>
Input clock source	<input type="text" value="Asynchronous clock"/>
Asynchronous clock output	<input checked="" type="checkbox"/>
Divide input clock source	<input type="text" value="The module input clock divided by 4"/>
Sample resolution mode	<input type="text" value="Single End 12-bit"/>
Long sample mode	<input type="text" value="Disable the long sample feature"/>
High-speed mode	<input type="checkbox"/>
Low power mode	<input type="checkbox"/>
Continuous conversion mode	<input type="checkbox"/>

Channel mux mode:

ADC16 Hardware compare configuration

Hardware compare mode	<input type="checkbox"/>
Perform auto calibration	<input type="checkbox"/>
Conversion offset	<input type="text" value="2047"/>
Use hardware trigger	<input type="checkbox"/>
Hardware average mode	<input type="text" value="Disable the hardware average feature"/>
DMA requests	<input type="checkbox"/>

Se utilizó un prescaler de 4 para la señal del clock, aunque en un principio se configuró el divisor en 8 pero notamos que el sistema no funcionaba correctamente para las frecuencias de muestreo más altas. Luego de mediciones llegamos a la conclusión de que esto se debía a que el PIT interrumpía a un periodo menor del que necesitaba el ADC para terminar la conversión, por tanto decidimos aumentar el clock del ADC.

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

Interrupt request ☒

Interrupt

Interrupt

Interrupt request

Enable priority initialization ☐

Priority

Enable custom handler name ☐

Interrupt handler name

Handler template

ADC16 channels configuration

#	0
Differential sample mode	<input type="checkbox"/>
Channel number	SE, 0 » [18]...
Second channel number	N/A
Conversion complete interrupt	<input checked="" type="checkbox"/>
Conversion control group	Group 0
Initialize channel	<input type="checkbox"/>

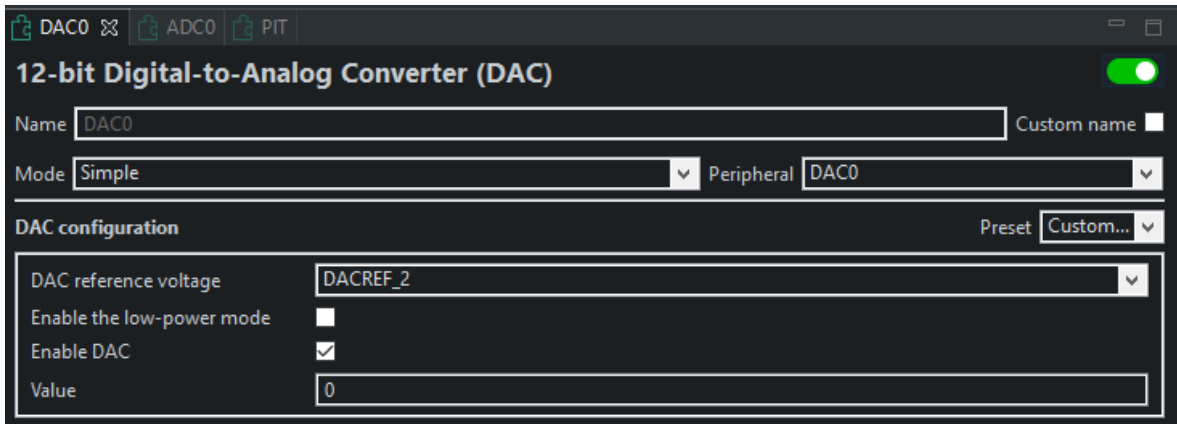
La entrada de señal del ADC es por el conector J2 pin 5 e interrumpirá al finalizar la conversión. Es este momento el handler se encarga de enviar el resultado de la conversión al DAC.

DAC

Por último para la salida de señal se utilizó el módulo DAC de 12 bits con salida en el conector J4 pin 11, y utilizando la referencia de tensión interna.

