Prima prova pratica in itinere: diagnostic voter

Corso di
Progetto e Sviluppo di
Sistemi in Tempo Reale
a.a. 2020/21

Marcello Cinque

Traccia

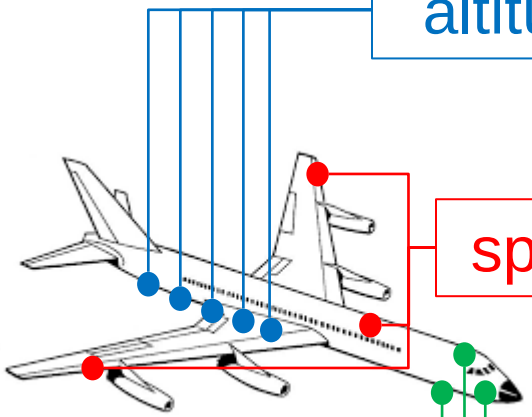
- Si realizzino in RTAI tre task hard real time kernel-level in grado di correggere, attraverso *voting* a maggioranza, gli errori di un'ipotetica diagnostica ridondante di un velivolo che riporta informazioni su:
  - Altitudine (5 sensori, aggiornati ogni 250ms)
  - Velocità (3 sensori, aggiornati ogni 500ms)
  - Temperatura (3 sensori, aggiornati ogni secondo)
- Ogni task kernel è dedicato alla correzione di una grandezza, e i periodi vanno scelti conformemente alla frequenza di aggiornamento della relativa grandezza
- I task kernel devono essere assegnati tutti alla stessa CPU e schedulati con RM

Traccia

- I sensori sono emulati da altrettanti task realtime (sorgente “sensor.c”) forniti insieme alla traccia.
- I task producono le misure dei sensori e li memorizzano in un’apposita struct in area di memoria condivisa

```
struct raw_sensors_data
{
    unsigned int altitudes[ALTITUDE_SIZE];
    unsigned int speeds[SPEED_SIZE];
    int temperatures[TEMP_SIZE];
};
```

Schema



altitudes

speeds

temperatures

τ_1

τ_2

τ_3

altitude

speed

temperature

sensor.c

voter.c

Failure modes

- Si ipotizzano i seguenti modi di fallimento:
  - Temperatures: un sensore su 3 può esibire nel tempo un comportamento di tipo stuck-at-zero (valore = 0). Due misure su tre sono sempre corrette.
  - Speeds: un sensore su 3 può assumere nel tempo un valore casuale diverso dagli altri due. Due misure su tre sono sempre corrette
  - Altitudes: un sensore su 5 può esibire nel tempo un comportamento di tipo stuck-at-zero (valore = 0) e un altro sensore su cinque può esibire nel tempo un valore casuale diverso dagli altri. Tre misure su cinque sono sempre corrette.

I task

- Ogni task (ciascuno per la sua grandezza) periodicamente legge i valori dal rispettivo array (temperatures, speeds, altitudes) e determina il valore corretto, a maggioranza.
- Il valore corretto, in ogni periodo, viene depositato nella struttura dati condivisa:

```
struct processed_sensors_data
{
    unsigned int altitude;
    unsigned int speed;
    int temperature;
};
```

Opzionale: slack e overrun

- Per ogni task di livello kernel si misurino lo slack e/o si determini se il task è in overrun attraverso il valore di ritorno della primitiva `rt_wait_next_period`.
- In caso di overrun o slack negativo, ci si limiti a stampare un messaggio nel log di sistema con `print_k`

Opzionale: attivazione dei task tramite parametri

- Si consenta la scelta dei task da attivare attraverso passaggio di tre parametri interi al modulo:
  - temperature: se 0 il relativo task non va attivato, 1 altrimenti
  - speed: se 0 il relativo task non va attivato, 1 altrimenti
  - altitude: se 0 il relativo task non va attivato, 1 altrimenti
- Per defalut, tutti e tre i task sono attivati. L'utente potrà eventualmente selezionare i task da non attivare all'atto dell'inserimento del modulo:
  - Ad es., con:

    ```
    insmod voter_rt.ko temperature=0 speed=0
    ```
  - sarà attivato solo il task per l'altitudine