Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas

Ingeniería mecatrónica

Implementación de sistemas digitales

Grupo: 4MM3

Práctica 6: Convertidor analógico digital.

Alumno:

Jesús Alfredo Juárez Madera

Docente: Ramón Jaramillo Martínez

Fecha de entrega: 23 de junio del 2022

Objetivo.

Analizar, diseñar e implementar una interfaz de entrada/salida, para resolver un problema específico por medio del módulo ADC y UART.

Introducción.

ADC.

Un convertidor analógico digital (ADC, por sus siglas en inglés) es usado para convertir una señal analógica, como el voltaje, a una forma digital para que pueda ser leída y procesada por un microcontrolador. Consiste en obtener una muestra del nivel de la señal cada cierto momento y así asignarle un valor binario que puede ser procesado digitalmente [1].

UART.

UART (Universal Asynchronous Receiver Transmitter, por sus siglas en inglés) es un protocolo ampliamente usado de comunicación serial. Es un protocolo de comunicación hardware que usa comunicación serial asíncrona con velocidad variable. Por ser asíncrona no tiene reloj, por lo que ambos dispositivos conectados necesitan estar de acuerdo en la velocidad de transmisión [2].

Ilustración 1. Paquete de datos UART [2].

Filtro EMA.

El filtro EMA (Exponential Moving Average, por sus siglas en inglés) consiste en suavizar la señal de entrada mediante la aplicación de la siguiente expresión:

$$y_n = \alpha x_n + (1 - \alpha) y_{n-1}$$

Donde y_n es el valor filtrado, y_{n-1} es el valor filtrado anterior, x_n es el valor muestreado de la señal de entrada y α es el factor que controla el suavizado, en un rango de 0 a 1 [3].

Filtro SMA.

Un filtro SMA (Simple Moving Average, por sus siglas en inglés) es un filtro que esencialmente se comporta como el promedio de una señal de manera que crea un suavizado muy grande en la misma.

Filtro FIR.

El filtro FIR (Finite Impulse Response, por sus siglas en inglés) es un filtro digital que puede ser configurado como pasa bajas como se muestra en la figura. En dicha configuración, su propósito es truncar las frecuencias que existen en una señal, dependiendo de la frecuencia de corte elegida.

$$h_d[n] = \begin{cases} \dfrac{\omega_c}{\pi} \dfrac{sen\left(\omega_c\left(n - \dfrac{M-1}{2}\right)\right)}{\omega_c\left(n - \dfrac{M-1}{2}\right)} & 0 \leq n \leq M-1 \\ \dfrac{\omega_c}{\pi} & n = \dfrac{M-1}{2} \end{cases}$$

Ilustración 2. Filtro FIR pasa bajas [4].

Desarrollo.

Para el desarrollo de la práctica se consideró utilizar un potenciómetro como entrada analógica de voltaje por la disponibilidad y facilidad de implementación.

