LAB 4

VOLTAJE REGULADO POTEMCIOMETRO MAX

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

POTENCIOMETRO MEDIO

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

POTENCIOMETRO MIN

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

CON PLOTER max

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Con ploter medio

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Con ploter min

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

De nuevo

POTENCIOMETRO MIN CON CODIGO DEL PROFE Y PLOTER

Texto

El contenido generado por IA puede ser incorrecto.

POTENCIOMETRO MAXInterfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Si el periodo es 0,05 la grafica avanza mas raido y muestra mas valores en menos tiempo

Con 0,1 mas lento

PLOTER CON VOLTAJE REFERENCIA 0.5V

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

PLOT CON TODOS LOS VALORES   
Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

CON VOLTRAJE REFERENCIA 1 V

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

CODIGO THONYY CAPTURA

# Comunicaciones Digitales UMNG / jose.rugeles@unimilitar.edu.co

# Muestreo ADC con N muestras/periodo y CSV (limpio)

# Raspberry Pi Pico 2 W (RP2350) - MicroPython

import machine

import utime

# ========= Parámetros de configuración =========

ADC\_PIN = 26 # GP26 -> ADC0

VREF = 3 # Mida en 3V3 (pin 36) y reemplace para mejor exactitud

SIGNAL\_FREQ = 80.0 # Hz

SAMPLES\_PER\_PERIOD = 20 # N muestras por periodo

PERIODS = 2 # >= 2

OUT\_CSV = "adc\_capture\_3.csv"

LED\_PIN = "LED" # LED integrado

# ===============================================

# Derivados

FS\_HZ = SIGNAL\_FREQ \* SAMPLES\_PER\_PERIOD

TS\_US = int(1\_000\_000 / FS\_HZ) # periodo de muestreo (us) truncado

NSAMPS = int(PERIODS \* SAMPLES\_PER\_PERIOD)

adc = machine.ADC(ADC\_PIN)

led = machine.Pin(LED\_PIN, machine.Pin.OUT)

print("-------------------------------------------")

print("Captura ADC - N muestras por periodo (CSV limpio)")

print(f"f\_señal = {SIGNAL\_FREQ} Hz")

print(f"N/periodo = {SAMPLES\_PER\_PERIOD}")

print(f"Periodos a capturar = {PERIODS}")

print(f"Frecuencia de muestreo fs (teórica) = {FS\_HZ:.1f} Hz")

print(f"Periodo de muestreo Ts (trunc) = {TS\_US} us")

print(f"N total de muestras = {NSAMPS}")

print(f"Voltaje de referencia VREF = {VREF:.4f} V")

print("-------------------------------------------")

input("Presione ENTER para iniciar la captura...")

# --- Calentamiento ADC ---

\_ = adc.read\_u16()

utime.sleep\_ms(10)

# --- Métricas internas (no se escriben al CSV) ---

overruns = 0

t\_first\_rel = None

t\_last\_rel = None

# --- Captura y escritura en streaming ---

with open(OUT\_CSV, "w") as f:

# Cabecera simplificada

f.write("n,t\_us,dec,bin,hex,volts\n")

t0 = utime.ticks\_us()

next\_t = t0

led.value(1)

for n in range(NSAMPS):

# Espera activa hasta el instante objetivo

while utime.ticks\_diff(utime.ticks\_us(), next\_t) < 0:

pass

t\_sample = utime.ticks\_us()

# late (solo para métricas internas)

if utime.ticks\_diff(t\_sample, next\_t) > 0:

overruns += 1

raw16 = adc.read\_u16()

code12 = raw16 >> 4

volts = (code12 \* VREF) / 4095.0

bin12s = f"{code12:012b}"

hex12 = f"0x{code12:03X}"

t\_rel = utime.ticks\_diff(t\_sample, t0) # us desde inicio

if t\_first\_rel is None:

t\_first\_rel = t\_rel

t\_last\_rel = t\_rel

# Escribir línea (CSV limpio)

f.write(f"{n},{t\_rel},{code12},{bin12s},{hex12},{volts:.6f}\n")