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales



12-bit Digital-to-Analog Converter (DAC)

Name: Custom name: ☐

Mode: Peripheral:

DAC configuration Preset:

DAC reference voltage:

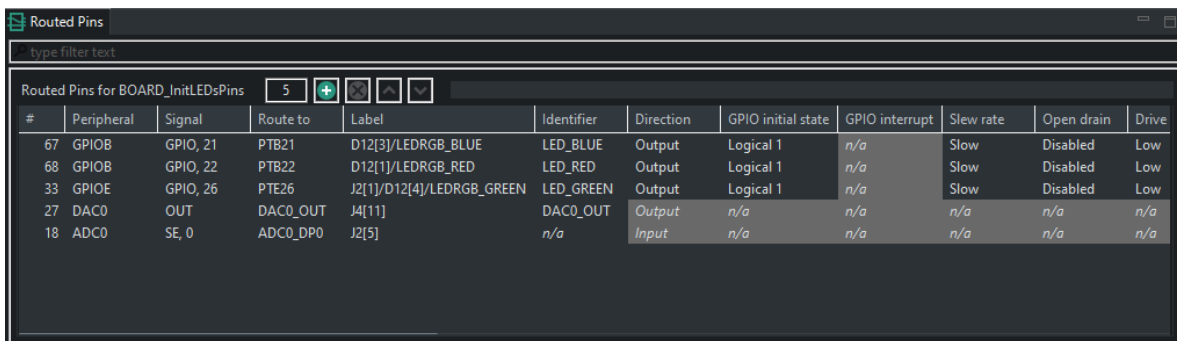
Enable the low-power mode: ☐

Enable DAC: ☒

Value:

Configuración de los pines

Resumen de la configuración de los pines tanto digitales como de entrada/salida analógica.



#	Peripheral	Signal	Route to	Label	Identifier	Direction	GPIO initial state	GPIO interrupt	Slew rate	Open drain	Drive
67	GPIOB	GPIO, 21	PTB21	D12[3]/LEDRGB_BLUE	LED_BLUE	Output	Logical 1	n/a	Slow	Disabled	Low
68	GPIOB	GPIO, 22	PTB22	D12[1]/LEDRGB_RED	LED_RED	Output	Logical 1	n/a	Slow	Disabled	Low
33	GPIOE	GPIO, 26	PTE26	J2[1]/D12[4]/LEDRGB_GREEN	LED_GREEN	Output	Logical 1	n/a	Slow	Disabled	Low
27	DAC0	OUT	DAC0_OUT	J4[11]	DAC0_OUT	Output	n/a	n/a	n/a	n/a	n/a
18	ADC0	SE, 0	ADC0_DP0	J2[5]	n/a	Input	n/a	n/a	n/a	n/a	n/a

