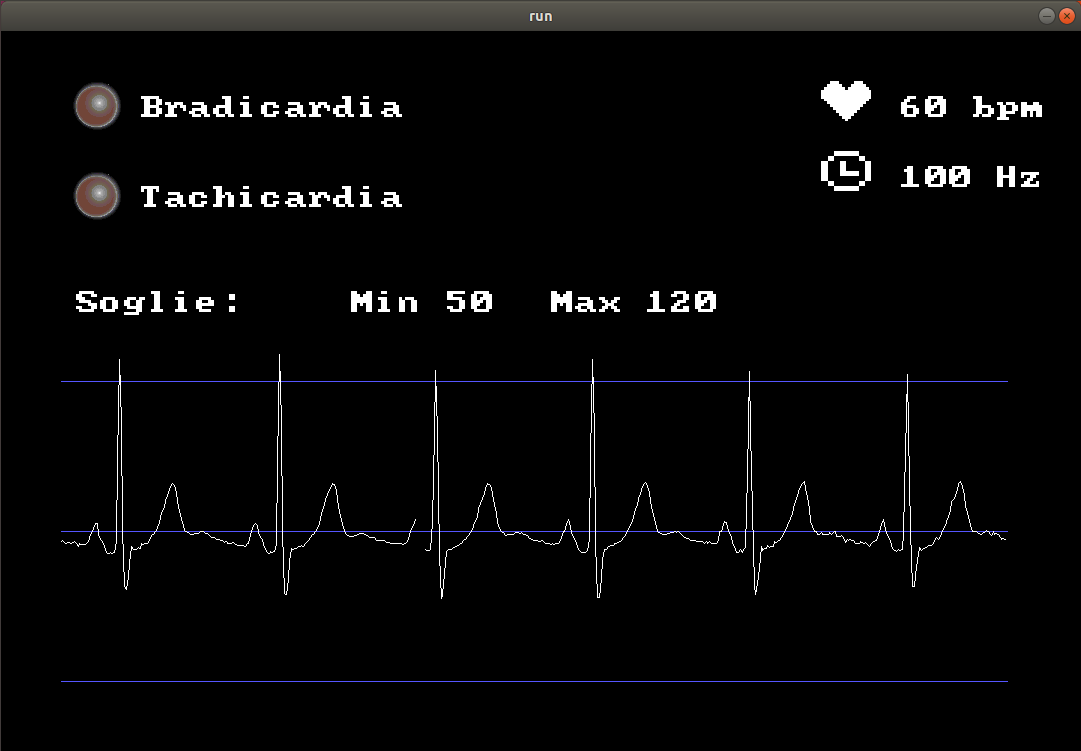
Informatica e Sistemi in tempo reale

Visualizzatore ECG



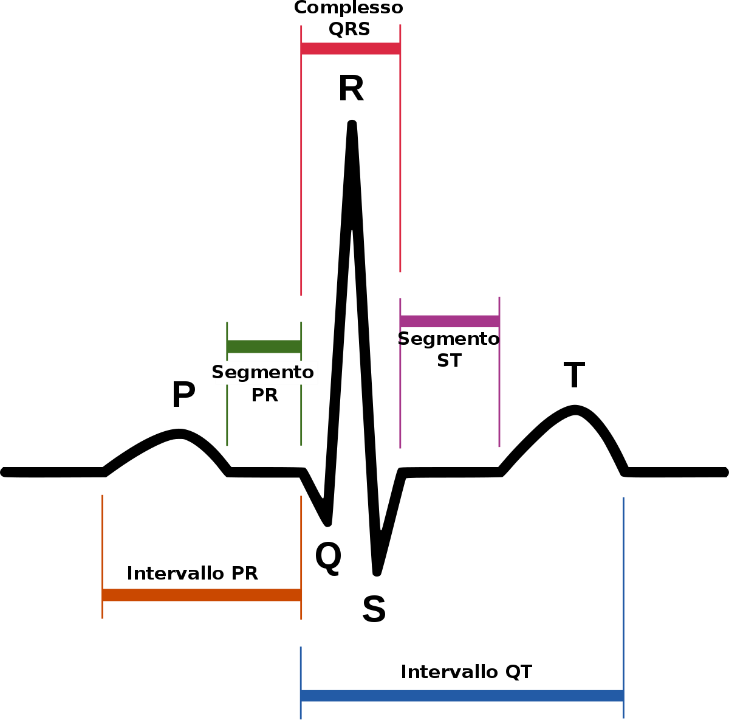
Data consegna Alessia Guida 522181 [alessiaguida6@gmail.com](mailto:alessiaguida6@gmail.com)

20/04/2018 Lorenzo Irace 501354 [irace.lorenzo@gmail.com](mailto:irace.lorenzo@gmail.com)

# Descrizione generale

L’applicazione realizzata si occupa di riprodurre un elettrocardiogramma prelevando i dati da un file e di rilevare la frequenza al fine di segnalare eventuali anomalie.

