



| | | | |
|---|--|---------------------------------|---|
|  | UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO FACULTAD DE INGENIERÍA Sistemas Embebidos | Semestre: 2021-1 |  |
| | Prof. Dr. Saúl De La Rosa Nieves | Grupo 3 Página 1 de 6 | |

| TAREA-EXAMEN 2 | |
|-----------------------|-------------------------------|
| Título: | Señal senoidal con SPI |
| Fecha: | 14-06-21 |
| Preparado por: | Fiel Muñoz Teresa Elpidia |
| Evaluación: | |

I. Planteamiento del proyecto

Objetivo:

Diseñar una señal senoidal a partir del protocolo SPI utilizando el potenciómetro digital MCP41010.

Descripción del proyecto:

El proyecto deberá tener un generador de señal senoidal que sea visible en la graficadora del CCS y a su vez se debe poder variar su frecuencia.

II. Requerimientos del proyecto

Para la construcción del proyecto se presentaron los siguientes requerimientos:

Requerimientos de Hardware

1. Como unidad de procesamiento de la caja registradora utilice el microcontrolador TM4C1294NCPDT.
2. Se debe emplear un potenciómetro digital con interfaz SPI MCP41010.

Requerimientos de funcionamiento

1. El generador debe mostrar una señal senoidal y debe ser capaz de variar su frecuencia.

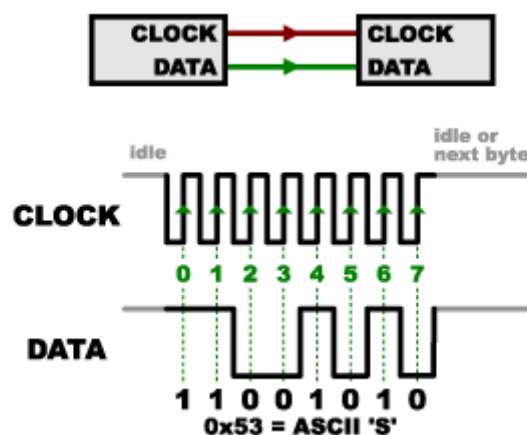
III. Marco teórico (antecedentes necesarios para el diseño)

Para el diseño de este proyecto se necesitaron conocimientos previos sobre el protocolo SPI.

Protocolo SPI

El SPI es un protocolo de comunicación síncrona de 4 hilos, entre dispositivos electrónicos presentado por Motorola, ahora Freescale en 1982, que ha ganado bastante aceptación en la industria como sistema de comunicación de muy corta distancia, normalmente dentro la placa de circuito impreso. Es un protocolo de transmisión que permite alcanzar velocidades muy altas y que se diseñó pensando en comunicar un microcontrolador con distintos periféricos y que funciona a full dúplex.

SPI utiliza una solución síncrona, porque utiliza unas líneas diferentes para los datos y el Clock. El Clock es una señal que indica al que escucha exactamente cuándo leer las líneas de datos, con lo que el problema de pérdida de sincronía se elimina de raíz.