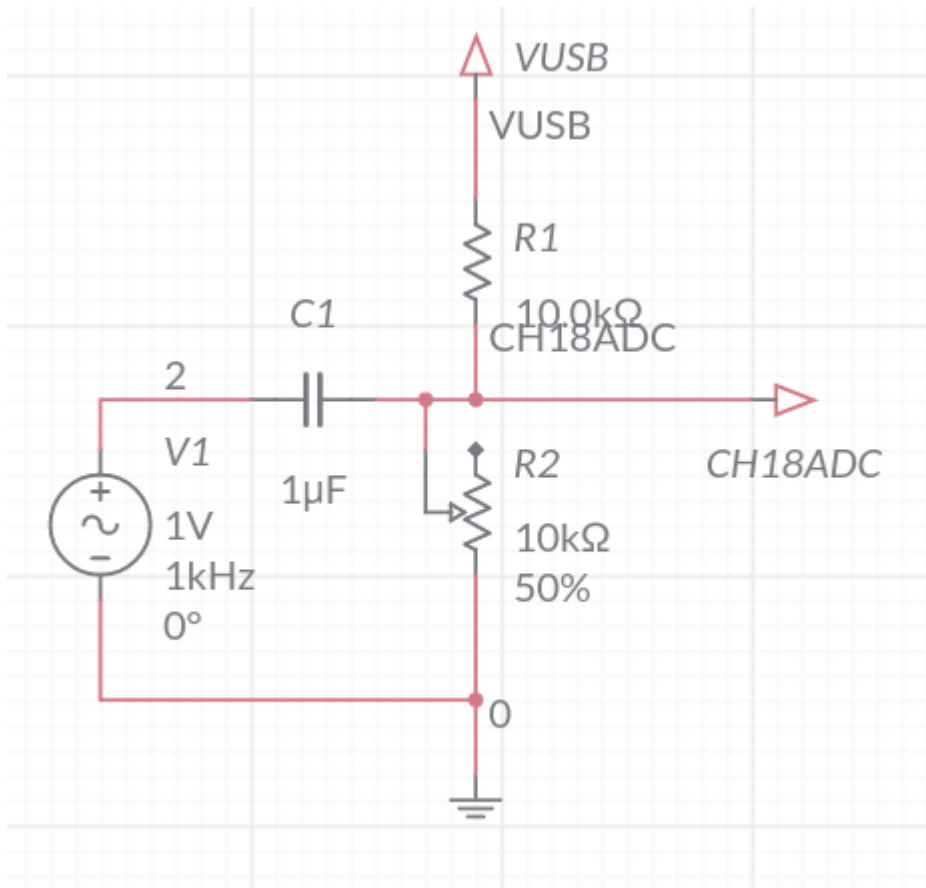
Algunas fotos tomadas del trabajo

Para poder complementar la experiencia y aprovechando que un integrante del grupo posee osciloscopio, se procedió a probar físicamente la performance del programa, para ello la señal analógica de entrada es la salida de la placa de sonido de la PC, la señal es de tipo senoidal, debido a que el conversor trabaja con tensiones positivas, se agregó una tensión de continua a la señal de alterna, mediante un divisor de tensión variable y colocando un capacitor en su punto medio. Básicamente la señal entra por el capacitor y en el punto