Il segnale ECG rappresenta le correnti rilevate sulla superficie del corpo umano generate dall’attività cardiaca. Ne risulta una struttura pseudoperiodica, come si può vedere in figura.



Il calcolo della frequenza cardiaca può essere effettuato tramite l’identificazione dei complessi QRS che corrispondono ai battiti. Essi sono dati dalla somma di tre onde:

* Onda Q: è negativa e di piccole dimensioni, e corrisponde alla depolarizzazione del setto interventricolare;
* Onda R: è un picco molto alto positivo, e corrisponde alla depolarizzazione dell’apice del ventricolo sinistro;
* Onda S: è un’onda negativa anch’essa di piccole dimensioni, e corrisponde alla depolarizzazione delle regioni basale e posteriore del ventricolo sinistro.

Per riconoscere tali complessi si utilizzano algoritmi basati sul calcolo delle derivate. La nostra scelta è ricaduta sull’algoritmo di Ahlstrom-Tompkins:

1. Calcolo del modulo della derivata prima
2. Filtraggio
3. Calcolo del modulo della derivata seconda
4. Calcolo della somma del modulo della derivata prima filtrato e del modulo della derivata seconda
5. Calcolo delle soglie

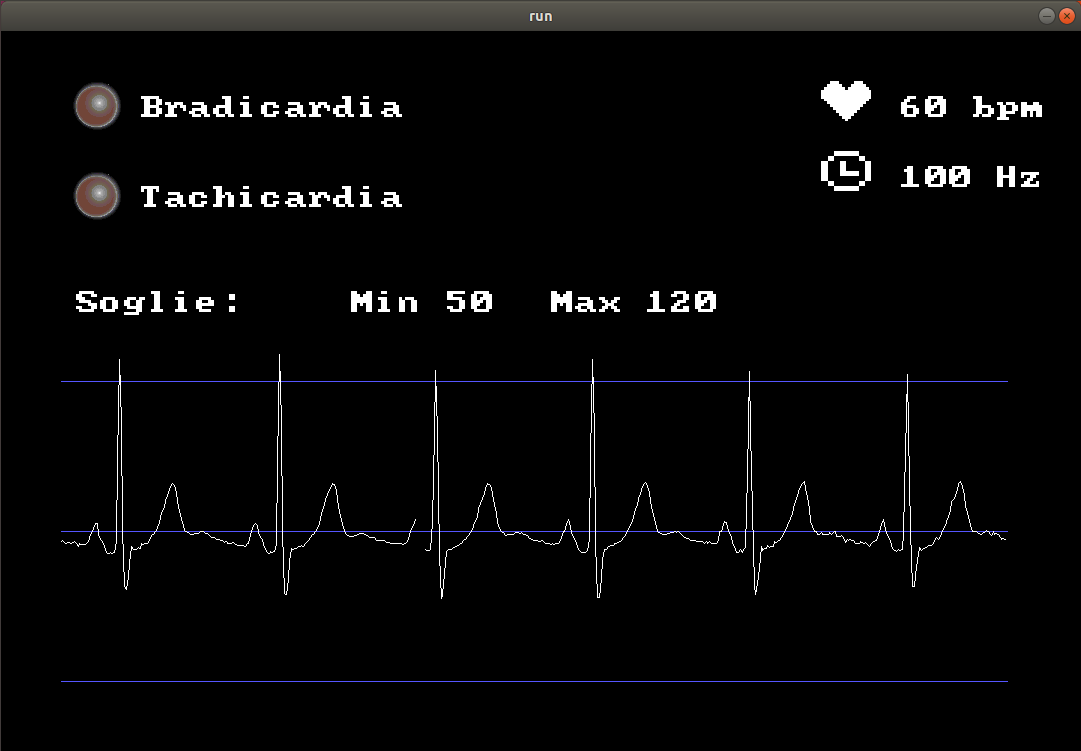
Si ha un complesso QRS se un punto supera la soglia primaria e un certo numero di punti successivi supera la soglia secondaria. Nel caso di un ECG campionato a 250 Hz si considerano i 6 valori successivi, 3-4 nel caso di 100 Hz.