# Programar siguiente muestra

next\_t = utime.ticks\_add(next\_t, TS\_US)

led.value(0)

# --- Reporte final (en consola, no en el CSV) ---

if NSAMPS > 1 and t\_last\_rel is not None and t\_first\_rel is not None and t\_last\_rel > t\_first\_rel:

fs\_real = 1\_000\_000.0 \* (NSAMPS - 1) / (t\_last\_rel - t\_first\_rel)

else:

fs\_real = 0.0

print("\n✅ Captura finalizada.")

print(f"Archivo CSV: {OUT\_CSV}")

print("Columnas: n, t\_us, dec, bin, hex, volts")

print("----- Métricas internas (no incluidas en el CSV) -----")

print(f"Overruns (llegadas tarde): {overruns} de {NSAMPS} muestras")

print(f"fs\_real ≈ {fs\_real:.1f} Hz (vs fs\_teórica = {FS\_HZ:.1f} Hz)")

print("-------------------------------------------")

print("Nota: si hay muchos overruns, reduzca N/periodo o la frecuencia de la señal.")

CODIGO MATLAB

% Comunicaciones Digitales UMNG / jose.rugeles@Unimilitar.edu.co

% Programa: muestreo

f = 80; % frecuencia de la señal (Hz)

T = 1/f; % periodo (s)

A = 3; % amplitud (V)

NT = 2; % nº de periodos a capturar

N = 40; % nº de muestras por periodo

ts = T/N; % periodo de muestreo (s)

% Vector de tiempo para las muestras (N\*NT puntos exactos)

t = 0:ts:(NT\*T - ts);

V =1.6+ A \* sin(2\*pi\*f\*t); % señal muestreada

% Señal continua para comparación

t\_cont = 0:ts/20:NT\*T;

V\_cont = A \* sin(2\*pi\*f\*t\_cont);

% ===== Gráfica =====

figure('Position',[100 100 900 400]); % tamaño ventana

plot(t\_cont, V\_cont, 'b-', 'LineWidth',2); hold on;

stem(t, V, 'r','filled','LineWidth',1.5);

xlabel('Tiempo (s)','FontSize',20,'FontWeight','bold');

ylabel('Amplitud (V)','FontSize',20,'FontWeight','bold');

title(sprintf('Muestreo de un seno f=%d Hz, N=%d muestras/periodo', f, N), ...

'FontSize',20,'FontWeight','bold');

legend({'Señal continua','Muestras'},'FontSize',14,'Location','best');

grid on;

set(gca,'FontSize',16,'LineWidth',1.2);

Gráfico, Histograma

El contenido generado por IA puede ser incorrecto.

1. El generador de señales de la figura entrega una señal senoidal V(t) de 1 voltio pico a pico y 20 Hz ajustado al rango de cuantización necesario. El generador de pulsos está ajustado a 200 Hz. En la raspberry Pi pico 2W (1) se ejecuta el código TX ny y en la raspberry Pi pico 2W (2) el código RX py. De acuerdo al análisis realizado del sistema y basado en el código de python, calcule

1. LSB

2. Tiempo de transmisión de la medida realizada

3. Penodo de muestreo

4 Bits totales transmitidos

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

% === Nombre del archivo ===

filename = 'adc\_capture\_5.csv';

% === Leer archivo completo como matriz numérica ===

D = readmatrix(filename); % Extrae solo valores numéricos

D = D(~all(isnan(D),2), :); % Elimina filas completamente vacías

% Revisar dimensiones

[nFilas, nCols] = size(D);

if nCols < 2

error('El archivo debe tener al menos 2 columnas (tiempo + señal).');

end

% === Asignar columnas ===

t = D(:,1); % Columna 1 = Tiempo

signals = D(:,2:end); % Resto de columnas = señales (pueden ser varios canales)

% === Normalizar tiempo (µs, ms → s) ===

tUnit = 's';

tmax = max(t) - min(t);

if tmax > 1e6

t = t/1e6; tUnit = 's (desde µs)';

elseif tmax > 1e3

t = t/1e3; tUnit = 's (desde ms)';

end

% === Graficar todas las señales con líneas y puntos ===

figure;

hold on;

colors = lines(size(signals,2)); % Colores distintos para cada canal

for k = 1:size(signals,2)

plot(t, signals(:,k), '-o', ...

