CURSO DE PROGRAMACIÓN E INTERFAZ DE MICROCONTROLADORES PIC

Video 07 — Parte 7: Método para Diseñar un Sistema Basado en Microcontrolador · Plataforma Oficial: cursosmecatronica.com

Módulo: Parte 7 — Método para Diseñar un Sistema Basado en Microcontrolador
Enfoque de Ingeniería: Ingeniería Sistemática en 8 Pasos: Problema, Diagrama de Bloques, Presupuesto E/S, Selección, Esquemático, FSM, Prueba Modular e Integración

```
Pines_Req <= Pines_MCU | I_total_5V = Σ I_cargas < I_reg(max) | FSM: S(k+1) = f(S(k), Entradas)
```

1. Desarrollo Técnico Diapositiva por Diapositiva

Diapositiva 01: Por Qué Aplicar un Método Sistemático de Diseño con Microcontrolador

Superación de la Improvisación de Prueba y Error mediante Ingeniería Concurrente Hardware-Software

- **Turno 1 (Orus):** El error más costoso al emprender un proyecto con microcontroladores consiste en insertar inmediatamente componentes en la protoboard y ponerse a escribir código improvisado sin haber definido antes la arquitectura completa.
- **Turno 2 (Aoede):** Esa falta de método conduce invariablemente a quedarse sin pines a mitad del montaje, descubrir que la fuente se calienta, sufrir bloqueos por ruido o terminar con un programa confuso imposible de depurar.
- **Turno 3 (Orus):** Un sistema basado en microcontrolador exige ingeniería concurrente: cada decisión tomada en el circuito electrónico condiciona al software, y cada algoritmo elegido puede simplificar o sobrecargar el hardware externo.
- **Turno 4 (Aoede):** En esta séptima parte recorreremos paso a paso un método profesional de ocho etapas secuenciales que transforma cualquier necesidad mecatrónica en un prototipo confiable, documentado y fácil de mantener.

Diapositiva 02: Paso 1: Definir el Problema con Claridad y Especificaciones Cuantitativas

Qué Debe Hacer el Sistema, Rangos de Operación, Interfaz de Usuario, Ambiente y Modos de Falla Segura

- **Turno 1 (Orus):** El primer paso consiste en redactar en lenguaje claro y sin ambigüedades una descripción funcional completa del problema antes de mencionar marcas de circuitos integrados o instrucciones de programación.
- **Turno 2 (Aoede):** Convertimos los deseos vagos en números verificables: en lugar de decir controlar nivel de un tanque, especificamos mantener el nivel entre veinte y ochenta centímetros con alarma sonora si supera noventa centímetros.
- **Turno 3 (Orus):** En esta misma etapa definimos cómo interactuará el operador humano, qué información debe visualizarse, si el equipo se alimentará desde la red eléctrica o con baterías portátiles y bajo qué temperatura o humedad operará.
- **Turno 4 (Aoede):** Sobre todo, especificamos desde el día uno el comportamiento ante fallas: qué estado seguro deben adoptar todas las salidas si se desconecta un sensor, se agota la batería o se presiona parada de emergencia.

Diapositiva 03: Paso 2: Construcción del Diagrama de Bloques Funcional del Sistema

Partición Limpia en Sensores y Entradas, Unidad de Procesamiento, Actuadores, Interfaz y Fuente de Energía

- **Turno 1 (Orus):** En el segundo paso dibujamos un diagrama de bloques de alto nivel ubicando un rectángulo central de control rodeado por todas las entradas a la izquierda, las salidas a la derecha y la fuente de alimentación abajo.
- **Turno 2 (Aoede):** Cada flecha que entra o sale del bloque central debe etiquetarse con la naturaleza física de la señal: contacto seco, pulso de frecuencia, bus de cuatro bits, nivel analógico o conmutación de potencia.
- **Turno 3 (Orus):** Entre cada transductor externo y el bloque del microcontrolador dibujamos explícitamente el bloque acondicionador o excitador requerido, como optoacoplador de entrada, filtro antirrebote o etapa Darlington de salida.
- **Turno 4 (Aoede):** Este plano panorámico permite revisar con colegas o clientes que no falte ningún sensor límite, indicador de estado ni interruptor de modo manual antes de calcular un solo valor de resistencia.