Tramite la frequenza è possibile diagnosticare le seguenti patologie:

* Bradicardia: riduzione critica del ritmo cardiaco, mediamente sotto i 50 battiti al minuto
* Tachicardia: seria accelerazione del battito cardiaco, mediamente sopra i 120 battiti al minuto

Tali parametri possono però variare in base alle caratteristiche ed alla storia clinica del paziente, pertanto è utile che all’utente venga permesso di modificare i valori limite per la segnalazione delle irregolarità.

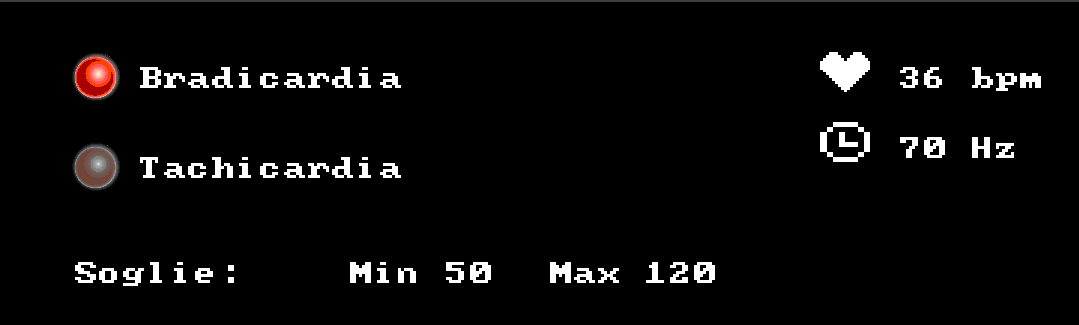
# Interfaccia utente



L’interfaccia può essere divisa in 4 sezioni: il grafico dei valori, la frequenza cardiaca, i led per le segnalazioni e le soglie di decisione.

Il grafico viene fatto scorrere inserendo progressivamente i nuovi valori con una rappresentazione ciclica (i primi valori vengono sovrascritti una volta raggiunto il termine destro dell’area dedicata).

La frequenza cardiaca è mostrata in alto a destra nella schermata. Quando questa non rispetta i vincoli indicati per le patologie si “accende” il relativo led.



I valori di soglia considerati sono mostrati a video sopra il grafico. L’utente li può modificare da tastiera:

* Tasto ↑ : incrementa la soglia massima (tachicardia)
* Tasto ↓ : decrementa la soglia massima
* Tasto → : incrementa la soglia minima (bradicardia)
* Tasto ← : decrementa la soglia minima

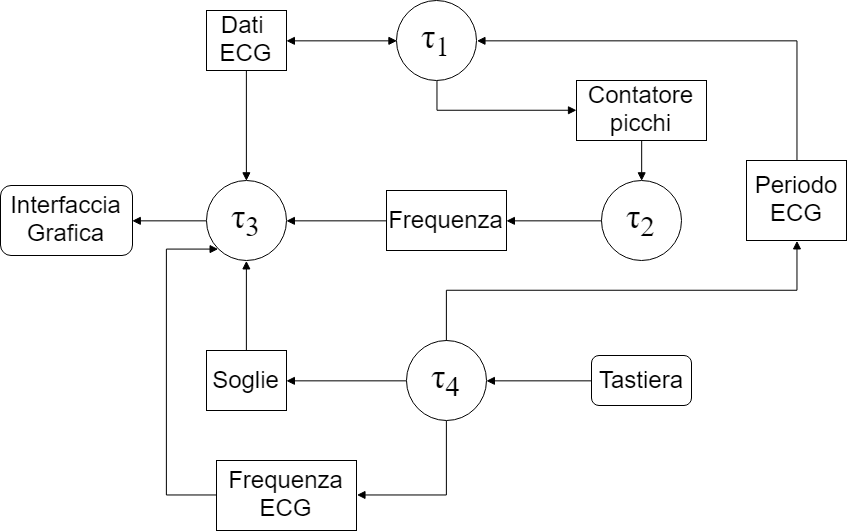
L’utente può inoltre andare ad agire sulla frequenza di lettura dell’ECG, incrementandola di 10Hz con la pressione del tasto M e decrementandola della stessa quantità con il tasto N.

