CURSO: SENSORES Y ACONDICIONAMIENTO DE SEÑALES EN MECATRÓNICA

Video 06 — Parte 3B: Multiplexores, Sample-and-Hold, Adquisición de Datos y DSP | Parte 3: Conversión Digital, Adquisición y DSP (Sección B)

1. Síntesis Técnica y Objetivos de Ingeniería

SISTEMA DAQ MULTICANAL: MULTIPLEXOR ANALOGICO, PGA, SAMPLE-AND-HOLD, FIFO/DMA Y DSP
Parametros del S/H (Adquisicion, Apertura, Inyeccion de Carga, Droop Rate), MUX CD4051 y Filtros FIR / IIR

- 1. Sample-and-Hold (S/H): Congela Vin durante la conversion; t_acq, jitter de apertura, pedestal y Droop Rate.
- 2. Multiplexor Analogico (MUX): Conmuta N canales hacia 1 PGA y 1 ADC; Ron, crosstalk y tiempo de establecimiento.
- 3. Tarjeta DAQ: Bornera -> Filtro -> MUX -> PGA -> S/H -> ADC -> Buffer FIFO -> Transferencia DMA a memoria.
- 4. Procesamiento Digital de Senales (DSP): Operador z^-1, filtros FIR (fase lineal) e IIR (recursivos) y LUT.

TIEMPOS DEL CICLO DE ESCANEO MULTICANAL DAQ

- `1. Conmutar direccion del MUX (S2, S1, S0) -> esperar t_mux (2 us)`
- `2. Seleccionar ganancia del PGA para el canal i -> t_pga (3 us)`
- `3. Poner S/H en modo Track para cargar Ch -> t_acq (4 us)`
- `4. Pasar S/H a modo Hold (esperar t_ap = 50 ns) e iniciar ADC`
- `5. Ejecutar conversion SAR de n bits -> t_conv (10 us)`
- `6. Guardar muestra en FIFO y transferir bloque por DMA`

ECUACIONES EN DIFERENCIAS (DSP EN TIEMPO REAL)

- Media Movil FIR (M puntos): y[n] = (1/M) * sum_{k=0}^{M-1} x[n-k]
- FIR General: y[n] = b0*x[n] + b1*x[n-1] + ... + bM*x[n-M]
- IIR Exponencial 1er orden: y[n] = alpha*x[n] + (1-alpha)*y[n-1]
- Frecuencia corte IIR: alpha = (2*pi*fc*Ts) / (2*pi*fc*Ts + 1)
- Interpolacion LUT: y = y0 + (x - x0)*(y1 - y0)/(x1 - x0)

