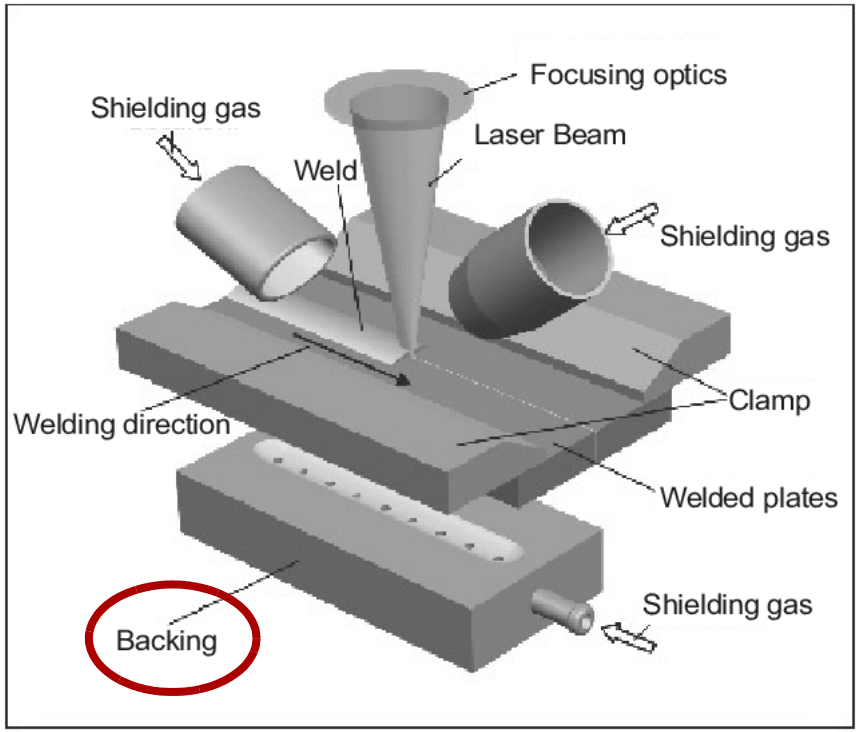
Desarrollo de software para sistemas empotrados

Examen Febrero 2019

Introducción

El objetivo de este ejercicio es realizar el análisis y diseño arquitectónico de un sistema de control del subsistema de gas de respaldo (*backing*) de un equipo de soldadura láser