# Task

L’applicazione viene realizzata tramite l’interazione di diversi task, ognuno con specifiche funzionalità. In particolare:

* Un task per il calcolo dei picchi tramite complessi QRS
* Un task per il calcolo della frequenza cardiaca
* Un task che si occupa della grafica a schermo
* Un task per leggere i comandi da tastiera

## Task di calcolo dei picchi

Il task si occupa di far scorrere l’indice di lettura dell’array con i dati dell’ECG e di calcolare il numero di complessi QRS. Per farlo, applica l’algoritmo di Ahlstrom-Tompkins e calcola , dove rappresenta l’indice del campione attualmente in considerazione. Sfruttando questo valore, verifica quindi se sono presenti le condizioni per un picco e, in caso affermativo, incrementa il relativo contatore. Tuttavia, essendo le soglie calcolate facendo riferimento al massimo valore di ottenuto con tutti i campioni, si necessita di un numero di campioni sufficiente ad approssimarle in modo adeguato, per poi procedere ad un progressivo aggiornamento. Tenendo in considerazione una frequenza di campionamento pari a 100 Hz e il fatto che in due secondi è generalmente compreso almeno un battito, si sfrutta un set di 250 valori per inizializzare le soglie. Una volta ottenute queste, esso si occupa di creare il task per il calcolo della frequenza cardiaca che non avrebbe avuto motivo di iniziare a lavorare prima.

Terminato lo scorrimento dei dati contenuti nell’array, si occupa della chiusura del programma chiamando la funzione cancel sul task di gestione della tastiera.

Il task è periodico con periodo dipendente dalla frequenza dell’ECG. Considerando tale valore pari a 100 Hz, è stato scelto di inizializzare il periodo a 10000 microsecondi. Sarà poi l’utente a variarne il valore con il suo intervento. La deadline coincide con il periodo in quanto non è necessario un vincolo più stringente.

La priorità del task è massima in quanto tutti gli altri task dipendono dallo scorrimento dell’array.

Una delle risorse condivise a cui accede è l’array che contiene i valori dei campioni del segnale, di cui preleva i cinque più recenti ai fini dell’algoritmo, con il relativo indice per lo scorrimento. Il semaforo che ne gestisce l’utilizzo segue il protocollo Priority Inheritance (PIP). Questa scelta è dovuta al fatto che è necessario garantire al task il minor tempo di attesa possibile. In questo modo si può infatti evitare che esso prolunghi l’attesa sulla wait a causa dell’intervento di altri task, con priorità inferiore al calcolo dei picchi ma maggiore del possessore della risorsa, all’interno di una sezione critica.

Si introduce poi una variabile contatore per i complessi QRS riscontrati condivisa con il task di calcolo della frequenza.

Infine, il task condivide con quello per la gestione della tastiera il proprio periodo, dal momento che questo dovrà essere modificato qualora l’utente andasse ad agire sulla frequenza dell’ECG.

## Task di calcolo della frequenza cardiaca

Il task legge ogni due secondi il valore del contatore dei picchi e lo azzera. Una volta ottenute cinque letture consecutive, somma e moltiplica per sei le singole misurazioni al fine di effettuare una stima della frequenza cardiaca in battiti al minuto, aggiornandola successivamente ad ogni nuova lettura.

Il task è periodico con un periodo di due secondi ma la deadline non coincide con la fine di questo intervallo. Ciò è dovuto al fatto che il numero dei picchi deve essere prelevato prima che ne avvengano ulteriori completamente al di fuori dei due secondi. Poiché la rilevazione di un picco richiede cinque campioni che rispettino i requisiti e viene letto un dato ogni 10000 microsecondi, è stato scelto un valore minore di 50000 microsecondi, in particolare 45000 microsecondi. In questo modo verranno comunque presi in considerazione i picchi già iniziati alla scadenza dei due secondi nel caso in cui questi vengano rilevati prima della lettura del relativo contatore.

Dal momento che tale deadline risulta comunque abbastanza lunga, è stata scelta una priorità inferiore rispetto al task di lettura.

Il task di calcolo della frequenza condivide con quello di calcolo dei picchi la variabile usata per mantenere il conteggio dei battiti rilevati, accedendovi in lettura e successivamente in scrittura per azzerarlo.

Inoltre, salva il valore della frequenza cardiaca calcolata in una variabile utilizzata in seguito in lettura dal task grafico per mostrarla a video.

## Task grafico

Il task grafico si occupa di riprodurre la schermata dell’applicazione, in tutte le sue parti. Per prima cosa crea gli assi su cui andare a disegnare l’andamento dell’ECG. Il segnale viene quindi riprodotto tracciando segmento per segmento il collegamento tra i dati prelevati dall’array fatto scorrere dal task di calcolo dei picchi.