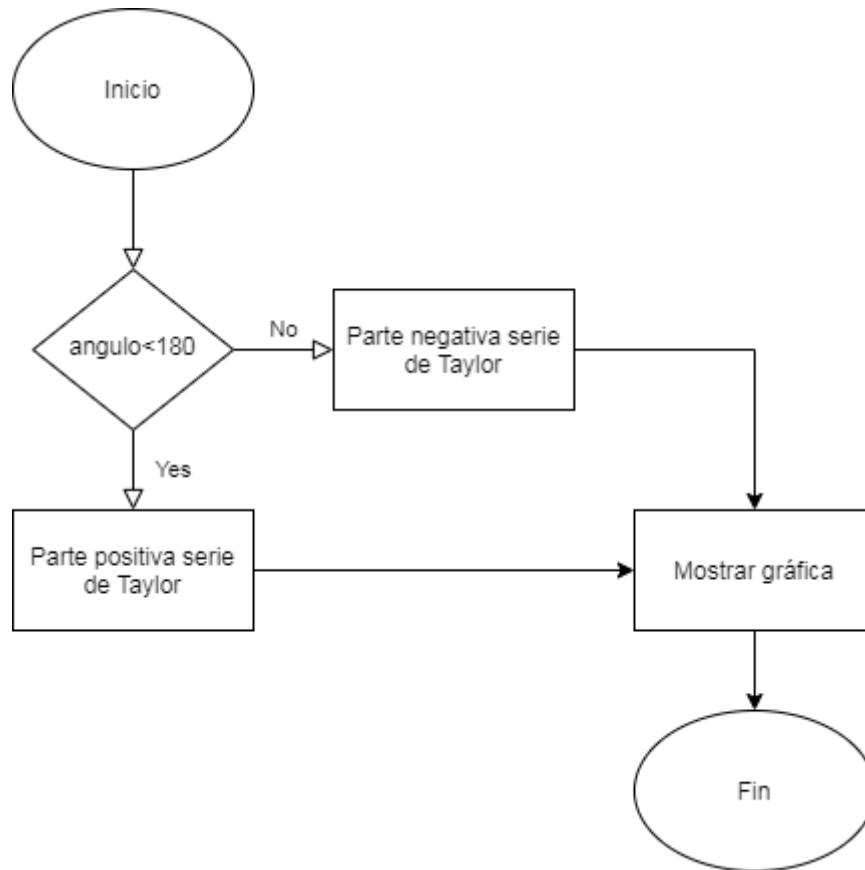


Diseño

Para el diseño de esta señal en el generador se utilizarán términos de la serie de Taylor para una función senoidal, una parte importante a resaltar es que se deben considerar las dos partes de la señal, la parte positiva y la parte negativa.

IV. Diagrama de flujo y código comentado (en caso de VHDL o código de programa).

Diagrama de flujo



Código

```

//----LIBRERIAS----
#include <stdint.h>
#include "inc/tm4c1294ncpdt.h"

//---VARIABLES---
void SSI0_init (void);
void SSI0_sendData (uint16_t dat);
void pot_setVal (uint8_t slider);
void SysTick_Init(void);
void SysTick_Wait(uint32_t retardo);

uint32_t adc_result;
uint8_t y = 0x00; // Valor inicial.
float x=0; //angulo
int ascendente = 1;
float val;
int val_int;
int signaltype=0;

int main(void){
    SysTick_Init();
    SSI0_init(); // Función que habilita el SPI-GPIO's-ADC
    int delay ;
    int m;
  
