Prima prova pratica in itinere: Real-time filtering

Corso di
Progetto e Sviluppo di
Sistemi in Tempo Reale
a.a. 2023/24

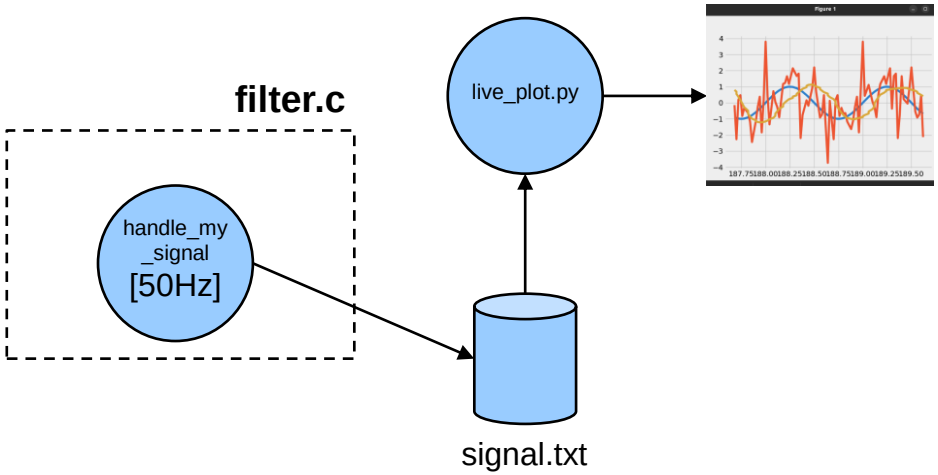
Marcello Cinque
Daniele Ottaviano

Codice di partenza

- Una tipica applicazione di digital signal processing real-time acquisisce un segnale affetto da rumore e lo elabora in più stadi:
  - Acquisizione (generatore del segnale)
  - Filtraggio
  - Storing su file
  - Visualizzazione
- Nel codice di partenza*, i primi tre stadi sono fusi in un'unica funzione, implementata come handler di un timer periodico
- Il quarto stadio è realizzato da uno script python