Accede poi alla frequenza salvata dal task che si occupa di calcolarla e la mostra a video nell’angolo in alto a destra. Utilizzando questo valore, verifica se è presente o meno una patologia e, in caso affermativo, “accende” il relativo led sostituendo l’immagine.

Infine, mostra a video gli attuali valori delle soglie su cui tale verifica viene effettuata e della frequenza dell’ECG.

Il task è periodico con un periodo di 20000 microsecondi, maggiore rispetto a quello di calcolo dei picchi ma in modo contenuto al fine di evitare scatti nel grafico. La deadline coincide con il periodo non avendo vincoli più stringenti.

La priorità è mediana rispetto al calcolo dei picchi e al calcolo della frequenza avendo una deadline intermedia tra questi.

Tra le risorse condivise si annovera l’array dei dati dell’ECG, a comune con il task di calcolo dei picchi, e la variabile con la frequenza con cui questi scorrono, aggiornata dal task della tastiera. Si accede poi alla variabile che riporta il valore della frequenza cardiaca calcolata dall’apposito task e a quelle contenenti le soglie, anch’esse modificabili dal task che legge i comandi da tastiera.

## Task per i comandi da tastiera

Poiché le soglie di frequenza e la velocità del segnale sono modificabili da tastiera, il task rimane bloccato in attesa della pressione di un tasto e agisce quindi sui valori, incrementandoli o decrementandoli a seconda del tasto scelto dall’utente. È possibile inoltre terminare il programma premendo il tasto ESC.

Il task è aperiodico dal momento che la sua azione è dipendente dall’interazione con l’utente.

Le risorse condivise a cui accede sono le variabili che mantengono i valori delle soglie per il riconoscimento delle patologie in cui esso può scrivere e da cui il task grafico può leggere. Per modificare la velocità, accede poi al periodo del task di calcolo dei picchi, calcolando il nuovo valore da assegnarvi sulla base della frequenza che condivide invece con il task grafico.

## Main

Il thread main si occupa dell’inizializzazione di variabili condivise, semafori, parametri dei task, finestra grafica e bottone di chiusura. Legge quindi il file contenente i dati dell’ECG e li salva in un apposito array. Successivamente crea i task, ad eccezione di quello che si occupa del calcolo della frequenza cardiaca, e chiama la funzione join sul task che gestisce i comandi da tastiera. In questo modo, quando tale task termina a causa della pressione del tasto ESC, il main prosegue e fa terminare tutti gli altri task tramite la funzione cancel. Si occupa quindi anche della terminazione di Allegro e della corretta chiusura del programma.

# Prove sperimentali

L’applicazione è stata sviluppata utilizzando il sistema operativo Ubuntu 17.10 installato su macchina virtuale.

Il primo sviluppo dell’applicazione è stato svolto includendo un quinto task che si occupava di leggere i campioni del segnale da file in maniera sequenziale, come se i valori fossero prelevati da un sensore in tempo reale. Successivamente è stato scelto di far leggere l’intero file dal task main per avere i dati a disposizione in un tempo minore.

## Prove con task di lettura

Il task di lettura era stato pensato periodico con periodo dipendente dalla frequenza dell’ECG. La deadline era stata scelta pari al periodo in quanto, per compiere la lettura, avrebbe avuto a disposizione tutto il tempo fino al passaggio ad un nuovo dato.

La prima prova è stata svolta eseguendo il solo task di lettura senza la schermata grafica. È stata riscontrata la presenza di deadline mancate a frequenze superiori in genere agli 80 Hz. Tuttavia, la perdita di alcuni dati risulta tollerabile a frequenze superiori dato che si ha un numero di campioni elevato, più dell’effettivo bisogno. Per creare un’applicazione verosimile è stato scelto quindi di mantenere comunque una frequenza di 100 Hz.

È stato quindi introdotto il task per la gestione della grafica effettuando prove con periodi differenti. Il risultato è stato che con un periodo circa doppio rispetto al task di lettura non si presentano deadline nella rappresentazione grafica, oltre a non aumentare quelle nell’acquisizione dei dati. Inoltre, lo scorrimento del grafico risulta essere abbastanza fluido. È stata effettuata anche una prova utilizzando il semaforo con la politica classica invece del protocollo PIP, dalla quale si è evidenziato un aumento significativo delle deadline mancate.