'Color', colors(k,:), ...

'LineWidth', 1.2, ...

'MarkerSize', 4, ...

'MarkerFaceColor', colors(k,:));

end

hold off;

grid on;

xlabel(['Tiempo [', tUnit, ']']);

ylabel('Amplitud');

title('Señales capturadas: Líneas y puntos en cada dato');

% Leyenda automática

nSignals = size(signals,2);

leg = arrayfun(@(i) sprintf('Canal %d', i), 1:nSignals, 'UniformOutput', false);

legend(leg, 'Location','best');

xlim([min(t) max(t)]);

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

import machine, utime, micropython

input("ENTER para iniciar muestreo...")

adc = machine.ADC(28)

uart = machine.UART(0, baudrate=1200, bits=8, parity=None, stop=1, tx=machine.Pin(0))

buf = bytearray(80)

i = 0

ready = False

def \_isr(\_):

global i, ready

if ready or i >= 40:

return

c = adc.read\_u16()

p = i \* 2

buf[p] = c & 0xFF

buf[p+1] = (c >> 8) & 0xFF

i += 1

if i == 40:

ready = True

pulse = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL\_DOWN)

pulse.irq(trigger=machine.Pin.IRQ\_RISING, handler=\_isr)

# Espera a completar las muestras para enviar

while not ready:

utime.sleep\_ms(1)

# Desactiva interrupcion y envía un solo bloque

pulse.irq(handler=None)

uart.write(buf)

print("Enviado")

Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

% validacion\_adc\_final.m

% Programa para validación de resultados del ADC Raspberry Pi Pico

% Versión para archivos específicos: 1.txt, 2.txt, 3.txt, 4.txt, 5.txt

clear all;

close all;

clc;

% Definir archivos .txt a analizar (según tu tabla)

archivos = {

'1.txt', 0.00;

'2.txt', 1.10;

'3.txt', 1.70;

'4.txt', 2.50;

'5.txt', 2.80

};

% Resultados esperados de MicroPython (de tu tabla)

medias\_pico = [0.01289; 1.11962; 1.72973; 2.54290; 2.79647];

sigma\_pico = [0.00510; 0.00575; 0.00555; 0.00597; 0.00683];

fprintf('=== VALIDACIÓN DE RESULTADOS ADC ===\n');

fprintf('Archivos: 1.txt, 2.txt, 3.txt, 4.txt, 5.txt\n\n');

% Crear tabla para resultados

resultados = table('Size',[5 5], 'VariableTypes',{'double','double','double','double','double'},...

'VariableNames',{'V\_in','Media\_Pico','Media\_MATLAB','Sigma\_Pico','Sigma\_MATLAB'},...

'RowNames',{'Test1','Test2','Test3','Test4','Test5'});

for i = 1:size(archivos, 1)

archivo\_actual = archivos{i,1};

voltaje\_aplicado = archivos{i,2};

fprintf('Procesando: %s (V\_in = %.2f V)\n', archivo\_actual, voltaje\_aplicado);

try

% Leer datos del archivo .txt

% Opción 1: Si el archivo tiene encabezado "Tiempo\_ms Voltaje\_V"

datos = readtable(archivo\_actual);

voltajes = datos.Voltaje\_V;

% Opción 2: Si el archivo solo tiene números (descomenta la línea siguiente)

% voltajes = load(archivo\_actual);

% Opción 3: Si tiene otra estructura (descomenta y adapta)

% datos = importdata(archivo\_actual);

% voltajes = datos.data(:,2); % si la columna 2 son voltajes

% Calcular estadísticas

media\_matlab = mean(voltajes);

sigma\_matlab = std(voltajes);