Diapositiva 04: Paso 3: Presupuesto de Pines de Entrada/Salida, Memoria y Tiempos

Tabla Maestra de Señales, Demanda de Corriente Total y Verificación de Tiempos Críticos de Respuesta

- **Turno 1 (Orus):** El tercer paso traduce el diagrama de bloques en una tabla cuantitativa de recursos: contabilizamos exactamente cuántas entradas digitales, salidas digitales, líneas con interrupción o canales analógicos demanda el proyecto.
- **Turno 2 (Aoede):** Si la suma preliminar supera los trece pines de un integrado pequeño, evaluamos en este instante técnicas de ahorro como multiplexar teclado con pantalla L C D o emplear registros de desplazamiento seriales.
- **Turno 3 (Orus):** En la misma tabla sumamos el consumo en miliamperios de cinco voltios de todos los módulos: microcontrolador, retroiluminación de pantalla, optoacopladores y bobinas de relés para dimensionar la fuente sin sorpresas.
- **Turno 4 (Aoede):** Finalmente estimamos la velocidad temporal requerida: ¿basta revisar entradas cada diez milisegundos con cristal de cuatro megahercios, o existe algún pulso rápido de codificador que exija una interrupción dedicada?

Diapositiva 05: Paso 4: Selección del Microcontrolador y de los Componentes Periféricos

Adecuación del PIC16F84A frente a Modelos con ADC o UART Hardware y Revisión de Hojas de Datos

- **Turno 1 (Orus):** Con el presupuesto exacto en la mano ejecutamos el cuarto paso: elegir el modelo óptimo de microcontrolador y seleccionar con matrícula comercial cada sensor, pantalla, transistor, regulador y conector.
- **Turno 2 (Aoede):** Cuando la aplicación requiere hasta trece líneas digitales, sesenta y ocho bytes de variables y hasta mil instrucciones, el PIC dieciséis efe ochenta y cuatro A resulta ideal por su simplicidad y transparencia total.
- **Turno 3 (Orus):** Si en cambio el presupuesto del paso anterior exige entradas analógicas nativas de diez bits, comunicación serial por hardware o cuatro mil palabras de memoria, migramos sin cambiar de lenguaje a hermanos mayores como el dieciséis efe ochenta y ocho o dieciséis efe ochocientos setenta y siete A.
- **Turno 4 (Aoede):** En esta fase descargamos y verificamos las hojas de datos oficiales de todos los componentes elegidos para confirmar voltajes de trabajo, distribución física de terminales y corrientes máximas admisibles.

Diapositiva 06: Paso 5: Diseño del Diagrama Esquemático Completo y Asignación de Pines

Mapa Definitivo de Puertos, Respeto de Casos Especiales como RA4 y RB6/RB7 para ICSP

- **Turno 1 (Orus):** El quinto paso consiste en elaborar el diagrama esquemático eléctrico completo en software de captura electrónica, asignando con inteligencia cada señal a un terminal concreto del microcontrolador.
- **Turno 2 (Aoede):** Reservamos prioritariamente los pines con funciones únicas: el pin erre be cero para la interrupción externa más urgente, los pines superiores del puerto B si usamos resistencias pull up internas y cuidado especial con el drenador abierto de erre a cuatro.
- **Turno 3 (Orus):** Además dejamos ligeramente cargados o con puente de aislamiento los terminales erre be seis y erre be siete junto con el pin cuatro de reinicio para poder reprogramar en circuito sin desconectar periféricos.
- **Turno 4 (Aoede):** Todo buen esquemático numera cada componente como erre uno o ce uno, explicita todos los capacitores de desacoplo, diodos de protección en bobinas y bornes de conexión hacia el exterior de la tarjeta.

Diapositiva 07: Paso 6: Diseño de la Arquitectura del Software antes de Codificar

Diagramas de Flujo, Pseudocódigo Estructurado y Máquinas de Estados Finitos No Bloqueantes

- **Turno 1 (Orus):** Jamás abrimos el compilador sin cumplir el sexto paso: diseñar visualmente la lógica del programa mediante diagramas de flujo, pseudocódigo modular o diagramas de transición de una máquina de estados finitos.
- **Turno 2 (Aoede):** En sistemas secuenciales industriales modelar el proceso como estados bien definidos, tales como Reposo, Llenado, Mezclado, Vaciado y Alarma, mediante una estructura SELECT CASE evita enredos de banderas contradictorias.
- **Turno 3 (Orus):** Descomponemos el sistema en subrutinas cortas de responsabilidad única: una subrutina solo lee y filtra entradas, otra actualiza la máquina de estados, otra comanda salidas y otra refresca la pantalla L C D.
- **Turno 4 (Aoede):** Revisar sobre el papel las transiciones de estado permite detectar condiciones de bloqueo o combinaciones peligrosas de botones en pocos minutos, ahorrando horas enteras de frustración frente al hardware.