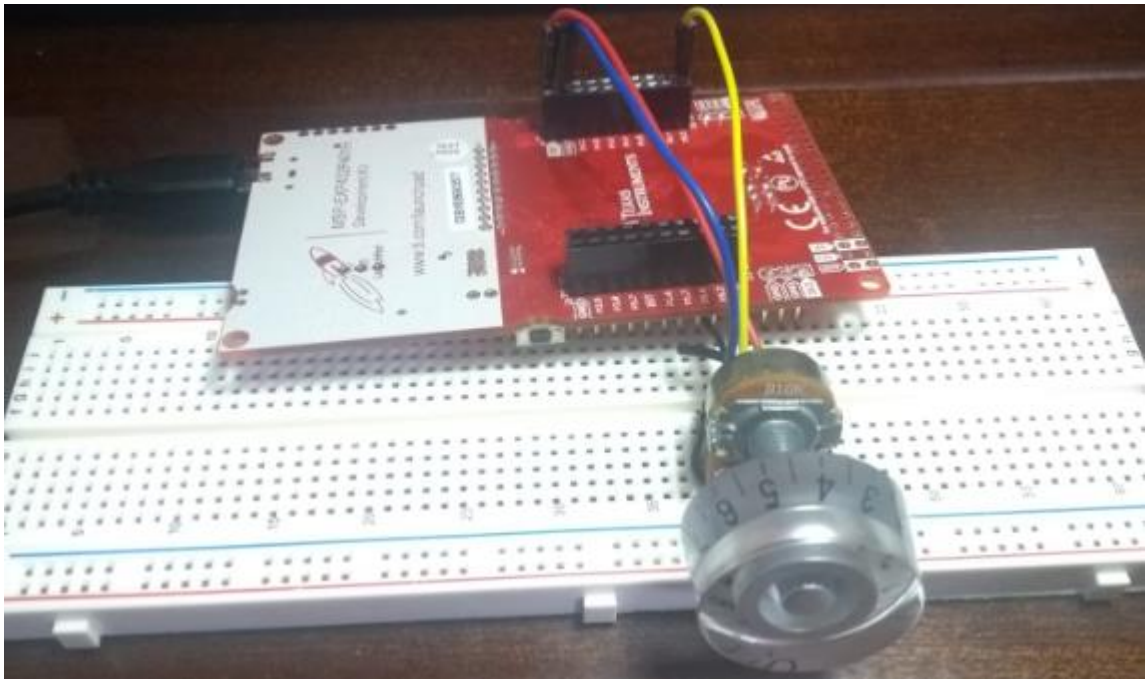

Ilustración 3. Circuito de prueba del ADC.

Adquirir una señal analógica.

Como entrada analógica se configuró el pin P5.5 modo analógico. En la hoja de datos se puede apreciar que para usar el pin P5.5 tiene funcionalidad del módulo ADC en el canal A0, por lo que para usarlo se configura en su función terciaria (P5SEL1.5 = 1, P5SEL0.5 = 1).

P5.5/A0	5	P5.5 (I/O)	I: 0; O: 1	0	0
		N/A	0	0	1
		DVSS	1		
		N/A	0	1	0
		DVSS	1		
		A0 (2)	X	1	1

Ilustración 4. Funcionalidades del P5.5 [5].

Para configurar el módulo ADC14 se siguieron los siguientes pasos:

1. Habilitar el módulo.

2. Inicializar el módulo utilizando el SMCLK sin divisor.
3. Se selecciona la ADC_MEM0 como el lugar para almacenar la conversión y se configura en modo repetitivo.
4. Para la referencia de voltaje positiva se elije VCC y para la negativa VSS y se deshabilitan las entradas diferenciales (no se requieren).
5. Se configura un muestreo automático.
6. Finalmente se habilita la conversión ADC.

El resultado de la conversión ADC se obtiene guardando en una variable el contenido de la memoria ADC_MEM0 y posteriormente se normaliza al voltaje admisible por la MSP432P401R de 3.3V. Para esto ultimo se toma en cuenta que estamos utilizando una resolución de 14 bits, por lo que la señal normalizada se obtiene de una regla de tres, considerando que a 3.3V le corresponde el valor 2^14.

Finalmente, se utiliza el servicio de interrupción del módulo ADC para asegurar que se obtendrá la conversión solo cuando este completa.

Aplicar filtro EMA, SMA y FIR.

Para obtener las salidas filtradas se crean nuevas variables en la que se almacenen las señales procesadas.

Para el caso del filtro EMA simplemente es necesario almacenar en una variable el resultado de la función de transferencia del filtro. Dado que la expresión matemática contempla salidas anteriores, se utiliza una variable auxiliar que guarda el valor de la señal filtrada para la siguiente vez que se dispara la interrupción de conversión completa.

Con el filtro SMA ocurre de manera similar, ya que solo es necesario apoyarse de una variable donde almacenar la suma de los valores de la señal y otra donde se cuente el número de muestra actual.