% Mostrar resultados en consola

fprintf('Media (Pico): %.5f V\n', medias\_pico(i));

fprintf('Media (MATLAB): %.5f V\n', media\_matlab);

fprintf('Desviación (Pico): %.5f V\n', sigma\_pico(i));

fprintf('Desviación (MATLAB): %.5f V\n', sigma\_matlab);

% Almacenar resultados en tabla

resultados.V\_in(i) = voltaje\_aplicado;

resultados.Media\_Pico(i) = medias\_pico(i);

resultados.Media\_MATLAB(i) = media\_matlab;

resultados.Sigma\_Pico(i) = sigma\_pico(i);

resultados.Sigma\_MATLAB(i) = sigma\_matlab;

fprintf('----------------------------------------\n');

catch ME

fprintf('❌ Error al procesar %s: %s\n\n', archivo\_actual, ME.message);

fprintf('Sugerencia: Verifica el formato del archivo.\n');

fprintf('Puede intentar usar: voltajes = load(''%s'');\n', archivo\_actual);

fprintf('----------------------------------------\n');

end

end

% Mostrar tabla completa de resultados

fprintf('\n=== TABLA COMPARATIVA DE RESULTADOS ===\n');

disp(resultados);

% Calcular diferencias

diferencias\_media = abs(resultados.Media\_Pico - resultados.Media\_MATLAB);

diferencias\_sigma = abs(resultados.Sigma\_Pico - resultados.Sigma\_MATLAB);

fprintf('\n=== ANÁLISIS DE DIFERENCIAS ===\n');

for i = 1:5

fprintf('Test %d: ΔMedia = %.6f V, ΔSigma = %.6f V\n',...

i, diferencias\_media(i), diferencias\_sigma(i));

end

fprintf('\nMáxima diferencia en medias: %.6f V\n', max(diferencias\_media));

fprintf('Máxima diferencia en desviaciones: %.6f V\n', max(diferencias\_sigma));

% Verificar coincidencia

tolerancia = 1e-5; % Tolerancia de 0.00001 V

if all(diferencias\_media < tolerancia) && all(diferencias\_sigma < tolerancia)

fprintf('\n✅ VALIDACIÓN EXITOSA: Los resultados coinciden perfectamente.\n');

fprintf(' El algoritmo de MicroPython es correcto.\n');

else

fprintf('\n❌ DISCREPANCIA ENCONTRADA: Los resultados difieren.\n');

fprintf(' Posibles causas:\n');

fprintf(' - Formato diferente en archivos .txt\n');

fprintf(' - Precisión numérica diferente\n');

fprintf(' - Error en los cálculos de MicroPython\n');

end

% Graficar comparación

figure('Name', 'Validación ADC - Resultados Comparativos', 'Position', [100, 100, 1200, 500]);

subplot(1,2,1);

bar(diferencias\_media \* 1000, 'FaceColor', [0.2 0.6 0.8]); % Convertir a mV

set(gca, 'XTickLabel', {'0V','1.1V','1.7V','2.5V','2.8V'});

xlabel('Voltaje de Entrada');

ylabel('Diferencia en Media [mV]');

title('Diferencias en Media (MATLAB vs Pico)');

grid on;

subplot(1,2,2);

bar(diferencias\_sigma \* 1000, 'FaceColor', [0.8 0.4 0.4]); % Convertir a mV

set(gca, 'XTickLabel', {'0V','1.1V','1.7V','2.5V','2.8V'});

xlabel('Voltaje de Entrada');

ylabel('Diferencia en Desviación [mV]');

title('Diferencias en Desviación Estándar');

grid on;

% Guardar resultados en archivo

writetable(resultados, 'resultados\_comparativos.csv');

fprintf('\nResultados guardados en: resultados\_comparativos.csv\n');

% Mostrar resumen final

fprintf('\n=== RESUMEN FINAL ===\n');

fprintf('Se procesaron %d archivos correctamente.\n', size(archivos, 1));

fprintf('Todos los resultados están dentro de la tolerancia de %.6f V.\n', tolerancia);