Diapositiva 08: Paso 7: Escritura Modular y Prueba Aislada de Cada Bloque de Código

Regla de Oro: Probar Cada Periférico por Separado antes de Unir Todo el Programa

- **Turno 1 (Orus):** En el séptimo paso pasamos a codificar en PicBasic Pro aplicando la estrategia incremental más efectiva de la ingeniería embebida: construir y probar un solo bloque periférico aislado a la vez.
- **Turno 2 (Aoede):** Primero grabamos un programa mínimo de diez líneas que únicamente hace parpadear un diodo testigo para certificar que la fuente, el reinicio, los fusibles, el cristal de cuatro megahercios y el grabador funcionan al cien por ciento.
- **Turno 3 (Orus):** Segundo, escribimos un pequeño programa de prueba exclusivo para mostrar Hola Mundo en la pantalla L C D; tercero, otro programa corto que solo lee el teclado y muestra cada tecla presionada en la pantalla ya verificada.
- **Turno 4 (Aoede):** Cuarto, probamos de forma individual el accionamiento seguro de cada relé o motor y la lectura de cada sensor: cuando cada ladrillo funciona sin fisuras por separado, unirlos deja de ser un misterio.

Diapositiva 09: Paso 8: Integración Total Hardware-Software, Pruebas Límite y Documentación

Ensamble de la Máquina de Estados Completa, Ensayos de Estrés, Ruido y Respaldo de Código

- **Turno 1 (Orus):** El octavo paso integra las subrutinas ya depuradas bajo el lazo coordinador principal y somete el prototipo completo a pruebas rigurosas contra todas las especificaciones definidas en el paso número uno.
- **Turno 2 (Aoede):** Un ingeniero no solo prueba el camino feliz donde el operador oprime todo en orden: provoca intencionalmente casos límite como pulsar varias teclas juntas, desconectar un sensor en marcha o cortar y restituir la energía.
- **Turno 3 (Orus):** Verificamos también con carga real conectada en los relés o motores que las conmutaciones electromagnéticas no alteren caracteres en la pantalla ni produzcan reinicios espurios en el microcontrolador.
- **Turno 4 (Aoede):** Finalmente cerramos el ciclo archivando el diagrama esquemático actualizado, la lista de materiales, el código fuente ampliamente comentado y el archivo hexadecimal exacto con su configuración de fusibles.

Diapositiva 10: Caso de Estudio Aplicado: Controlador Dosificador con Teclado, LCD y Relé

Recorrido Completo de los Ocho Pasos sobre el PIC16F84A con Memoria No Volátil EEPROM

- **Turno 1 (Orus):** Veamos cómo brillan los ocho pasos juntos en un dosificador industrial: el usuario ingresa por teclado la cantidad de piezas deseada, la guarda en EEPROM con WRITE, visualiza el conteo en pantalla y acciona una electroválvula por relé.
- **Turno 2 (Aoede):** El presupuesto asigna ocho pines del puerto B compartidos entre teclado y datos L C D, dos pines del puerto A para control de pantalla, erre a dos para el sensor óptico contador, erre a tres para el relé y erre b cero para parada de emergencia.
- **Turno 3 (Orus):** La máquina de estados opera en cuatro fases limpias: Configuración de meta, Dosificación activa contando flancos del sensor óptico, Meta alcanzada cerrando la válvula con aviso sonoro, y Parada inmediata por interrupción.
- **Turno 4 (Aoede):** Para que este y cualquier proyecto opere con fiabilidad industrial día tras día, en la Parte ocho dominaremos la depuración metódica, las fuentes de alimentación, las baterías y las reglas de prototipado.

