Desarrollo de software para sistemas empotrados

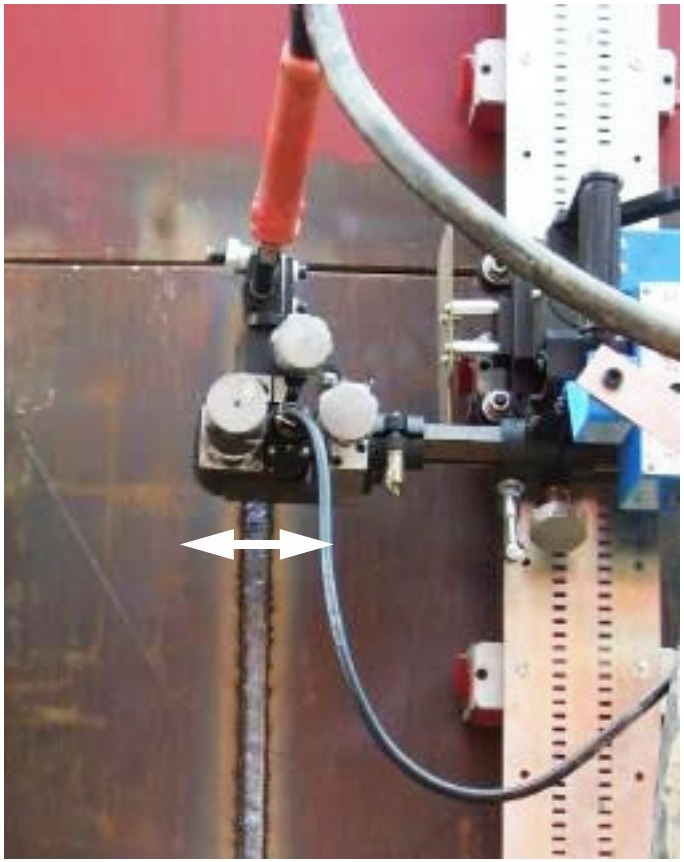
Examen Febrero 2020

Introducción

El objetivo de este ejercicio es realizar el análisis y diseño arquitectónico de un sistema de control de la oscilación de una antorcha de soldadura

En un sistema de soldadura es beneficioso hacer que la antorcha oscile en dirección perpendicular al avance de la soldadura, al objeto de ensanchar el área que se cubre

Paralelamente al movimiento de la antorcha es preciso mover el alimentador del hilo que aporta el material de soldadura, para que ambos se muevan sincronizadamente



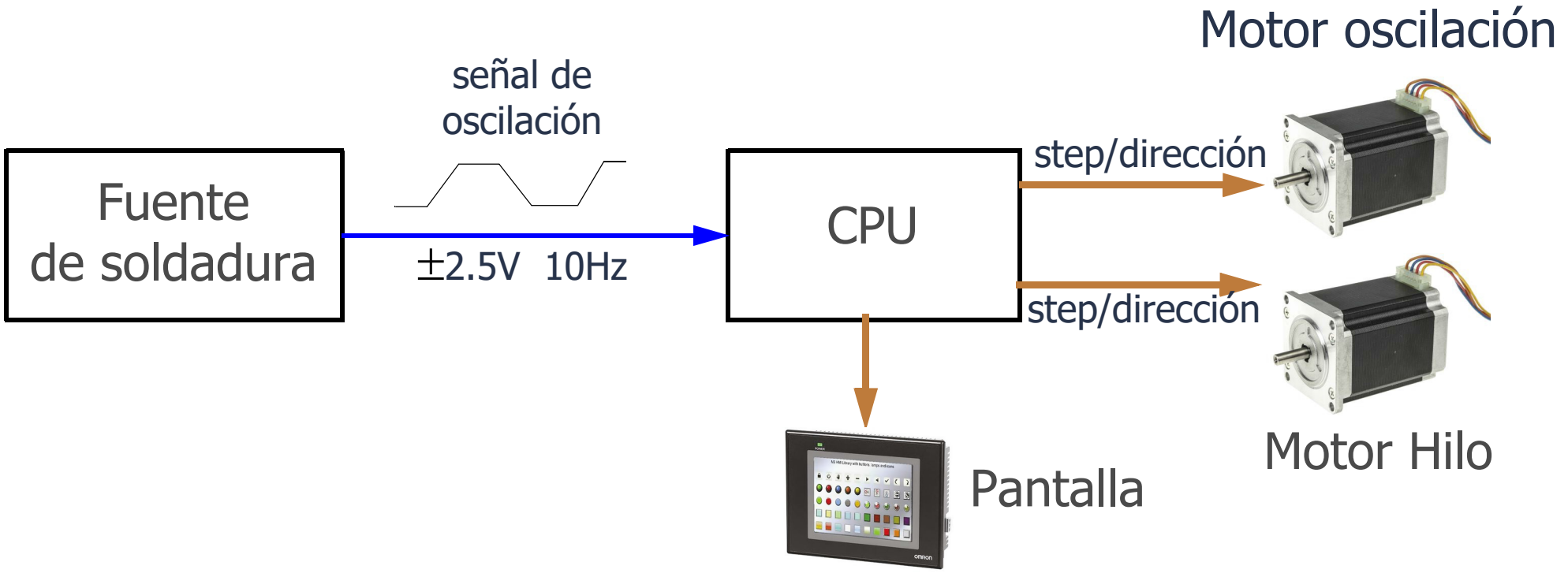
Sistema

La señal que determina la trayectoria de la oscilación es analógica y proviene de un subsistema llamado "fuente de soldadura"