El filtro FIR se diseña como pasa bajas, con una frecuencia de corte de 1.2kHz y utilizando ventana rectangular para truncar la señal después de la frecuencia de corte. Para implementar el filtro se consideró una frecuencia normalizada con el

número de muestras por segundo, que se estimó como 200ksps ya que con el reloj a 12MHz y con las referencias de voltaje internas, el máximo de muestras por segundo está limitado a 200ksps.

Enviar señales por UART.

El módulo UART se configuró a 9600 bps en el canal EUSCI_A0 que transmite datos de manera serial a la computadora.

Para el envió de las señales, se consideró enviar un valor flotante a tres dígitos de cada salida, seguido de una coma y finalizando con \r\n para identificar el fin de la cadena de datos por muestra.

Análisis de resultados.

En la figura se puede observar como línea blanca a la señal original, mientras que la línea roja muestra al filtro EMA. Con un factor Alpha de 0.1, se puede apreciar como la señal de salida es suavizada en todos los cambios.

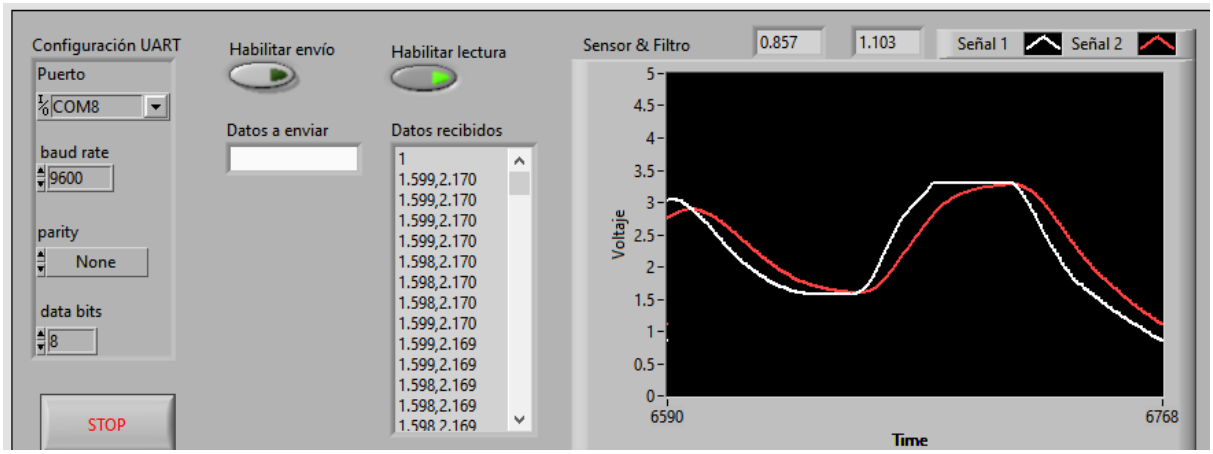

Ilustración 5. Prueba del filtro EMA.

Respecto al filtro SMA se puede apreciar que tarda en alcanzar a la señal original, debido a que se necesita de muchas muestras para tener un cambio visible en la salida.