```



```
while (1) {
    //senoidal
    signaltype=4;
    delay = ((1/0.1)*4000000)/32;
    for (m = 0; m < 180; m=m+10){
        x=(2*3.14159265358979*m)/360; //grados a radianes
        // aproximacion con serie de taylor
        val= x - ((x*x*x)/(2*3)) + ((x*x*x*x*x)/(2*3*4*5)) - ((x*x*x*x*x*x*x)/(2*3*4*5*6*7)) +
        ((x*x*x*x*x*x*x*x*x)/(2*3*4*5*6*7*8*9)) - ((x*x*x*x*x*x*x*x*x*x*x)/(2*3*4*5*6*7*8*9*10*11)) ;
        val=(val*100);
        val_int = val+150;
        pot_setVal(val_int);
        ADC0_PSSI_R = 0x0008; // Inicia conversión del SS3
        while ((ADC0_RIS_R & 0x08)==0); // Espera a que SS3 termine conversión (polling)
        adc_result = (ADC0_SSFIF03_R & 0xFFF); // Resultado en FIF03 se asigna a variable "result"
        ADC0_ISC_R = 0x0008; // Limpia la bandera RIS del ADC0
        GPIO_PORTK_DATA_R = adc_result>>4;
        SysTick_Wait(delay);
    } //end FOR
    for (m = 180; m > 0; m=m-10){
        x=(2*3.14159265358979*m)/360; //grados a radianes
        // aproximacion con serie de taylor
        val= x - ((x*x*x)/(2*3)) + ((x*x*x*x*x)/(2*3*4*5)) - ((x*x*x*x*x*x*x)/(2*3*4*5*6*7)) +
        ((x*x*x*x*x*x*x*x*x)/(2*3*4*5*6*7*8*9)) - ((x*x*x*x*x*x*x*x*x*x*x)/(2*3*4*5*6*7*8*9*10*11)) ;
        val=- (val*100);
        val_int = val+150;
        pot_setVal(val_int);
        ADC0_PSSI_R = 0x0008; // Inicia conversión del SS3
        while ((ADC0_RIS_R & 0x08)==0); // Espera a que SS3 termine conversión (polling)
        adc_result = (ADC0_SSFIF03_R & 0xFFF); // Resultado en FIF03 se asigna a variable "result"
        ADC0_ISC_R = 0x0008; // Limpia la bandera RIS del ADC0
        GPIO_PORTK_DATA_R = adc_result>>4;
        SysTick_Wait(delay);
    } //END for
} //END while(1)
} //END MAIN
void SSI0_init (void) {
    SYSCCTL_RCGCSSI_R = SYSCCTL_RCGCSSI_R0; // Activa reloj al SSI0
    while ((SYSCCTL_PRSSI_R & SYSCCTL_PRSSI_R0) == 0); // Espera a que este listo
    SYSCCTL_RCGCGPIO_R |= 0x0211; // Activa reloj del GPIO A/K
    while ((SYSCCTL_PRGPIO_R & SYSCCTL_PRGPIO_R0) == 0); // Espera a que este listo
    SYSCCTL_RCGCADC_R = 0x01; // 6) Habilita reloj para ADC0
    while((SYSCCTL_PRADC_R&0x01)==0); // Espera a reloj este listo

    //PortK
    GPIO_PORTK_DIR_R = 0xFF;
    GPIO_PORTK_DEN_R = 0xFF;
    GPIO_PORTK_PUR_R = 0xFF;
    GPIO_PORTK_DATA_R = 0x00;
    //PORT A
    GPIO_PORTA_AHB_AFSEL_R |= 0x3C; // Selecciona la función alterna de PA[2:5].
    GPIO_PORTA_AHB_PCTL_R = (GPIO_PORTA_AHB_PCTL_R & 0xFFF000FF) | 0x00FFFF00; // Configura las terminales de PA a su
función de SSI0.
    GPIO_PORTA_AHB_AMSEL_R = 0x00; // Deshabilita la función analógica
    GPIO_PORTA_AHB_DIR_R = (GPIO_PORTA_AHB_DIR_R & ~0x3C) | 0x1C; // Configura al puerto como salida
    GPIO_PORTA_AHB_DEN_R |= 0x3C; // Habilita la función digital del puerto
    //PORT E
    GPIO_PORTE_AHB_DIR_R = 0x00; // 2) PE4 entrada (analógica)
    GPIO_PORTE_AHB_AFSEL_R |= 0x10; // 3) Habilita Función Alterna de PE4
    GPIO_PORTE_AHB_DEN_R = 0x00; // 4) Deshabilita Función Digital de PE4
    GPIO_PORTE_AHB_AMSEL_R |= 0x10; // 5) Habilita Función Analógica de PE4
    //ADC
    ADC0_PC_R = 0x01; // 7) Configura para 125Ksmp/s ( 1 octavo de la frecuencia de conversión configurada)(p.1159)
    ADC0_SSPI_R = 0x0123; // 8) SS3 con la más alta prioridad (p.1099)
    ADC0_ACTSS_R = 0x0000; // 9) Deshabilita SS3 antes de cambiar configuración de registros (p. 1076)
    ADC0_EMUX_R = 0x0000; // 10) iniciar muestreo por software (p.1091)
    ADC0_SSEMUX3_R = 0x00; // 11) Entradas AIN(15:0) (p.1146)
    ADC0_SSMUX3_R = (ADC0_SSMUX3_R & 0xFFFFFFF0) + 9; // canal AIN9 (PE4) (p.1141)
    ADC0_SSCTL3_R = 0x0006; // 12) Si: sensor de temperatura, Habilitación de INR3, Fin de secuencia; No:muestra
diferencial (p.1142)
    ADC0_IM_R = 0x0000; // 13) Deshabilita interrupciones de SS3(p. 1081)
```