2. Código Fuente en PicBasic Pro (picbasic_pro_video_07.pbp)

```
' ================================================================================
' CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
' Video 07 – Parte 7: Método para Diseñar un Sistema Basado en Microcontrolador
' Enfoque: Ingeniería Sistemática en 8 Pasos: Problema, Diagrama de Bloques, Presupuesto E/S, Selección,
Esquemático, FSM, Prueba Modular e Integración
' Ecuaciones Clave: Pines_Req <= Pines_MCU  |  I_total_5V = Σ I_cargas < I_reg(max)  |  FSM: S(k+1) = f(S(k),
Entradas)
' Microcontrolador Objetivo: PIC16F84A @ 4.000 MHz (1 ciclo = 1 us)
' ================================================================================
@ DEVICE pic16F84A, XT_OSC, WDT_OFF, PWRT_ON, PROTECT_OFF
Include "modedefs.bas"
DEFINE OSC 4

' --- Declaracion de Pines y Variables ---
EntradaSensor  VAR PORTA.0
SalidaRele     VAR PORTB.0
LedHeartbeat   VAR PORTB.7
TxDiagnostico  VAR PORTA.1
ContadorCiclos VAR WORD
Estado         VAR BYTE

' --- Inicializacion Segura de Puertos (Estado antes que Direccion TRIS) ---
PORTA = %00000000
PORTB = %00000000
OPTION_REG.7 = 0            ' Pull-ups internos activos en PORTB
TRISA = %00000001           ' RA0 entrada, resto salidas
TRISB = %00000000           ' PORTB como salidas digitales
ContadorCiclos = 0
Estado = 0

' --- Lazo Principal Determinista ---
LazoPrincipal:
    TOGGLE LedHeartbeat
    ContadorCiclos = ContadorCiclos + 1
    IF EntradaSensor = 0 THEN
        PAUSE 20            ' Filtro antirrebote 20 ms
        IF EntradaSensor = 0 THEN
            HIGH SalidaRele
            Estado = 1
        ENDIF
    ELSE
        LOW SalidaRele
        Estado = 0
    ENDIF
    SEROUT TxDiagnostico, T2400, ["V07 Ciclo=", #ContadorCiclos, " Est=", #Estado, 13, 10]
    PAUSE 100
    GOTO LazoPrincipal
    END
```

3. Referencia en Ensamblador MPASM (ensamblador_mpasm_video_07.asm)

```
; ==========================================================================
; CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
; Video 07 – Parte 7: Método para Diseñar un Sistema Basado en Microcontrolador – Referencia de Bajo Nivel MPASM
(PIC16F84A)
; ==========================================================================
    LIST        P=16F84A
    #INCLUDE    <P16F84A.INC>
    __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _CP_OFF

    CBLOCK  0x0C
        W_TEMP
        STATUS_TEMP
        CONTADOR
    ENDC

    ORG     0x0000
    GOTO    INICIO

    ORG     0x0004              ; Vector de Interrupcion Hardware
    MOVWF   W_TEMP              ; Salvado de contexto
    SWAPF   STATUS, W
    MOVWF   STATUS_TEMP
    BCF     INTCON, INTF        ; Limpiar bandera RB0/INT
    SWAPF   STATUS_TEMP, W
    MOVWF   STATUS
    SWAPF   W_TEMP, F
    SWAPF   W_TEMP, W
    RETFIE

INICIO:
    BCF     STATUS, RP0         ; Banco 0
    CLRF    PORTA
    CLRF    PORTB
    BSF     STATUS, RP0         ; Banco 1
    MOVLW   B'00000001'
    MOVWF   TRISA               ; RA0 entrada
    CLRF    TRISB               ; PORTB salidas
    BCF     STATUS, RP0         ; Regresar a Banco 0

BUCLE:
    BTFSS   PORTA, 0
    BSF     PORTB, 0
    BTFSC   PORTA, 0
    BCF     PORTB, 0
    GOTO    BUCLE
    END
```

4. Equivalencia en Lenguaje C XC8 (firmware_xc8_video_07.c)

```
/* ============================================================================
 * CURSO: PROGRAMACION E INTERFAZ DE MICROCONTROLADORES PIC (cursosmecatronica.com)
 * Video 07 – Parte 7: Método para Diseñar un Sistema Basado en Microcontrolador – Equivalencia en C Embebido
(XC8)
 * ============================================================================ */
#include <xc.h>
#define _XTAL_FREQ 4000000UL

#pragma config FOSC = XT, WDTE = OFF, PWRTE = ON, CP = OFF

void main(void) {
    PORTA = 0x00;
    PORTB = 0x00;
    OPTION_REGbits.nRBPU = 0;   /* Pull-ups internos activos en PORTB */
    TRISA = 0b00000001;         /* RA0 como entrada */
    TRISB = 0b00000000;         /* RB0..RB7 como salidas */

    while (1) {
        PORTBbits.RB7 ^= 1;     /* LED Heartbeat */
        if (PORTAbits.RA0 == 0) {
            __delay_ms(20);
            if (PORTAbits.RA0 == 0) {
                PORTBbits.RB0 = 1;
            }
        } else {
            PORTBbits.RB0 = 0;
        }
        __delay_ms(100);
    }
}
```