Desarrollo de software para sistemas empotrados

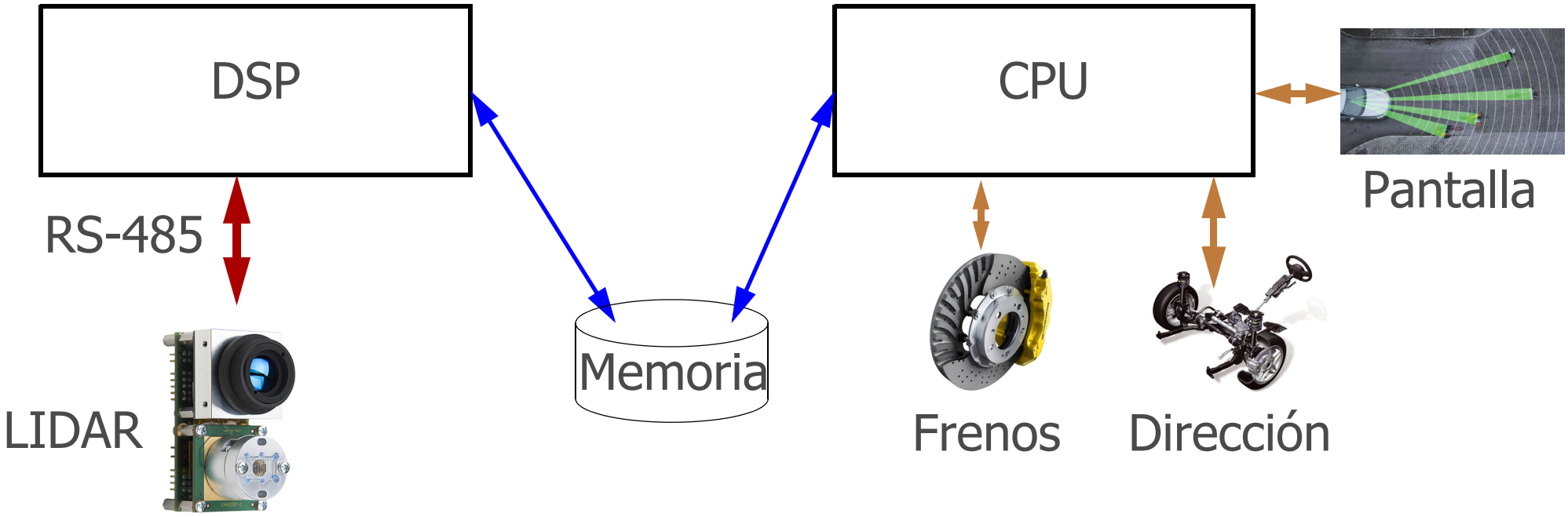
Examen Febrero 2018

Introducción

El objetivo de este ejercicio es desarrollar parte de un sistema de un automóvil autónomo utilizando un LIDAR (*Light Detection and Ranging*) como sensor para detectar el entorno y los posibles obstáculos

- El LIDAR, modelo Leddar Vu de LeddarTech tiene un alcance de hasta 180 m y cubre un ángulo horizontal de hasta 100º con tasas de refresco de datos de 100Hz
  - Transmite los datos obtenidos por un bus serie RS-485
- Un procesador digital de señal (DSP) recibe los datos del LIDAR y efectúa funciones de reconocimiento y seguimiento de objetos
  - Los resultados de estas funciones se depositan en una memoria compartida con otro procesador. Se utiliza un mutex para la sincronización de lectura y escritura de estos datos
- Una CPU convencional recoge los datos del seguimiento de objetos y ejecuta diversas operaciones concurrentes

Plataforma



La *memoria* conecta el DSP y la CPU

- contiene los datos compartidos, protegidos por un mutex

La línea serie RS-485 se utiliza para el envío de mensajes desde el LIDAR hacia el DSP