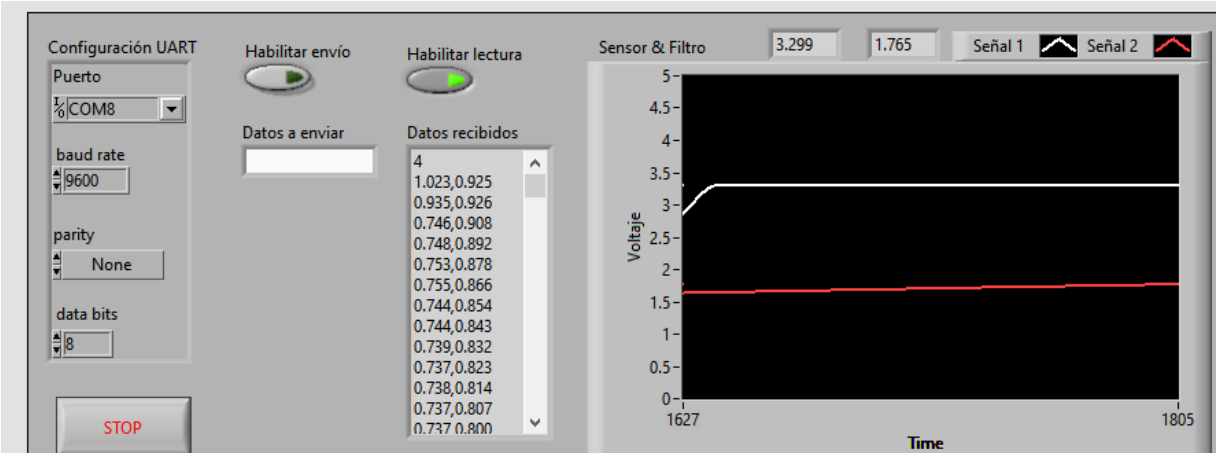

Ilustración 6. Prueba del filtro SMA.

El filtro FIR arrojó frecuencia pasante en el tiempo que funcionó. Probablemente error fue una indeterminación en la expresión del filtro cuando llego a cierto número de muestras.

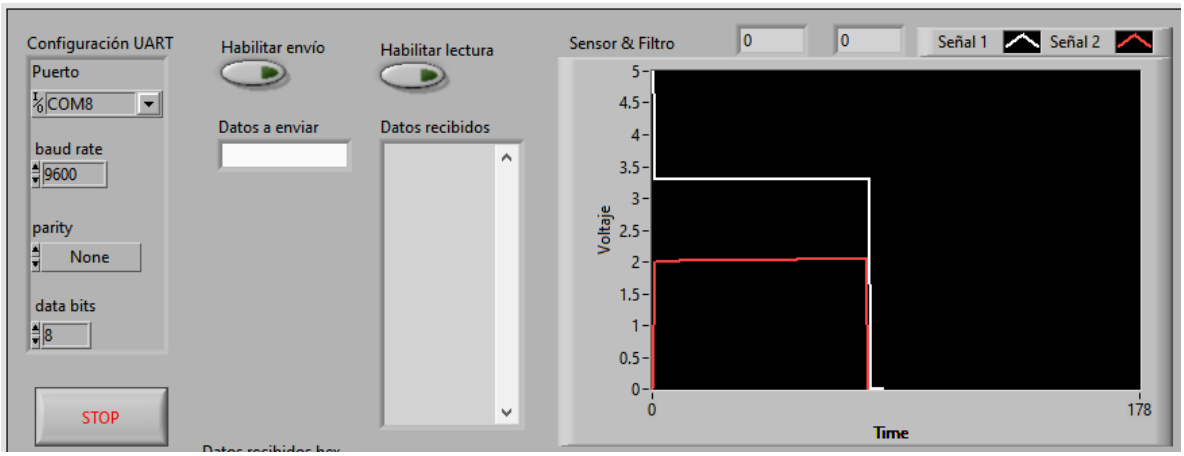

Ilustración 7. Prueba del filtro FIR.

Conclusión.

El módulo ADC se utilizó con éxito al tener de interfaz la salida de un divisor de voltaje controlado por potenciómetro. El uso del potenciómetro permitió enviar una señal analógica provocada desde la misma fuente de la tarjeta de desarrollo, por lo que muy simple su implementación. El uso de un generador de señales hubiese sido ideal solo en el caso de pruebas del filtro FIR, por lo que no se llegó a una comprobación satisfactoria de este.

El módulo UART permitió enviar las señales a la vez a la PC, que en todo caso puede llegar a ser cualquier otro dispositivo que admita UART, por lo que se puede hacer una comparación al mismo tiempo de todas las señales.

Referencias

[1] «Analog-to-Digital Converter,» ScienceDirect, [En línea]. Available: https://www.sciencedirect.com/topics/engineering/analog-to-digital-converter. [Último acceso: 23 Junio 2022].