Facultad de Ciencia Exactas, Físicas y Naturales

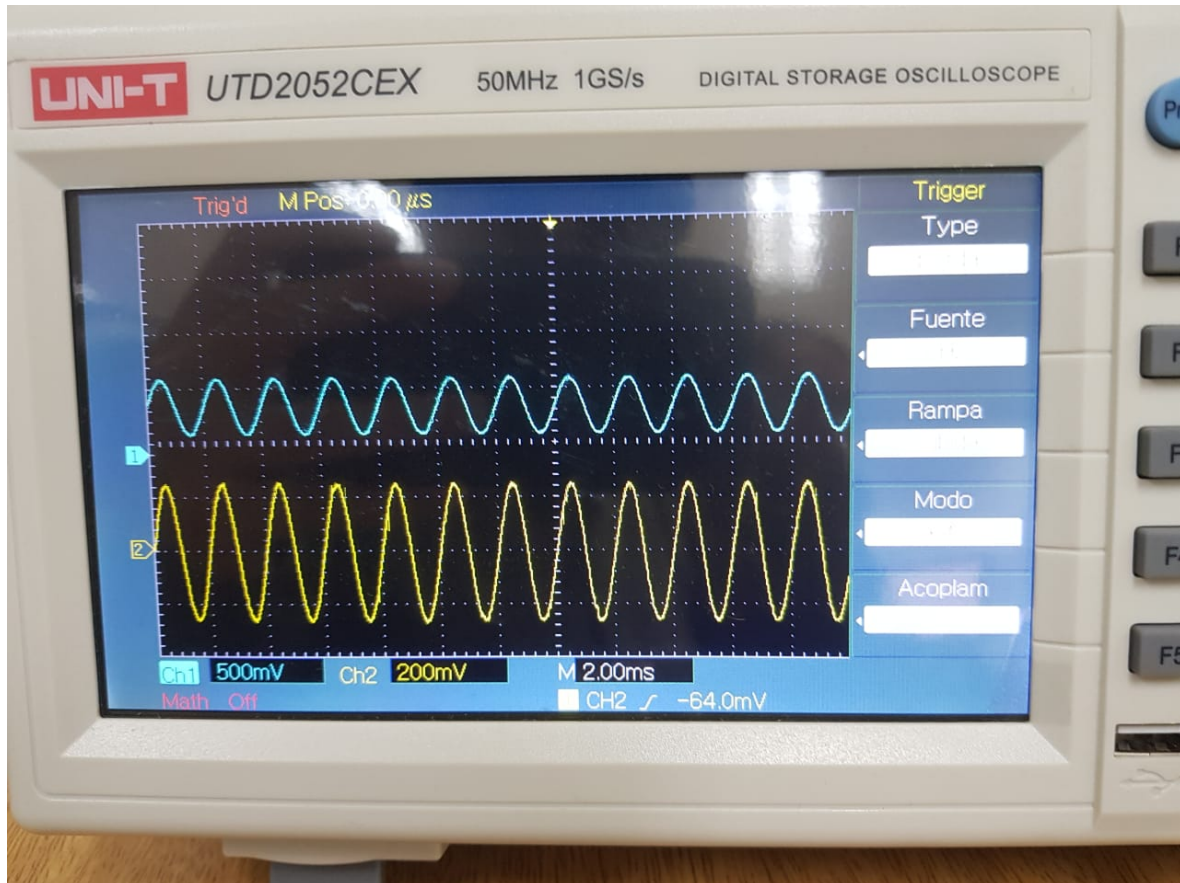
Procesamiento digital de señales

medio del divisor tenemos la señal montada al nivel de continua obtenida por el mismo.



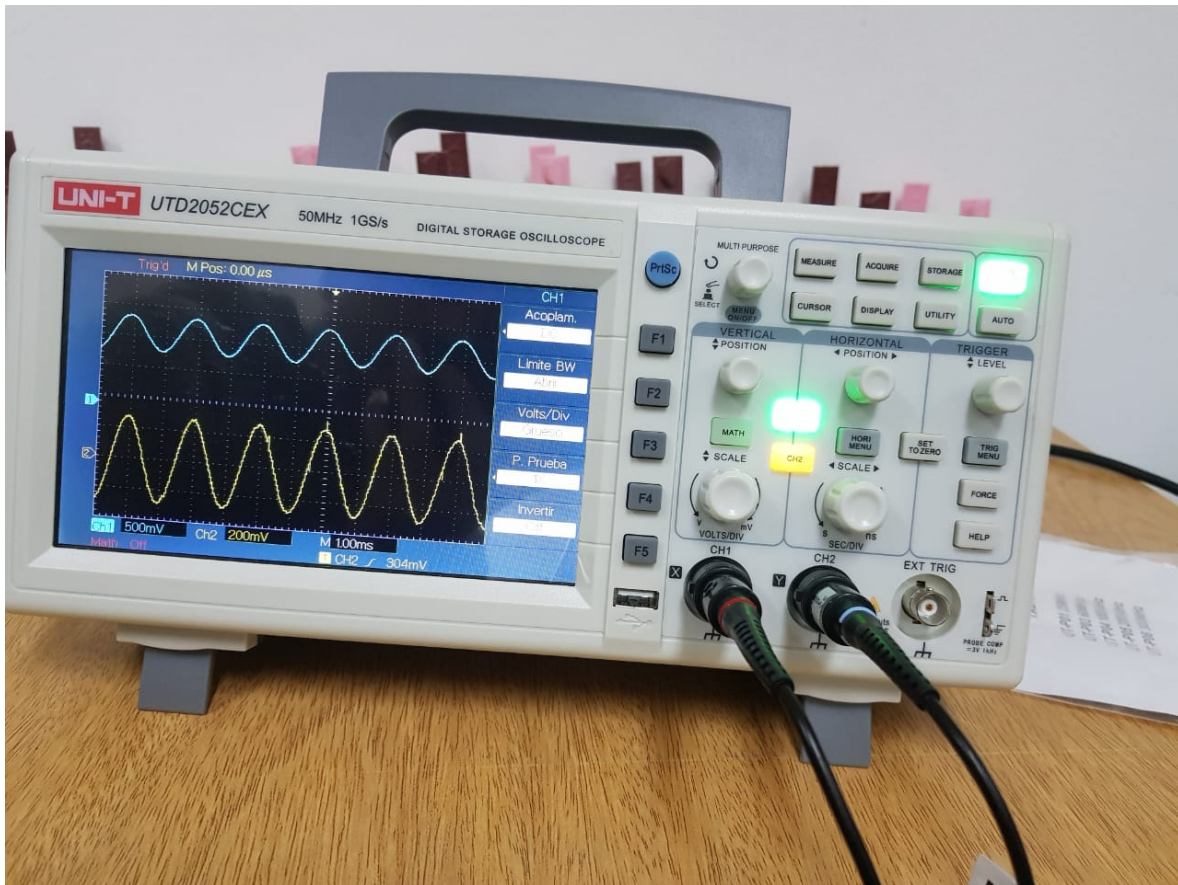
En las imágenes del osciloscopio la señal de color azul es la generada desde la PC, la de color amarillo es la salida del DAC de la placa.