In questa realizzazione il task per il calcolo dei picchi era strutturato in maniera pseudoperiodica, attivandosi ogni volta che veniva segnalato un nuovo dato. Questo meccanismo era stato realizzato con una variabile condition sui cui veniva effettuata l’attesa. Tale soluzione era stata preferita ad un semplice semaforo di sincronizzazione per evitare molteplici letture degli stessi dati da parte del task di calcolo, cosa che si sarebbe altrimenti potuta verificare in seguito a più letture consecutive senza riuscire ad eseguire il calcolo, e quindi a più signal sul semaforo successive. Ad esempio, supponiamo che il task abbia eseguito due letture e che, prima della successiva, il task riesca ad effettuare due volte il calcolo. legge ogni volta a partire dall’indice per l’inserimento, quindi:

* Con il semaforo di sincronizzazione, il task ha incrementato due volte il semaforo e pertanto le due wait del task vanno a buon fine ma, prendendo i dati a partire dallo stesso indice, si calcolano due valori di uguali, uno dei quali risulta scorretto.
* Con la variabile condition, il prende i dati la prima volta e resetta la variabile. Di conseguenza, la seconda volta trova la variabile a zero e si blocca. In questo modo viene prodotto un solo (“saltando” il valore del periodo in cui non è stato eseguito il calcolo) ma non vengono generati dati scorretti.

L’inserimento del task per il calcolo dei picchi non ha portato alla comparsa di ulteriori deadline mancate. Il valore esatto di campioni oltre la soglia secondaria per la rilevazione dei picchi è stato testato fino ad assestarlo a 4 campioni per un ECG a 100 Hz.

A questo punto è stato attivato il task per il calcolo della frequenza, il quale non ha portato ad ulteriori deadline mancate.

Il task per la gestione dei comandi da tastiera non influisce sulle prestazioni generali con la sua sola attivazione, rimanendo in attesa della pressione di un tasto. Prove con pressione dei tasti in momenti critici non hanno evidenziato peggioramenti sensibili dei risultati.

Un tentativo di alleggerimento dei vincoli temporali è stato effettuato rendendo aperiodico il task grafico. Questo, invece di essere attivato ad intervalli prefissati, si bloccava su una variabile condition tramite cui il task per la lettura segnalava l’arrivo di un nuovo dato, similmente a quanto fatto per quello che si occupa di calcolare i picchi. Il risultato è stato, non solo un aumento delle deadline segnalate, ma anche il mancato rilevamento di alcuni picchi, in quanto il sovracampionamento non compensava la perdita dei dati. È stato calcolato che per il corretto funzionamento in questa forma andrebbe limitata la frequenza dell’ECG a 60Hz, con un rapido peggioramento delle prestazioni al crescere della stessa.

Una prova ulteriore è stata fatta per tarare le priorità dei processi. Il task di lettura ha subito mostrato la necessità di avere la priorità maggiore in assoluto (32). I task per la schermata grafica e per il calcolo della frequenza hanno mostrato una certa interdipendenza: nel caso il task per la frequenza abbia una priorità uguale o maggiore a quello per la grafica, viene introdotto un ritardo nel secondo, causando delle deadline mancate. Le priorità migliori sono quindi risultate essere 30 per il task che si occupa della grafica e 28 per quello che si occupa della frequenza cardiaca.

## Prove senza il task di lettura

Eliminando il task di lettura e facendo ereditare i suoi compiti al task di calcolo dei picchi, è stato possibile eliminare le deadline mancate dovute ai lunghi tempi di accesso al file. È stata quindi riscontrata una significativa diminuzione di queste, cosa che ha permesso una migliore esecuzione dell’applicazione anche a frequenze maggiori. Infatti, le deadline mancate risultano quasi completamente assenti fino ad una frequenza di 150 Hz.

È stato poi esteso il protocollo PIP anche alle altre risorse condivise del task di calcolo dei picchi, riscontrando un ulteriore leggero miglioramento.

Un ulteriore tentativo di incremento delle prestazioni è stato eseguito testando un algoritmo alternativo per il calcolo dei picchi. La scelta è ricaduta sull’algoritmo di Menard, basato sul calcolo della sola derivata prima filtrata. Tuttavia, la riduzione del costo computazionale non è stata tale da portare un effettivo miglioramento ed è stato quindi scelto di mantenere l’algoritmo iniziale, il quale offre una maggiore precisione.

L’applicazione è stata infine testata a frequenze del segnale decisamente più alte di quella effettiva di funzionamento ed è stato visto che, nonostante le deadline perse, il comportamento è rimasto sostanzialmente corretto dal punto di vista delle varie funzionalità.