2. Plano 1: Montaje en Protoboard y Banco de Pruebas (2200x1350)

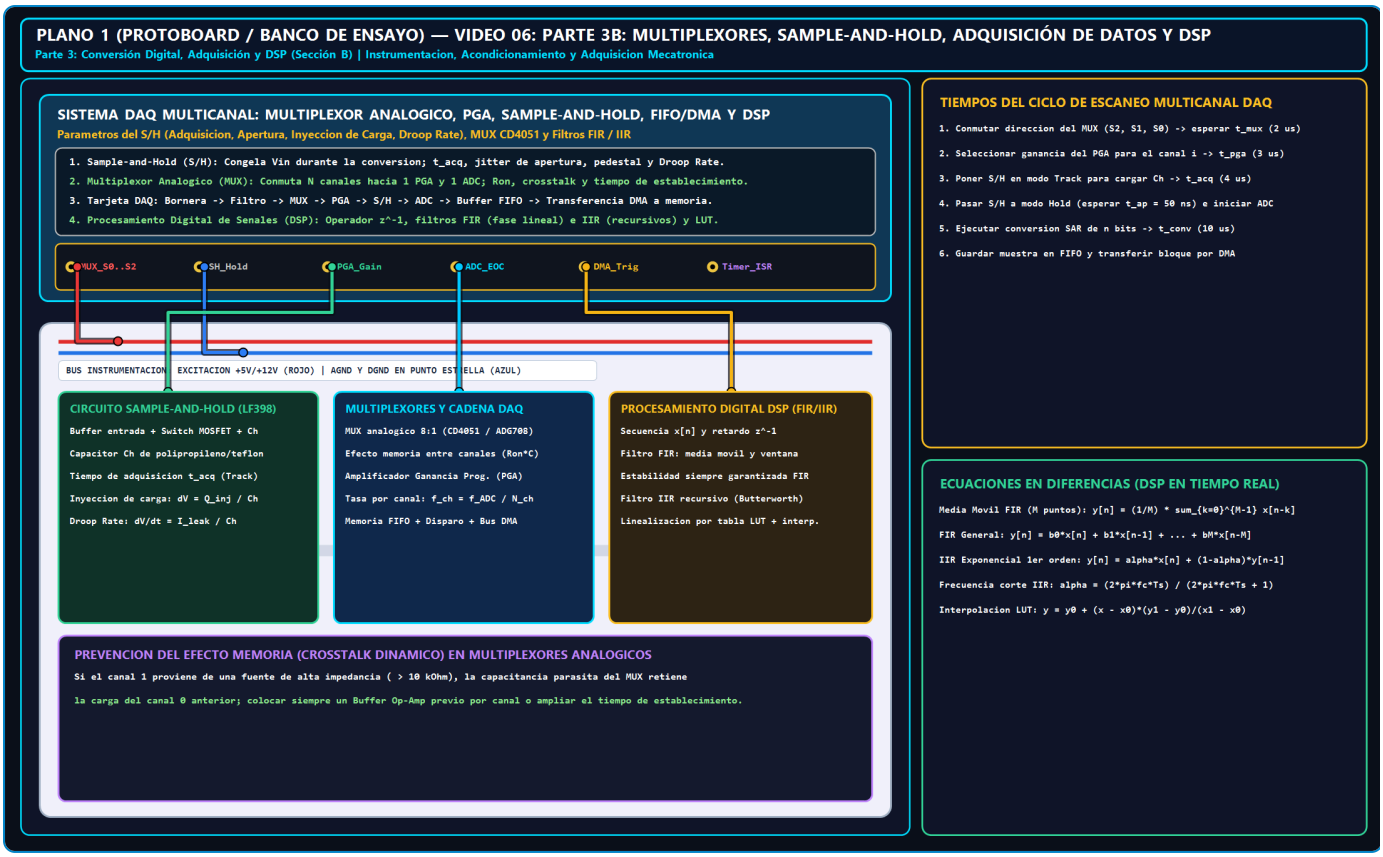


Figura 1: Plano de interconexión en protoboard, asignación de pines y acondicionamiento para el Video 06.