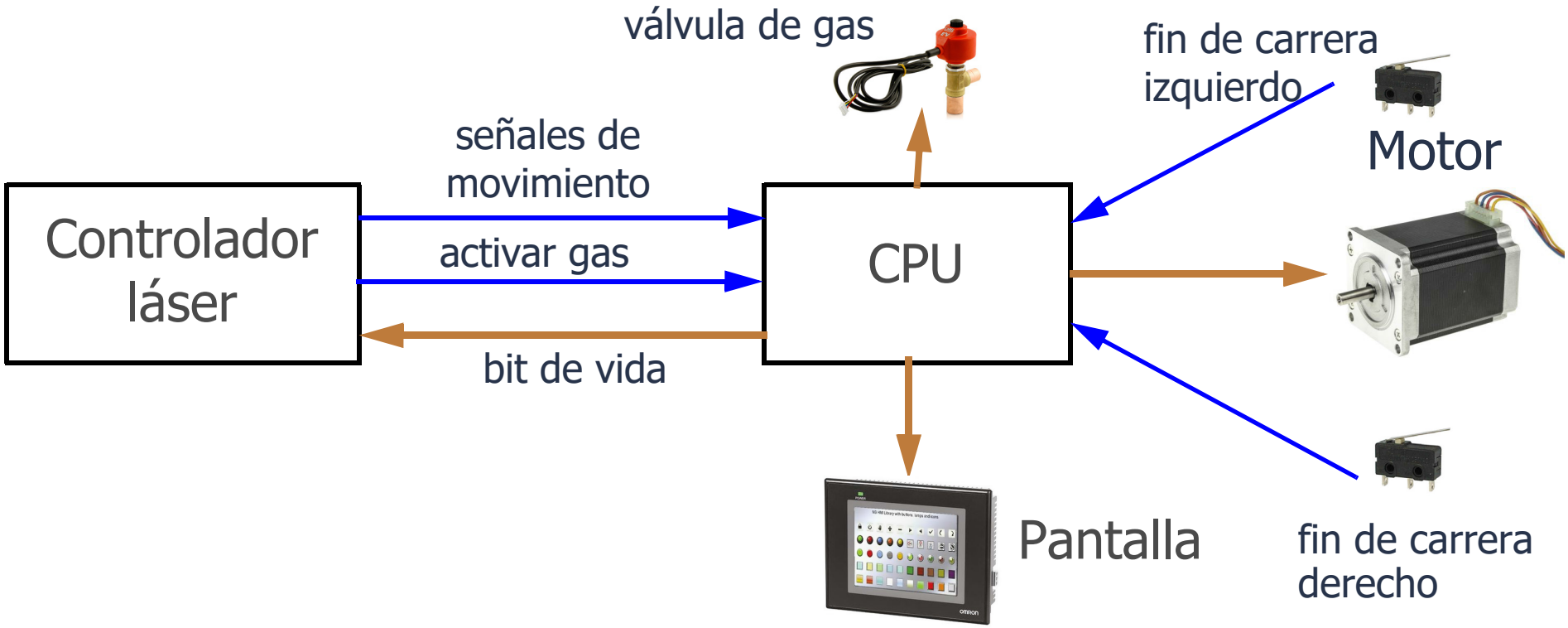


El gas de respaldo permite crear una atmósfera inerte en el lado opuesto al láser, con objeto de que el material no se oxide durante la soldadura

Sincronizadamente al movimiento lineal del láser es preciso mover el subsistema de gas de respaldo

El controlador que se pretende diseñar se comunica con señales digitales con el controlador del láser para realizar estos movimientos

Sistema



Todas las señales desde/hacia los periféricos son digitales, excepto la del motor que es analógica

- La CPU contiene una tarjeta de entrada/salida digital/analógica que lee y escribe estas señales

Descripción funcional del Software

El acceso a las entradas y salidas se hace desde una librería llamada `entrada_salida` que actúa como un recurso compartido con exclusión mutua y tiene las siguientes operaciones

- `lee_entradas_digitales`(): lee todas las entradas
- `actua_salida_digital`(): actúa sobre una salida digital concreta
- `actua_salida_analogica`(): actúa sobre la salida hacia el motor

La funcionalidad del software de aplicación está ya desarrollada y contiene

- un recurso compartido, con exclusión mutua, llamado `alarmas`
- y tres funciones que se ejecutarán periódicamente

Descripción funcional (cont.)

Recurso `alarmas`: Almacena los errores que se producen en el sistema. Dispone de dos operaciones:

- `lee_alarmas`(): lee el estado de todas las alarmas
- `escribe_alarma`(): escribe el estado de una alarma concreta

Funciones que se ejecutarán periódicamente:

- `estado`(): lee las alarmas y genera la señal digital de bit de vida
- `control`(): lee las señales digitales que provienen del controlador láser y de los finales de carrera y en función de estas señales genera la señal de actuación de la válvula de gas y la consigna de actuación analógica del motor; también escribe las alarmas
- `reportero`(): lee los datos protegidos (las entradas digitales y las alarmas) y los representa en la pantalla
  - la pantalla no necesita exclusión mutua pues solo la usa esta tarea

Requisitos no funcionales

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

Actividad	T (ms)	D (ms)
`estado()`	500	100
`control()`	50	10
`reportero()`	1000	500

Requisitos no funcionales (cont.)

2. 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)
`lee_entradas_digitales()`	0.2	0.1
`actua_salida_digital()`	0.1	0.05
`actua_salida_analogica()`	0.5	0.2
`lee_alarmas()`	0.8	0.7
`escribe_alarma()`	0.2	0.15
`estado()`	1.5	1.2
`control()`	3	2
`reportero()`	23	20

3. Los requisitos temporales deben validarse con un análisis de planificabilidad

Requisitos no funcionales (cont.)

4. Los modelos *temporal* y *arquitectónico* solo tendrán en cuenta las actividades en la CPU y los datos compartidos. No se modelan los dispositivos como el controlador de soldadura, motor, pantalla, ...
5. Sin embargo, el modelo de requisitos sí debe tener en cuenta estos dispositivos

Otros requisitos no funcionales

Los datos compartidos usan una política de techo inmediato de prioridad

Los desarrollos y el software básico estarán basados en la plataforma indicada arriba, disponiendo la CPU de 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
2. Modelar con AADL una arquitectura para este sistema, con 3 threads, correspondientes a cada función periódica:
   - estado
   - control
   - reportero
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

1. 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
2. Workspace de UCMNav comprimido
3. Workspace de OSATE comprimido
4. Ficheros del modelo MAST en un archivo comprimido