El controlador se ejecuta en una CPU

- hace un seguimiento de la señal de la fuente y mueve los motores de la antorcha y del hilo

Plataforma



Los motores son paso a paso y requieren cada uno una señal de pulsos (llamada *step*) y una señal de dirección

- Por cada pulso el motor se mueve un paso en la dirección especificada

Software funcional

La funcionalidad del software está ya desarrollada en forma de 6 funciones que se ejecutan en la CPU

- `seguimiento`(): lee la señal de oscilación y escribe su valor en un *dato protegido* con exclusión mutua, llamando a `escribe_señal`()
- `servo_oscilación`(): lee el *dato protegido* mediante la operación `lee_señal`(), calcula y pone la señal de dirección, genera un pulso por la línea *step* del motor de oscilación y calcula en función de la velocidad deseada la próxima vez que esta función debe volver a ejecutarse para dar el siguiente paso
- `servo_hilo`(): lo mismo que la anterior, pero para el servo de hilo
- `reportero`(): lee el *dato protegido* y lo representa en la pantalla

Software funcional (cont.)

Hay dos funciones para la lectura/escritura del *dato protegido*

- `lee_señal`(): toma el mutex, lee el dato y libera el mutex
- `escribe_señal`(): toma el mutex, escribe el dato y libera el mutex

Requisitos funcionales

1. El sistema debe realizar el seguimiento de la señal de entrada de forma periódica, llamando a `seguimiento`()
2. El sistema debe invocar a las funciones `servo_oscilación`() y `servo_hilo`() cada vez que hay que producir un flanco de subida o de bajada en la salida *step* del servo respectivo (por tanto, de forma aperiódica)

Requisitos no funcionales

3. Las actividades periódicas tienen los siguientes periodos (T) y plazos (D)

Actividad	T (ms)	D (ms)
`seguimiento`	0.5	0.2
`reportero`	1000	500

4. Las actividades de `servo_hilo` y `servo_oscilación` son aperiódicas con los siguientes tiempos mínimos entre llegadas (T_{min}) y plazos (D)

Actividad	T_{min} (ms)	D (ms)
`servo_hilo`	1	1
`servo_oscilación`	1	1

5. El sistema representa información en la pantalla periódicamente, invocando a la función `reportero`()

Requisitos no funcionales

6. Los tiempos de ejecución de peor y mejor caso (C y C^b) medidos para las funciones software ya disponibles son los siguientes:

función	C (ms)	C^b(ms)
`seguimiento()`	0.1	0.04
`servo_oscilación()`	0.1	0.05
`servo_hilo()`	0.1	0.05
`reportero()`	3	2
`lee_señal()`	0.04	0.03
`escribe_señal()`	0.04	0.03

7. Los requisitos temporales deben validarse con un análisis de planificabilidad
8. Los modelos *temporal* y *arquitectónico* solo tendrán en cuenta las actividades en la CPU y el dato compartido. No se modelan los dispositivos como fuente de soldadura, motores o pantalla

Otros requisitos no funcionales

El dato compartido usa una política de techo inmediato de prioridad

Los desarrollos y el software básico estarán basados en una plataforma que dispone de una CPU con un sistema operativo gobernado por eventos, con prioridades fijas y las siguientes características

Propiedad	Valores
Rango de prioridades	1...32
Rango de prioridades de interrupción	32...32
Tiempo de cambio de contexto (ms)	0.002...0.003
Overhead de las interrupciones	0.002...0.002
Tipo de Temporizador	Alarm Clock
Overhead del temporizador (ms)	0.001...0.002

Ejercicios

1. Dentro del proceso de análisis de requisitos, generar uno o varios diagramas UCM para los requisitos del sistema, exceptuando los de la plataforma
2. Modelar con AADL una arquitectura para este sistema, con 4 flujos de eventos:
   - seguimiento de la señal de entrada
   - servo de oscilación
   - servo de hilo
   - reportero

Ejercicios (cont.)

3. Modelar con MAST la arquitectura del sistema para realizar un análisis de planificabilidad inicial
   - ¿Cuál es la asignación óptima de prioridades?
   - ¿Es planificable el sistema con esa asignación?
   - ¿Cuánto podríamos aumentar los tiempos de ejecución y seguir manteniendo el sistema planificable?
   - ¿Cuáles son los tiempos de bloqueo?
     - justifica los valores obtenidos

Entregar 4 ficheros

Un informe en pdf con:

- diagrama(s) UCM
- diagrama AADL de nivel de sistema, con el máximo nivel de detalle
- una captura de pantalla de los resultados de MAST
- las respuestas a las preguntas planteadas

Workspace de UCMNav comprimido

Workspace de OSATE comprimido

Ficheros del modelo MAST en un archivo comprimido