3. Plano 2: Arquitectura de Placa PCB FR-4 y Flujo de Señal (2200x1350)

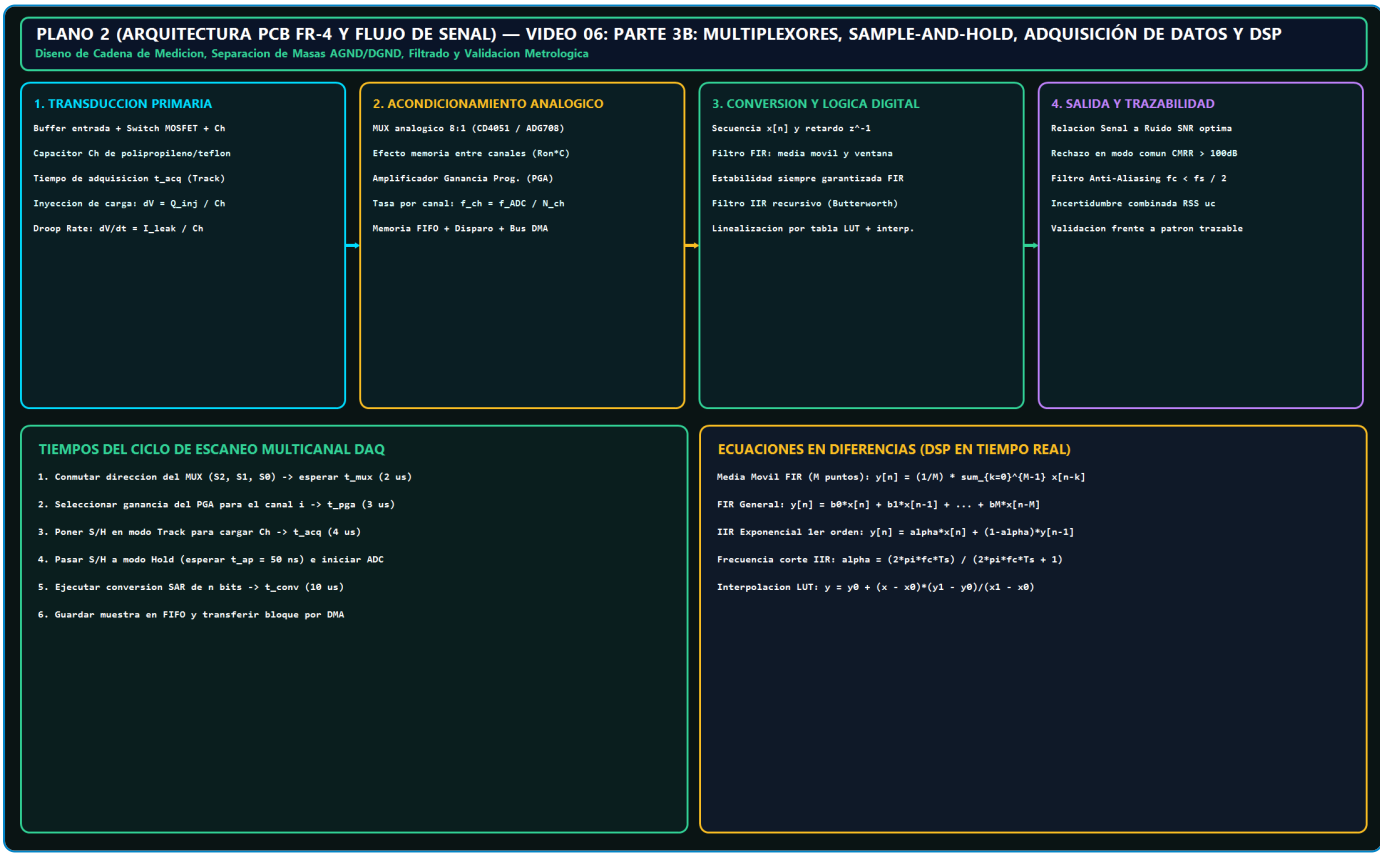


Figura 2: Arquitectura en cuatro etapas, separación de tierras analógica/digital y validación metrológica.