*Credit per il codice di partenza: R. Mancuso, Boston University, USA

Schema del codice di partenza



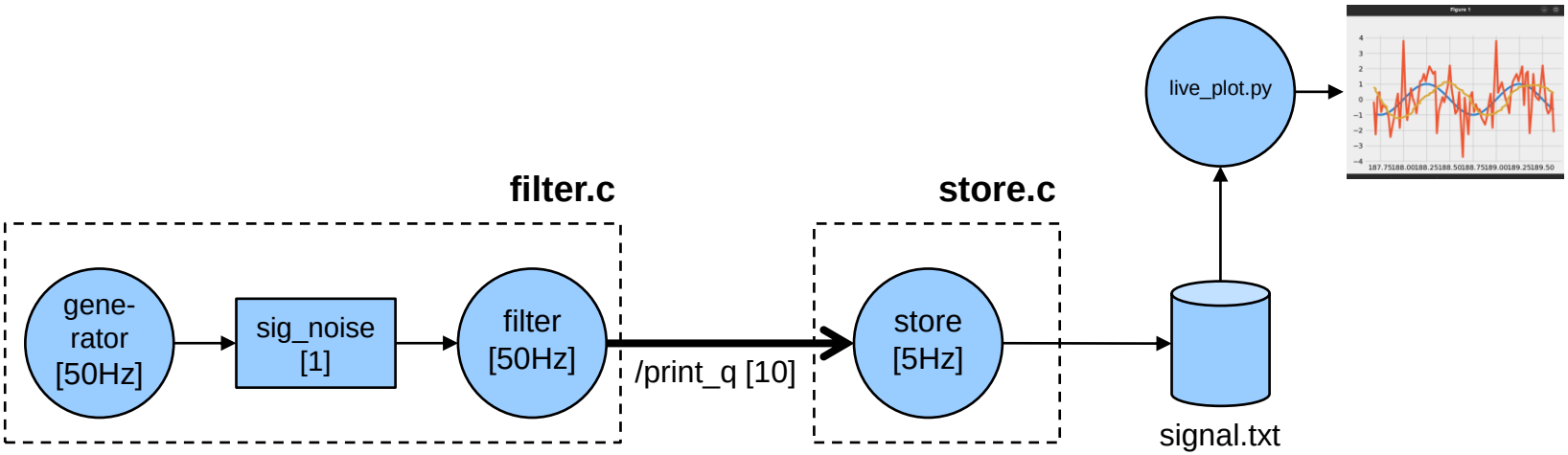
Codice di partenza

- Il codice di partenza già contiene le implementazioni
  - del generatore dell'onda sinusoidale con aggiunta di rumore
  - di due filtri: uno a media mobile e un butterworth del 2 ordine
  - dello storing su file, configurabile da linea di comando
- Il codice di partenza può essere interamente riutilizzato

Traccia (parte 1)

- Il codice di partenza va ristrutturato realizzando i tre stadi di generazione, filtraggio e storing in altrettanti task periodici
  - I task **generator** e **filter** vanno realizzati come thread periodici a frequenza 50Hz, in un sorgente filter.c; i due task comunicano attraverso una variabile condivisa acceduta in **mutua esclusione**
  - Il task **store**, periodico con frequenza 5Hz va realizzato all'interno di un sorgente a parte, store.c
  - filter e store comunicano attraverso una **coda di messaggi** di 10 elementi: ad ogni periodo, store preleva 10 elementi dalla coda e li scrive sul file signal.txt
  - I task sono realizzati riutilizzando il codice di partenza

Schema da realizzare (parte 1)

