Comunicaciones Digitales — Actividad de Laboratorio

Exploración de conceptos de conversion A/D con Raspberry Pi Pico

Programa: Ingeniería en Telecomunicaciones
Institución: Universidad Militar Nueva Granada
Duración sugerida: 2–3 horas de laboratorio

Objetivo general

Comprender los conceptos de conversión A/D y su aplicación el el contexto de la arquitectura de un sistema de comunicación digital.

Procedimiento

Parte A — Uso del conversor A/D

1. Seleccione una entrada del conversor A/D del dispositivo Raspberry pi pico 2W. Modifique el programa de la figura 1 como corresponda.

```
# Comunicaciones Digitales UMNG / jose.rugeles@Unimilitar.edu.co
# Programa 1: Lectura ADC -> Voltios

import machine
import utime

ADC_PIN = 26      # GP26 -> ADC0 (cambia a 27 o 28 si usas ADC1/ADC2)
VREF    = 3.300   # mide tu 3V3 real y ajústalo aquí para mejor exactitud
PERIOD  = 0.05    # segundos entre lecturas (0.05–0.5 )

adc = machine.ADC(ADC_PIN)

# El conversor analógico-digital (ADC) de los microcontroladores RP2040 ( Pico W) ó RP2350 ( Pico 2W) poseen
# una resolución de 12 bits, lo que significa que puede entregar valores enteros entre 0 y 4095
#(es decir, 212 – 1 niveles de cuantización). Sin embargo, la función read_u16() de MicroPython devuelve siempre
# un valor entero de 16 bits en el rango 0 a 65535. Para lograr esto, el valor de 12 bits obtenido
# del ADC es alineado hacia la izquierda dentro de los 16 bits, rellenando con cuatro ceros los
# bits menos significativos

while True:
    raw16  = adc.read_u16()      # 0..65535 (12 bits alineado a la izq.)
    code12 = raw16 >> 4          # 0..4095 (12 bits reales)
    volts  = (code12 * VREF) / 4095.0
    print(f"{volts:.4f}")        # <-- SOLO un número por línea
    utime.sleep(PERIOD)
```

Figura 1: Conversion analoga-digital con Raspberry pi pico 2W

2. Conecte un voltaje variable con ayuda de un potenciometro o empleando una fuente de tensión DC entre 0 y 3.3 Voltios.(Tenga cuidado de no exceder este rango, podria dañar el microcontrolador).
3. Visualice el resultado de la conversion. Analice el empleo de la función función `read_u16()` en el programa.(Lea cuidadosamente el Anexo: Relación entre `read_u16()` y el valor de 12 bits del ADC).
4. Active el *Plotter* en Thonny. Ajuste PERIOD entre 0.05 y 0.5 y observe la visualización.
5. Complemente los ejemplos descritos en el anexo con otros tres valores de conversion. Explique claramente como se pasa del valor de 16 bits al de 12 bits en cada uno de ellos.

Parte B — Muestreo de señales

1. Ejecute el programa **adc.m** dado por el profesor. Analice el resultado y verifique el número de muestras por periodo y la tasa de muestreo para la señal.
2. Cargue el programa **ADC_Sampling.py** en el microntrolador.Analice el código.¿ Que nombre tiene el archivo .CSV generado por el programa?
3. Ajuste una señal de 3Vpp en el generador de señales, con componente DC=1.6 Voltios.Verifique con el osciloscopio antes de conectarla al microcontrolador.
4. Conecte la señal a la entrada del conversor indicada en el programa o seleccionada por su grupo.
5. Analice la información contenida en el archivo .CSV generado.
6. Procese el archivo y grafique la señal con respecto al tiempo. Ajuste los parámtros del programa de matlab **adc.m** para que le permita comparar la señal con la reconstrucción obtenida de las muestras.

Parte C — Análisis estadístico

1. Conecte la entrada del conversor A/D a un voltaje DC constante. (Puede usar un divisor de tensión con un potenciómetro o emplear una fuente DC ó un generador de señales).
2. Seleccione 5 valores diferentes en un rango entre 0 y 3.2 voltios.Pongalos en la tabla.

Test	**V in (DC)**	**Media**	**Desviación estándar**	**Nombre de los archivos**
1				
2				
3				
4				
5				

3. Ejecute el programa **sampling_2.py**. Llene la tabla con los resultados.
4. Descargue los archivos almacenados en el microcontrolador.
5. Realice un programa en matlab o Python que calcule la media y la desviación estándar a partir de las muestras almacenadas. Valide sus resultados con los entregados previamente por el programa realizado en Micropython.
6. Genere las gráficas de los resultados obtenidos. Grafique los histogramas correspondientes . Analice los resultados obtenidos.

Anexo: Relación entre `read_u16()` y el valor de 12 bits del ADC

El conversor analógico–digital (ADC) del microcontrolador RP2040 posee una resolución de **12 bits**, lo que significa que puede entregar valores enteros entre 0 y 4095 (es decir, $2^{12} - 1$ niveles de cuantización). Sin embargo, la función `read_u16()` de MicroPython devuelve siempre un valor entero de **16 bits** en el rango 0 a 65535. Para lograr esto, el valor de 12 bits obtenido del ADC es **alineado hacia la izquierda** dentro de los 16 bits, rellenando con cuatro ceros los bits menos significativos. En otras palabras:

$$\texttt{raw16} = \texttt{code12} \times 16 = \texttt{code12} \ll 4$$

donde:

- `raw16` es el valor leído directamente con `read_u16()`,
- `code12` es el valor real del ADC de 12 bits,
- $\ll 4$ indica un desplazamiento de 4 posiciones a la izquierda en binario.

Para recuperar el valor real de 12 bits, se aplica un desplazamiento hacia la derecha:

$$\texttt{code12} = \texttt{raw16} \gg 4$$

donde $\gg 4$ equivale a dividir entre $2^4 = 16$.

Ejemplo 1: Valor medio de escala (2048)

$$\texttt{code12} = 2048 = 1000\,0000\,0000_{(bin,12b)}$$

El valor devuelto por `read_u16()` es:

$$\texttt{raw16} = 2048 \ll 4 = 32768 = 1000\,0000\,0000\,0000_{(bin,16b)}$$

Al aplicar el corrimiento a la derecha:

$$\texttt{raw16} \gg 4 = 1000\,0000\,0000_{(bin,12b)} = 2048$$

Ejemplo 2: Un cuarto de escala (1024)

$$\texttt{code12} = 1024 = 0100\,0000\,0000_{(bin,12b)}$$

$$\texttt{raw16} = 1024 \ll 4 = 16384 = 0100\,0000\,0000\,0000_{(bin,16b)}$$

$$\texttt{raw16} \gg 4 = 0100\,0000\,0000_{(bin,12b)} = 1024$$

Ejemplo 3: Máxima escala (4095)

$$\texttt{code12} = 4095 = 1111\,1111\,1111_{(bin,12b)}$$

$$\texttt{raw16} = 4095 \ll 4 = 65520 = 1111\,1111\,1111\,0000_{(bin,16b)}$$

$$\texttt{raw16} \gg 4 = 1111\,1111\,1111_{(bin,12b)} = 4095$$

Estos ejemplos muestran que los últimos 4 bits de `raw16` son siempre ceros y que el corrimiento 4 es necesario para recuperar el valor real de 12 bits del ADC. Esta operación es esencial para interpretar correctamente las lecturas y convertirlas a voltaje usando la ecuación:

$$V = \frac{\texttt{code12} \times V_{REF}}{4095}$$

donde V_{REF} es el voltaje de referencia del ADC.