[2] «UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter,» Analog, [En línea]. Available: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. [Último acceso: 23 Junio 2022].

[3] C. C. Baquero Castañeda y V. F. Martínez Diaz, «Diseño y construcción de un prototipo electrónico para la detección del robo en tapas del acueducto mediante señales de vibraciones,» Universidad de La Salle, Bogotá, 2020.

[4] «Ejemplos filtros FIR,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=_bHTfY57JoY&ab_channel=%E3%80%82%E3%83%AA%E3%82%B4. [Último acceso: 23 Junio 2022].

[5] Texas Instruments, MSP432P401R, MSP432P401M Mixed-Signal Microcontrollers, Texas Instruments, 2015.

Anexo: Código en C.

```
/*
 * Práctica 6
 * -Adquirir señal analógica.
 * -Aplicar filtro EMA, SMA y FIR.
 * -Filtro FIR debe ser diseñado como: filtro pasa bajas, Fc=1200kHz, método de ventanas, ventana rectangular.
 * -Enviar todas las señales obtenidas a la salida de los filtros por UART, también la original.
 */

/* DriverLib Includes */
#include <ti/devices/msp432p4xx/driverlib/driverlib.h>

/* Standard Includes */
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <stdio.h>

// Declaración de variables
static volatile uint16_t ValorADC;
static volatile float VoltajeNormalizado, y, alpha, y_n, suma, y2, y3, pi, hn;
static volatile char Signals[33], Original[6], EMA[9], SMA[9], FIR[9];
static volatile int n, M;
static volatile double fc, wc;

const eUSCI_UART_ConfigV1 uartConfig =
{
        EUSCI_A_UART_CLOCKSOURCE_SMCLK,          // SMCLK Clock Source
        78,                                      // BRDIV = 78
        2,                                       // UCxBRF = 2
        0,                                       // UCxBRS = 0
        EUSCI_A_UART_NO_PARITY,                  // No Parity
        EUSCI_A_UART_LSB_FIRST,                  // LSB First
        EUSCI_A_UART_ONE_STOP_BIT,               // One stop bit
        EUSCI_A_UART_MODE,                       // UART mode
        EUSCI_A_UART_OVERSAMPLING_BAUDRATE_GENERATION,  // Oversampling
        EUSCI_A_UART_8_BIT_LEN                   // 8 bit data length
};

int main(void)
{
    /* Stop Watchdog  */
    MAP_WDT_A_holdTimer();

    // Inicialización de variables
    ValorADC = 0;
    VoltajeNormalizado = 0;
    y = 0;
    y_n = 0;
    alpha = 0.1;
    n = 1;
    y2 = 0;
```

```
    suma = 0;
    M = 21;
    fc = 1200;
    wc = fc/200000;
    pi = 3.1416;

    // Configuración de GPIOs
    GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P5, GPIO_PIN5,              // GPIO
P5.5 como entrada analógica
                                       GPIO_TERTIARY_MODULE_FUNCTION);     // A0

    /* P1.2 and P1.3 en modo UART  */
    GPIO_setAsPeripheralModuleFunctionInputPin(GPIO_PORT_P1,
            GPIO_PIN2 | GPIO_PIN3, GPIO_PRIMARY_MODULE_FUNCTION);

//-- ADC ----------------------------------------------------------------------
    /* Configura Flash estado espera */
    FlashCtl_setWaitState(FLASH_BANK0, 1);
    FlashCtl_setWaitState(FLASH_BANK1, 1);

    /* Configuracion DCO 12 MHZ*/
    PCM_setPowerState(PCM_AM_LDO_VCORE1);
    CS_setDCOCenteredFrequency(CS_DCO_FREQUENCY_12);

    /* Habilitamos FPU*/
    FPU_enableModule();
    FPU_enableLazyStacking();

    // Configuración del ADC
    ADC14_enableModule();                                                  // Se habilita el modulo
    ADC14_initModule(ADC_CLOCKSOURCE_SMCLK, ADC_DIVIDER_1, ADC_DIVIDER_1,