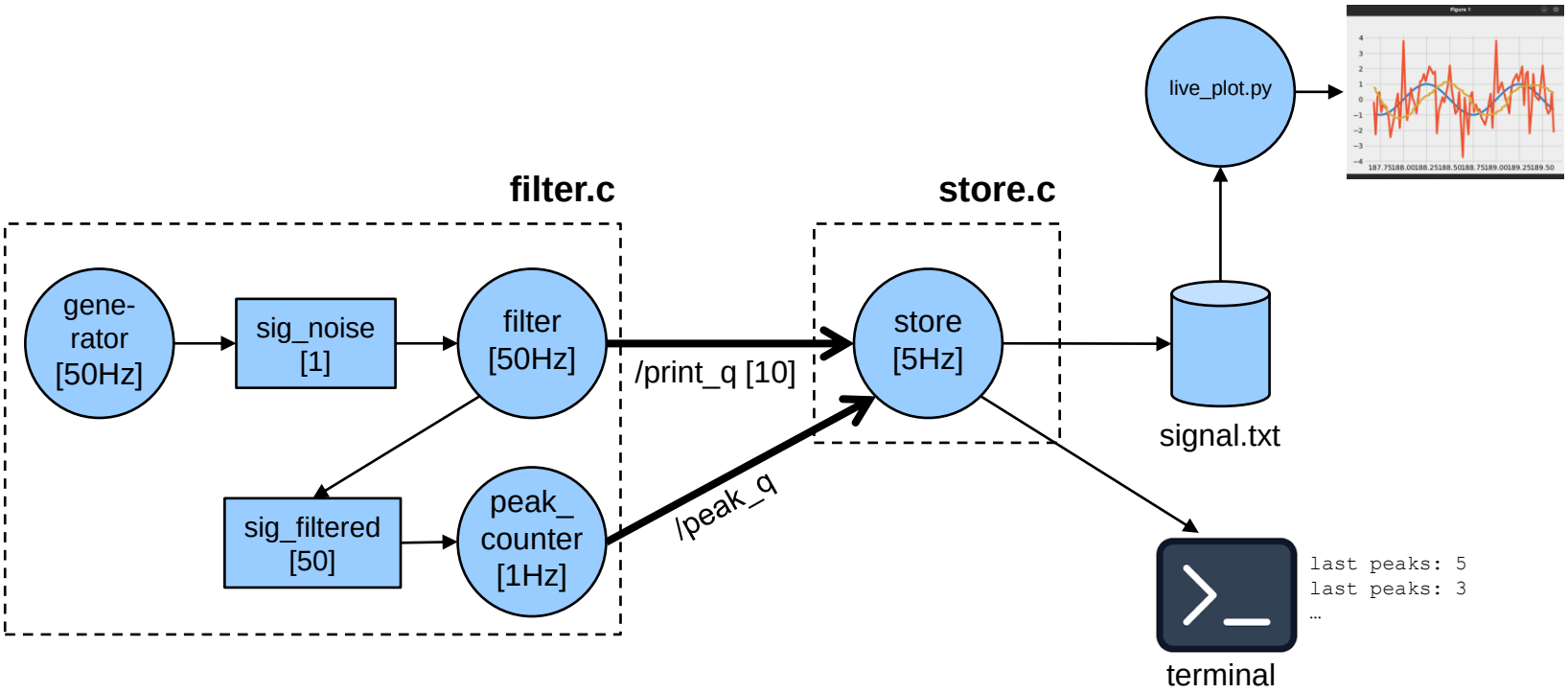


Traccia (parte 2)

- Aggiungere un quarto task periodico **peak_counter**, a frequenza 1Hz, il quale ad ogni ciclo analizza gli ultimi 50 campioni del segnale filtrato per cercare i «picchi», ossia i minimi o massimi locali
  - Un buon filtro dovrebbe generare meno picchi
- Il peak_counter va realizzato come thread in filter.c, e comunica con il filter attraverso un buffer condiviso `sig_filtered` di 50 elementi che deve essere acceduto in **mutua esclusione** evitando fenomeni di **inversione di priorità**
- Il numero di picchi conteggiati ogni periodo va inviato al task store tramite una coda; quando ricevuto, store li scrive a terminale:

```
last peaks: 5
last peaks: 3
…
```

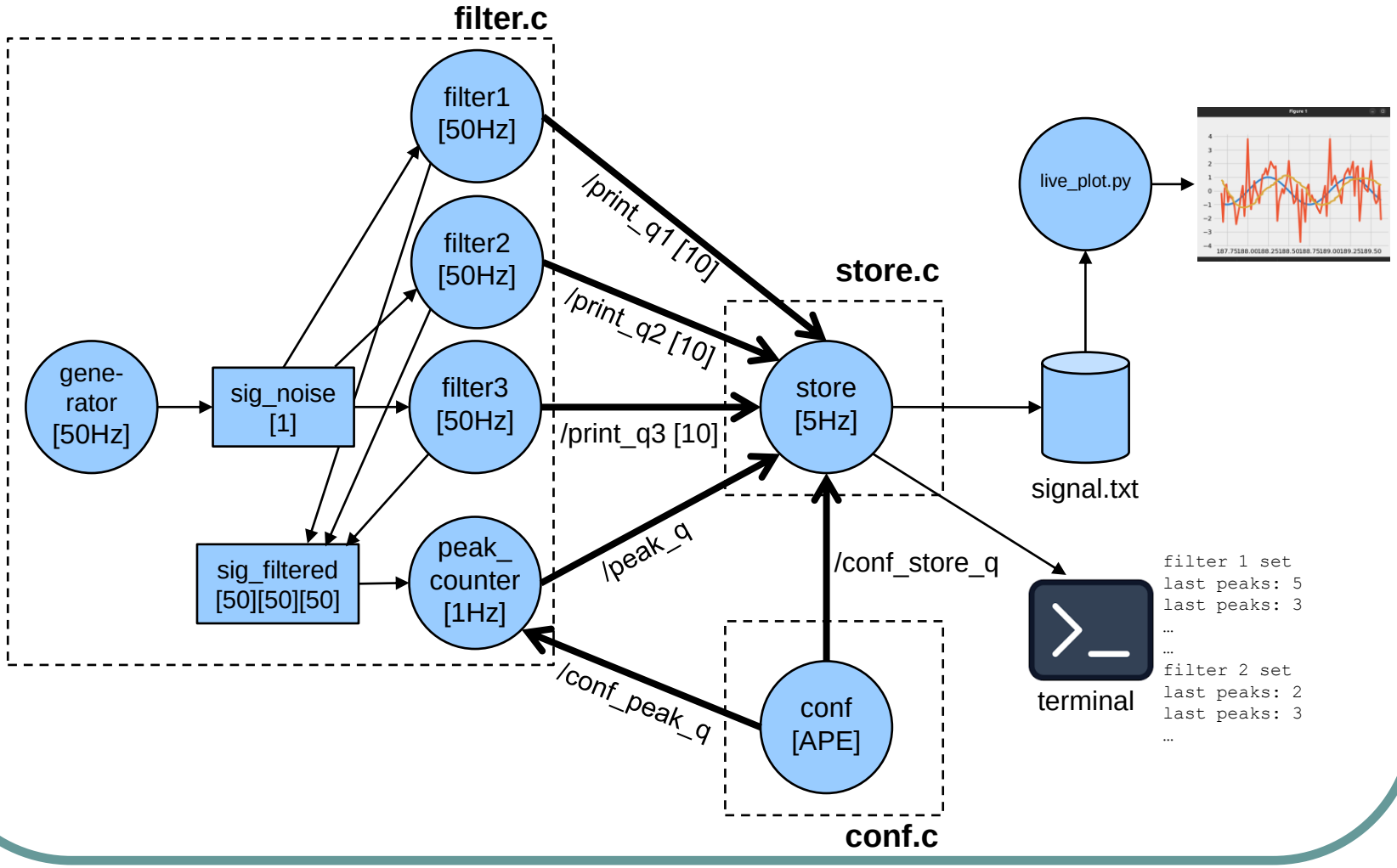
Schema da realizzare (parte 2)