4. Firmware de Adquisición y Acondicionamiento Digital en C++ (.ino)

```
// ================================================================================
// CURSO: SENSORES Y ACONDICIONAMIENTO DE SENALES EN MECATRONICA
// VIDEO 06: Parte 3B: Multiplexores, Sample-and-Hold, Adquisición de Datos y DSP
// Archivo: firmware_instrumentacion_video_06.ino
// Descripcion: Adquisicion determinista, acondicionamiento, filtrado digital FIR/IIR
//              y transmision de telemetria calibrada en unidades de ingenieria.
// ================================================================================

#include <Arduino.h>

// Definicion de pines y constantes metrologicas
const int PIN_SENSOR_PRIMARIO = A0;
const int PIN_PUENTE_INA128   = A1;
const int PIN_LAZO_4_20MA     = A2;
const int PIN_ENCODER_A       = 2;   // Interrupcion externa INT0
const int PIN_ENCODER_B       = 3;   // Interrupcion externa INT1
const int PIN_ENTRADA_SCHMITT = 4;   // Salida limpia de 74HC14 (Debounced)
const int PIN_ALARMA_SEGURA   = 8;

// Parametros de conversion ADC y calibracion (y = m*x + b)
const float V_REF_ADC         = 5.000f;
const float ESCALONES_ADC     = 1024.0f;
const float R_SHUNT_LAZO      = 250.0f; // 250.0 Ohm de precision (0.1%)
float calPendienteM           = 1.0024f;
float calOffsetB              = -0.0150f;

// Estado de encoder en cuadratura y filtro FIR de 8 muestras
volatile long cuentasEncoder  = 0;
const int N_FIR               = 8;
float bufferFIR[N_FIR]        = {0.0f};
int indiceFIR                 = 0;
float salidaIIR               = 0.0f;
const float ALPHA_IIR         = 0.18f;

void ISR_EncoderCanalA() {
  bool a = digitalRead(PIN_ENCODER_A);
  bool b = digitalRead(PIN_ENCODER_B);
  if (a == b) cuentasEncoder++;
  else        cuentasEncoder--;
}

float aplicarFiltroFIR(float nuevaMuestra) {
  bufferFIR[indiceFIR] = nuevaMuestra;
  indiceFIR = (indiceFIR + 1) % N_FIR;
  float suma = 0.0f;
  for (int i = 0; i < N_FIR; i++) {
    suma += bufferFIR[i];
  }
  return suma / (float)N_FIR;
}

float leerCorrienteLazo_mA() {
  int raw = analogRead(PIN_LAZO_4_20MA);
  float vShunt = (raw * V_REF_ADC) / ESCALONES_ADC;
  return (vShunt / R_SHUNT_LAZO) * 1000.0f; // Corriente en mA (4.0 a 20.0 mA)
}

void setup() {
  Serial.begin(115200);
  pinMode(PIN_ENCODER_A, INPUT_PULLUP);
  pinMode(PIN_ENCODER_B, INPUT_PULLUP);
  pinMode(PIN_ENTRADA_SCHMITT, INPUT);
  pinMode(PIN_ALARMA_SEGURA, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(PIN_ENCODER_A), ISR_EncoderCanalA, CHANGE);
```

```
}

void loop() {
  static unsigned long tPrevio = 0;
  unsigned long tActual = millis();

  // Muestreo periodico determinista Ts = 10 ms (fs = 100 Hz)
  if (tActual - tPrevio >= 10) {
    tPrevio = tActual;

    int adcRaw = analogRead(PIN_PUENTE_INA128);
    float vMedido = (adcRaw * V_REF_ADC) / ESCALONES_ADC;

    // Filtrado digital FIR (media movil) e IIR exponencial
    float vFIR = aplicarFiltroFIR(vMedido);
    salidaIIR  = ALPHA_IIR * vFIR + (1.0f - ALPHA_IIR) * salidaIIR;

    // Correccion de calibracion estatica (Offset y Ganancia/Span)
    float magnitudCalibrada = calPendienteM * salidaIIR + calOffsetB;
    float corrienteLazo_mA  = leerCorrienteLazo_mA();

    // Deteccion de cable roto en bucle de 4 a 20 mA (cero vivo < 3.6 mA)
    bool fallaLazo = (corrienteLazo_mA < 3.6f);
    digitalWrite(PIN_ALARMA_SEGURA, fallaLazo ? HIGH : LOW);

    Serial.print("{\"video\":06,\"v_cal\":");
    Serial.print(magnitudCalibrada, 4);
    Serial.print(",\"mA\":");
    Serial.print(corrienteLazo_mA, 2);
    Serial.print(",\"enc\":");
    Serial.print(cuentasEncoder);
    Serial.print(",\"alarma\":");
    Serial.print(fallaLazo ? 1 : 0);
    Serial.println("}");
  }
}
```