ADC_NOROUTE);      // Se inicializa el modulo con reloj maestro,

    // Configuración de memoria del ADC
    ADC14_configureSingleSampleMode(ADC_MEM0, true);                       // La conversión se
almacena en la memoria MEM0
                                                        // repitiendo la conversión
    ADC14_configureConversionMemory(ADC_MEM0, ADC_VREFPOS_AVCC_VREFNEG_VSS,
// Referencia de memoria default
                                ADC_INPUT_A0, ADC_NONDIFFERENTIAL_INPUTS);  // Canal A0 del
P5.5. Sin diferencial entre canales

    // Configuración de TIMER
    ADC14_enableSampleTimer(ADC_AUTOMATIC_ITERATION);

    // Se habilita la conversión ADC
    ADC14_enableConversion();
    ADC14_toggleConversionTrigger();

//-- UART ---------------------------------------------------------------------------------
    /* Configuración UART con base a la estructura de arriba */
    UART_initModule(EUSCI_A0_BASE, &uartConfig);

    /* Habilitamos UART */
```

```
    UART_enableModule(EUSCI_A0_BASE);

//-- Interrupciones ------------------------------------------------------------------------
    ADC14_enableInterrupt(ADC_INT0);            // Se habilita la interrupción del canal 0 del ADC
    Interrupt_enableInterrupt(INT_ADC14);       // Se habilita la interrupcion por ADC

    UART_enableInterrupt(EUSCI_A0_BASE, EUSCI_A_UART_RECEIVE_INTERRUPT);
    Interrupt_enableInterrupt(INT_EUSCIA0);

    Interrupt_enableMaster();                   // Se habilitan las interrupciones en general

//-------------------------------------------------------------------------------------------
    while(1)
    {
        PCM_gotoLPM0();
    }
}
//-------------------------------------------------------------------------------------------

    /* Servicio de interrupción ADC */
void ADC14_IRQHandler(void){
    uint16_t status = ADC14_getEnabledInterruptStatus();    // Estatus de la interrupción de
activación
    ADC14_clearInterruptFlag(ADC_INT0);                     // Se limpia la bandera de interrupción
    if (ADC_INT0 & status){
        // Obtener la señal de voltaje
        ValorADC = ADC14_getResult(ADC_MEM0);
        VoltajeNormalizado = (ValorADC*3.3)/16383;          // Se normaliza el voltaje a 3.3V con
resolución de 14 bits

        // Aplicar filtro EMA
        y = (alpha * (VoltajeNormalizado)) + ((1 - alpha)*y_n); // Expresión del filtro EMA
        y_n = y;

        // Aplicar filtro SMA
        suma = VoltajeNormalizado + suma;
        y2 = suma/n;                                // Expresión del filtro SMA
        n++;

        // Aplicar filtro FIR
        hn = (wc/pi) * (sin(wc * (n - (M - 1)/2)) / wc*(n - (M - 1)/2));
        y3 = hn * VoltajeNormalizado;                           // Filtro FIR aplicado

        // Enviar señales por UART
        sprintf(Original, "%.3f,", VoltajeNormalizado);         // Se guardan las señales como strings
        sprintf(EMA, "%.3f,", y);
        sprintf(SMA, "%.3f,", y2);
        sprintf(FIR, "%.3f\r\n", y3);

        char aux1 = strcat(Original,EMA);                   // Se concatenan las señales
        char aux2 = strcat(aux1,SMA);
        char aux3 = strcat(aux2,FIR);
        UART0_OutString(aux3);                          // Se envian por UART
    }
```

```
}

/*Funcion para envio de string */
void UART0_OutString(char *pt){
   while(*pt){
      UART0_OutChar(*pt);
      pt++;
   }
}
void UART0_OutChar(char letra){
   UART_transmitData(EUSCI_A0_BASE, letra);
}
```