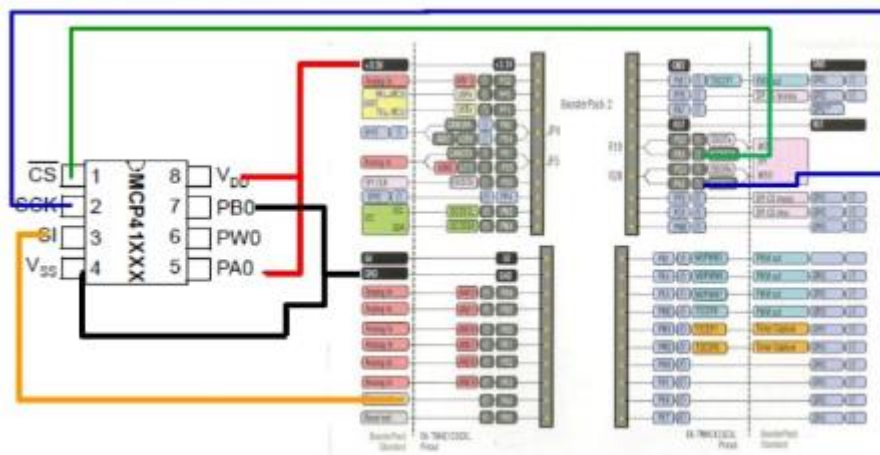
```

ADC0_ACTSS_R |= 0x0008; // 14) Habilita SS3 (p.1077)
SYSCCTL_PLLFREQ0_R |= SYSCCTL_PLLFREQ0_PLLPWR; // encender PLL
while((SYSCCTL_PLLSTAT_R&0x01)==0); // espera a que el PLL fije su frecuencia
SYSCCTL_PLLFREQ0_R &= ~SYSCCTL_PLLFREQ0_PLLPWR; // apagar PLL
// Se recomienda Limpiar la bandera RIS del ADC0
ADC0_ISC_R = 0x0004;
//PSI
SSI0_CR1_R = 0x00; // Selecciona modo maestro/deshabilita SSI0. (p. 1247)
SSI0_CPSR_R = 0x02; // preescalador (CPSDVSR) del reloj SSI (p. 1252)
// configura para Freescale SPI; 16bit; 4 Mbps; SPO = 0; SPH = 0 (p. 1245)
SSI0_CR0_R = (0x0100 | SSI_CR0_FRF_MOTO | SSI_CR0_DSS_16) & ~(SSI_CR0_SPO | SSI_CR0_SPH);
SSI0_CR1_R |= SSI_CR1_SSE; // Habilita SSI0.
} //END SSI0_init
void SSI0_sendData (uint16_t dat) {
  // Envia dato de 16-bit
  while ((SSI0_SR_R & SSI_SR_BSY) != 0); // espero si el bus está ocupado
  SSI0_DR_R = dat; // envia dato.
}
void pot_setVal(uint8_t slider) {
  //Combine el valor del control deslizante con el código de comando de escritura.
  // Estructura del mensaje SPI: [comando (8-bits)][deslizador (8-bits)]
  SSI0_sendData(0x1100 | slider);
}
void SysTick_Init(void){ NVIC_ST_CTRL_R = 0;
  NVIC_ST_RELOAD_R = NVIC_ST_RELOAD_M;
  NVIC_ST_CURRENT_R = 0; NVIC_ST_CTRL_R = 0x00000001;
}
void SysTick_Wait(uint32_t retardo){
  //ECUACION
  //retardo=T[s]*4 000 000
  NVIC_ST_RELOAD_R= retardo-1; //número de cuentas por esperar
  NVIC_ST_CURRENT_R = 0; while((NVIC_ST_CTRL_R & 0x00010000)==0){} //espera hasta que la bandera COUNT sea valida
}

```

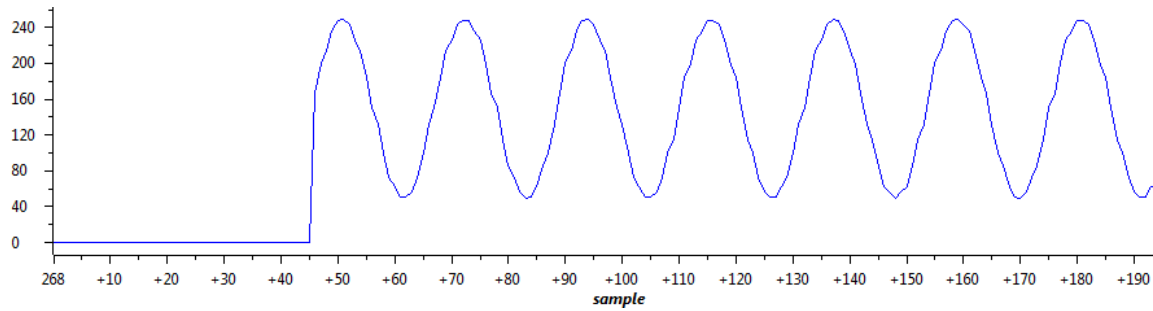
V. Construcción

Para la construcción se realizó el siguiente alambrado tomado de la presentación de clase para el protocolo SPI:

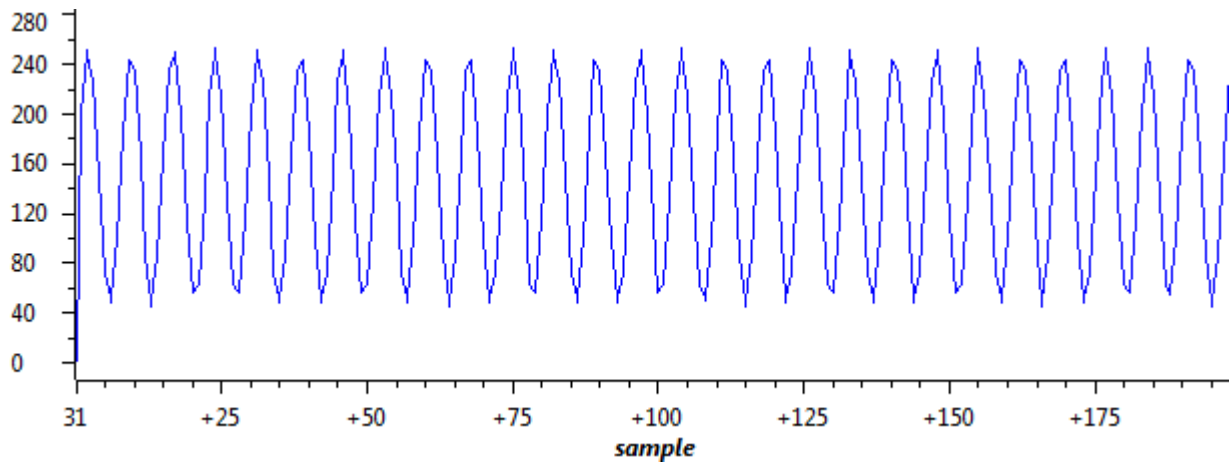


VI. Resultados y – Conclusiones

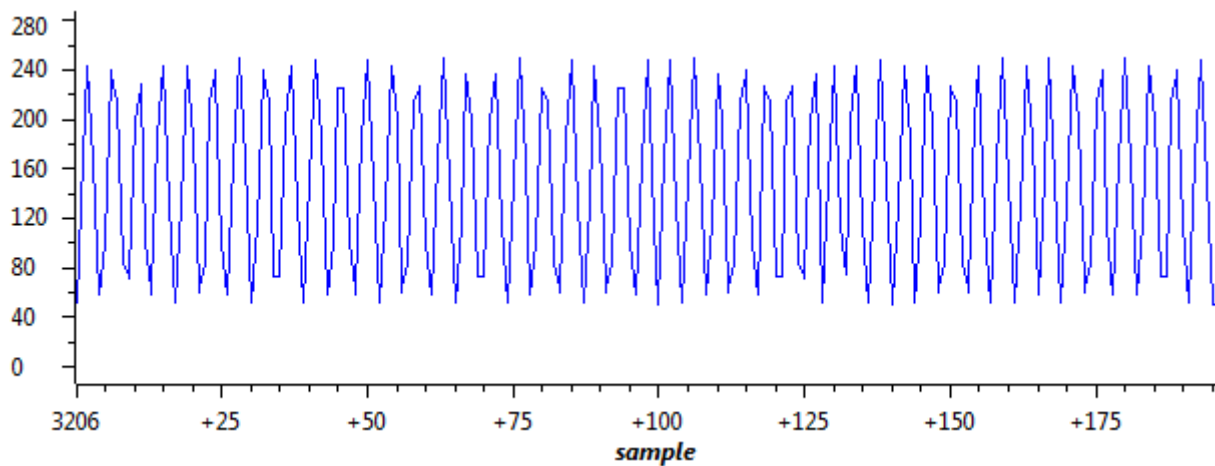
0.1 Hz:



0.3 Hz:



0.5 Hz:



En conclusión, podemos ver cómo es posible realizar una señal senoidal para poder mostrarla con el protocolo SPI, por otra parte, tenemos que tomar en cuenta sus limitaciones sobre la frecuencia ya que al llegar a 0.5Hz la señal se comienza a deformar. Esta práctica nos muestra cómo se puede llegar a generar un generador de funciones que es lo que se realizará en la práctica conjunta con I2C y App Inventor.