Facultad de Ciencia Exactas, Físicas y Naturales
Procesamiento digital de señales



Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales



Código fuente

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MK64F12.h"
#include "fsl_debug_console.h"
/* TODO: insert other include files here. */
#include "fsl_gpio.h"
```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
#include "fsl_common.h"
#include "stdbool.h"
#include "stdint.h"
#include "arm_math.h"

/* TODO: insert other definitions and declarations here. */

//bandepara para apagar el pit
volatile bool enable_pit = false;

//badera para habilitar los led cada vez que pulso para cambiar el periodo
volatile bool enable_led = false;

volatile bool adc_convert = false;
volatile q15_t adc_value;

//variable para encender los led en base a la frecuencia del clock
volatile uint8_t count;

//las variables para guardar memoria se declaran globales y
// se inicializan dentro del main
long *direc;

//para guardar la ultima posiciond de memoria
long *lim_direc;

//Periodos del PIT para 8, 16, 22, 44 y 48kHz
const long time[5] = {USEC_TO_COUNT(125, PIT_CLK_FREQ),
                      USEC_TO_COUNT(63, PIT_CLK_FREQ),
                      USEC_TO_COUNT(45, PIT_CLK_FREQ),
                      USEC_TO_COUNT(23, PIT_CLK_FREQ),
                      USEC_TO_COUNT(21, PIT_CLK_FREQ)};

q15_t memori[511];

//indice de memoria
uint32_t i;
```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```

/*
 * @brief   Realizar un programa aplicativo que sea capaz de digitalizar
 *          una señal a través del modulo del ADC disponible en la board
FRDM-K64F
 *          a distintas velocidades de muestreo, las velocidades requeridas
son
 *          8K/S, 16K/S 22K/S, 44K/S y 48K/S.
 *          Los cambios de las velocidades de muestreo serán realizados
 *          con una de las teclas de la placa de evaluación, en forma
 *          de un buffer circular. Cada velocidad de muestreo se indicará
 *          a través de un color RGB del LED. Con otra tecla de la placa
 *          se habilitara la adquisición o se parara la misma (Run/Stop).
 *          Los valores adquiridos serán almacenados en memoria en un
buffer
 *          circular de 512 muestras del tipo q15 (fraccional 15bits) y a
 *          su vez serán enviados a través del DAC (de 12bits).
 *
 *          modificaciones: DAC como generador de funciones.
 */
// Handler del puerto A, cambia periodos de muestreo
void GPIOA_IRQHANDLER(void)
{
    enable_led = true;
    GPIO_PortClearInterruptFlags(BOARD_SW3_GPIO, 1U << BOARD_SW3_GPIO_PIN);
    if (direc == lim_direc)
    {
        direc = &time[0];
        count = 0;
    }
else

```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
{
    direc++;
    count++;
}

PIT_SetTimerPeriod(PIT_PERIPHERAL, PIT_CHANNEL_0, (uint32_t)*direc);
__DSB();
}

// Handler del puerto C, pone a correr el PIT
void GPIOC_IRQHANDLER(void)
{
    GPIO_PortClearInterruptFlags(BOARD_SW2_GPIO, 1U << BOARD_SW2_GPIO_PIN);
    if (!enable_pit)
    {
        PIT_StartTimer(PIT_PERIPHERAL, PIT_CHANNEL_0);
        enable_pit = true;
    }
    else
    {
        PIT_StopTimer(PIT_PERIPHERAL, PIT_CHANNEL_0);
        enable_pit = false;
    }
    __DSB();
}

// Handler del PIT, pide una muestra al ADC
```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
void PIT_CHANNEL_0_IRQHANDLER(void)
{
    PIT_ClearStatusFlags(PIT_PERIPHERAL, PIT_CHANNEL_0, kPIT_TimerFlag);