Traccia (parte 3)

- Al fine di confrontare le prestazioni di filtri diversi, è possibile far elaborare il filtraggio contemporaneamente da più task
- Ad es: 3 filtri in contemporanea
  - 1 filtro a media mobile
  - 1 filtro butterworth del 2 ordine
  - 1 filtro a scelta (opzionale)
- I filtri scrivono tutti nella struttura `sig_filtered` (che conterrà tanti array quanti sono i filtri). La struttura nella sua interezza deve essere acceduta in mutua esclusione evitando priority inversion
- Un ulteriore task **conf** aperiodico invia ai task store e peak_counter, tramite due code, l'indice del filtro da usare (ad es. 1, 2 o 3)
  - Se nel ciclo corrente peak_counter riceve un nuovo indice da conf, cambia l'array di `sig_filtered` su cui contare i picchi. Di default usa l'array 1.
  - Se nel ciclo corrente store riceve un nuovo indice da conf, cambia la coda da cui prelevare il segnale filtrato. Di default usa la /print_q1.

Schema da realizzare (parte 3)



Note

- Tutti i task periodici devono essere schedulati con Rate Monotonic
- E' opportuno dunque che eseguano tutti sulla stessa CPU (impostando l'affinity o utilizzando il comando taskset)

- Suggerimento: per l'implementazione dei thread periodici si consiglia di utilizzare la libreria rt-lib presente nell'esercitazione E07-Controller-ipc

Nota su python

- Lo script live-plot.py necessita l'interprete python e la libreria per i plot, che possono essere installati con:

```
sudo apt install python3
sudo apt install python3-pip
pip install matplotlib
```

Consegna

- Entro le 20:00 del 10 maggio 2024
- E’ possibile lavorare in gruppi di massimo 4 persone
- Ogni studente deve consegnare individualmente l’elaborato su Teams, aggiungendo, se ha lavorato in gruppo, un file di testo con i nominativi e matricole dei componenti del gruppo
- I file consegnati saranno soggetti a controlli di differenza: se due gruppi consegnano lo stesso elaborato, sarà annullato per tutti i componenti di entrambi i gruppi