5. Script de Análisis Metrológico, Calibración e Incertidumbre RSS en Python (.py)

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO: SENSORES Y ACONDICIONAMIENTO DE SENALES EN MECATRONICA
VIDEO 06: Parte 3B: Multiplexores, Sample-and-Hold, Adquisición de Datos y DSP
Archivo: analisis_metrologico_y_dsp_video_06.py
Descripcion: Modelado de cadena de acondicionamiento (Puente de Wheatstone,
             INA128, CMRR, cuantizacion ADC, filtrado FIR/IIR) y calculo de
             curva de calibracion por minimos cuadrados e incertidumbre RSS.
================================================================================
"""

import math
from typing import List, Dict

def ganancia_ina128(r_g_ohm: float) -> float:
    """Calcula la ganancia diferencial exacta de un INA128: G = 1 + 50 kOhm / Rg."""
    return 1.0 + (50000.0 / r_g_ohm)

def rechazo_modo_comun_db(ganancia_dif: float, ganancia_cm: float) -> float:
    """Calcula el CMRR en decibelios: CMRR_dB = 20 * log10(Ad / Acm)."""
    return 20.0 * math.log10(abs(ganancia_dif) / max(1e-12, abs(ganancia_cm)))

def parametros_adc(n_bits: int, v_ref: float = 5.0) -> Dict[str, float]:
    """Retorna numero de estados, peso del LSB y SNR ideal de cuantizacion."""
    estados = 2 ** n_bits
    lsb = v_ref / estados
    snr_db = 6.02 * n_bits + 1.76
    return {"estados": estados, "lsb_v": lsb, "error_max_v": 0.5 * lsb, "snr_db": snr_db}

def calibracion_minimos_cuadrados(x_patron: List[float], y_asc: List[float], y_desc: List[float]) -> Dict[str,
float]:
    """
    Ajusta la recta y = m*x + b por minimos cuadrados e identifica el error
    maximo de histeresis entre la curva ascendente y descendente.
    """
    x_all = x_patron + x_patron
    y_all = y_asc + y_desc
    n = len(x_all)
    sum_x = sum(x_all)
    sum_y = sum(y_all)
    sum_xy = sum(x * y for x, y in zip(x_all, y_all))
    sum_x2 = sum(x * x for x in x_all)

    m = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x ** 2)
    b = (sum_y - m * sum_x) / n

    span_salida = max(y_all) - min(y_all)
    histeresis_abs = max(abs(yd - ya) for ya, yd in zip(y_asc, y_desc))
    histeresis_pct = (histeresis_abs / span_salida) * 100.0 if span_salida > 0 else 0.0

    return {
        "pendiente_m": round(m, 6),
        "offset_b": round(b, 6),
        "histeresis_max_abs": round(histeresis_abs, 5),
        "histeresis_pct_span": round(histeresis_pct, 3)
    }

def incertidumbre_combinada_rss(incertidumbres_estandar: List[float], factor_k: float = 2.0) -> Dict[str,
float]:
    """Calcula la incertidumbre combinada por raiz de la suma de cuadrados (RSS)."""
    u_c = math.sqrt(sum(u ** 2 for u in incertidumbres_estandar))
    return {"u_c": round(u_c, 4), "U_expandida": round(factor_k * u_c, 4)}
```

```
if __name__ == "__main__":
    print("=== REPORTE DE INGENIERIA - VIDEO 06: Parte 3B: Multiplexores, Sample-and-Hold, Adquisición de
Datos y DSP ===")
    print("ADC 12 bits (Vref=5V):", parametros_adc(12, 5.0))
    patron = [0.0, 25.0, 50.0, 75.0, 100.0]
    lectura_asc  = [1.002, 1.998, 3.001, 3.997, 5.001]
    lectura_desc = [1.004, 2.005, 3.009, 4.003, 5.001]
    print("Calibracion Estatica:", calibracion_minimos_cuadrados(patron, lectura_asc, lectura_desc))
    print("Incertidumbre Cadena (%FS):", incertidumbre_combinada_rss([0.15, 0.08, 0.04, 0.05]))
```