    ADC16_SetChannelConfig(ADC0_PERIPHERAL, ADC0_CH0_CONTROL_GROUP,
&ADC0_channelsConfig[0]);

    __DSB();
}

// Handler del ADC, envia el resultado al DAC
void ADC0_IRQHANDLER(void)
{
    adc_convert = true;

    ADC16_ClearStatusFlags(ADC0_PERIPHERAL,
kADC16_ChannelConversionDoneFlag);

    adc_value = ADC16_GetChannelConversionValue(ADC0_PERIPHERAL,
ADC0_CH0_CONTROL_GROUP);

    DAC_SetBufferValue(DAC0_PERIPHERAL, 0U, adc_value);

    printf("valor convertido:%d", adc_value);

    __DSB();
}

// App
int main(void)
{
    /* Init board hardware. */

    //BOARD_InitBootPins();

    BOARD_InitLEDPins();
    BOARD_InitButtonsPins();

    BOARD_InitBootClocks();

    BOARD_InitBootPeripherals();

#ifdef BOARD_INIT_DEBUG_CONSOLE_PERIPHERAL
    /* Init FSL debug console. */

    BOARD_InitDebugConsole();

```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
#endif

direc = &time;
//guardo la posicion de memoria del ultimo periodo
lim_direc = &time[4];
count = 0;
//arranca con la primera frecuencia de muestreo
PIT_SetTimerPeriod(PIT_PERIPHERAL, PIT_CHANNEL_0, (uint32_t)*direc);
//habilito el led del primer periodo
enable_led = true;
i = 0;
while (1)
{
    if (enable_led)
    {
        switch (count)
        {
            case 0:
                LED_BLUE_ON();
                LED_GREEN_OFF();
                LED_RED_OFF();
                break;
            case 1:
                LED_BLUE_OFF();
                LED_GREEN_ON();
                LED_RED_OFF();
                break;
            case 2:
```


Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
        LED_BLUE_OFF();  
        LED_GREEN_OFF();  
        LED_RED_ON();  
        break;  
    case 3:  
        LED_BLUE_ON();  
        LED_GREEN_ON();  
        LED_RED_OFF();  
        break;  
    case 4:  
        LED_BLUE_ON();  
        LED_GREEN_OFF();  
        LED_RED_ON();  
        break;  
    case 5:  
        LED_BLUE_OFF();  
        LED_GREEN_ON();  
        LED_RED_ON();  
        break;  
    default:  
        LED_BLUE_OFF();  
        LED_GREEN_OFF();  
        LED_RED_OFF();  
    }  
    enable_led = false;  
}  
  
if (adc_convert)  
{  
    memori[i] = adc_value;
```

Facultad de Ciencia Exactas, Físicas y Naturales

Procesamiento digital de señales

```
        i++;  
        if (i == 512)  
            i = 0;  
        adc_convert = false;  
    }  
}  
return 0;  
}
```