Software funcional

La funcionalidad del software está ya desarrollada en forma de 6 funciones

- una de ellas para funcionar en el DSP
  - `seguimiento`(): hace el procesamiento de los datos recibidos del LIDAR y deposita los resultados en la memoria
- tres para la CPU
  - `evitar_obstaculos`(): lee los datos de la memoria y realiza las acciones necesarias sobre la dirección para evitar los obstáculos
  - `representacion_grafica`(): lee los datos de la memoria y representa los obstáculos detectados en la pantalla
  - `frenada_emergencia`(): lee los datos de la memoria y si es necesario realiza las acciones requeridas sobre los frenos
- dos para la lectura/escritura de datos compartidos
  - `lee_datos`(): toma el mutex, lee los datos y libera el mutex
  - `deposita_datos`(): toma el mutex, escribe datos y libera el mutex

Requisitos funcionales

1. El LIDAR recoge los datos y los envía de forma aperiódica por el bus serie
2. El DSP dispone de un thread que invoca a `seguimiento`() cada vez que recibe datos del LIDAR (por tanto, de forma aperiódica)
3. La CPU dispone de tres threads que se activan de forma periódica y a cada periodo invocan, respectivamente, a las funciones
   - `evitar_obstaculos`()
   - `representacion_grafica`()
   - `frenada_emergencia`()

Requisitos temporales (no funcionales)

1. El LIDAR produce datos con un tiempo mínimo entre cada uno de 10ms
2. El plazo del thread de seguimiento es de 10ms tras recibir el dato
3. Los threads periódicos tienen los siguientes periodos y plazos

Thread	T (ms)	D (ms)
`evitar_obstaculos`	20	20
`representacion_grafica`	1000	500
`frenada_emergencia`	10	5

4. El dato compartido usa una política de techo inmediato de prioridad
   - al ser un recurso compartido entre dos procesadores diferentes se usará un techo de prioridad igual a la máxima prioridad del sistema

Requisitos temporales (no funcionales)

5. 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()`	2	0.8
`evitar_obstaculos()`	8	3
`representacion_grafica()`	37	12
`frenada_emergencia()`	3	2
`lee_datos()`	0.4	0.3
`deposita_datos()`	0.4	0.3

6. Estos requisitos deben validarse con un análisis de planificabilidad
7. El modelo temporal solo tendrá en cuenta las actividades en los procesadores y el dato compartido
   - No se modelan el LIDAR, ni la línea RS-485, ni los dispositivos como frenos, dirección o pantalla

Otros requisitos no funcionales

Los desarrollos y el software básico estarán basados en la plataforma indicada arriba disponiendo tanto el DSP como la CPU de sendos sistemas operativos gobernado por eventos, con prioridades fijas y las siguientes características

Procesador	DSP	CPU
Rango de prioridades	1...32	1...32
Rango de prioridades de interrupción	32...32	32...32
Tiempo de cambio de contexto (ms)	0.6...0.8	0.005...0.007
Overhead de las interrupciones	0.1...0.1	0.002...0.002
Tipo de Temporizador	-	Alarm Clock
Overhead del temporizador (ms)	-	0.002...0.003

Ejercicios

1. Dentro del proceso de análisis de requisitos, generar 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:

a) seguimiento de obstáculos, con:
   - el LIDAR, capaz de enviar datos de forma aperiódica
   - el mensaje con el dato enviado por la línea serie RS-485
   - un thread para el seguimiento, activado por la llegada de cada dato

b) evitar obstáculos, con un thread periódico que actúa sobre la dirección

c) representación gráfica, con un thread periódico que actúa sobre la pantalla

d) frenada de emergencia, con un thread periódico que actúa sobre los frenos

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?
     - en la CPU
     - en el DSP
   - ¿Cuáles son los tiempos de bloqueo?
     - ¿se te ocurre una explicación para estos valores?

Entregar

Un informe en pdf con:

- diagrama(s) UCM
- diagrama AADL de nivel de sistema
- una captura de pantalla de los resultados de MAST y las respuestas a las preguntas planteadas

Workspace de UCMNav comprimido

Workspace de OSATE comprimido

Ficheros del modelo MAST en un archivo comprimido