ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

SVILUPPO DI UN PLUG-IN SOFTWARE PER L'ANALISI STATISTICA DI DATI PER STUDI DI ELETTROFISIOLOGIA

Relazione finale in
PROGRAMMAZIONE AD OGGETTI

Relatore
Prof. MIRKO VIROLI

Presentata da
LORENZO COTTIGNOLI

Seconda Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

Elements

Elettrofisiologia

Analisi in Post-Processing

Curve fitting

Filtri digitali

Indice

Introduzione

Lo scopo di questa tesi è fornire una relazione in merito alla progettazione e allo sviluppo di un modulo software che permetta l'elaborazione e l'analisi di dati acquisiti nell'ambito dell'elettrofisiologia, ossia lo studio delle proprietà elettriche dei canali ionici. La creazione di questo plug-in software è stata richiesta da parte dell'Azienda Elements S.r.l., presso la quale era stato svolto il tirocinio curriculare previsto nel piano di studi.

L'azienda si occupa della produzione di amplificatori miniaturizzati in grado di misurare correnti a bassa intensità prodotte dai canali ionici, pur mantenendo alte frequenze di acquisizione e accuratezza. I software rilasciati da Elements per poter interfacciarsi ai propri dispositivi sono EDR (Elements Data Reader) ed EDA (Elements Data Analyzer).

Mentre EDR si occupa di visualizzare, memorizzare e analizzare dati acquisiti in real-time, EDA permette la visualizzazione di dati in post-processing, ovvero consente di visionare dati precedentemente registrati su file. Il modulo software illustrato all'interno della tesi si pone come obiettivo quello di utilizzare i dati letti da EDA per effettuare una serie di analisi statistiche quali istogrammi, fitting, identificazione di tempi medi di eventi di blocking molecolari, determinazione della conduttanza, cinetica del canale, visualizzazione di spettri del segnale, FFT, filtraggio digitale ed altre misure specifiche. Esso quindi sarà integrato nel software denominato EDA già sviluppato dall'azienda.

L'azienda ha richiesto questo tipo di software, anche se ne esistono già molti altri, per poter avere un'applicazione semplice da utilizzare che sia mirata alle analisi utilizzate nel campo dell'elettrofisiologia. Inoltre, gli algoritmi utilizzati per avere dati consistenti dal tracciato di partenza in questo campo sono in continuo miglioramento ed avere un proprio software di analisi permette di creare nuovi tipi di analisi o migliorarne dei vecchi.

Il plug-in software è stato progettato ed implementato secondo le specifiche richieste dall'azienda, inoltre è stata mantenuta la massima flessibilità ed un'ottima estendibilità in previsione di nuove funzionalità future. Esso infatti è stato strutturato in maniera modulare cercando di rendere ogni componente il più indipendente possibile rispetto agli altri, questo permette un minor numero di modifiche a livello di codice.

La tesi è suddivisa in 7 capitoli:

Nel primo capitolo verrà introdotta l'azienda e i dispositivi da essa prodotti.

Il secondo capitolo ha come scopo quello di descrivere le tecnologie utilizzate per il completamento del progetto.

Nel terzo capitolo verranno analizzati i requisiti richiesti dall'azienda.

Nel quarto capitolo verrà fatta un'analisi più approfondita con l'ausilio di diagrammi UML.

Il quinto capitolo contiene la progettazione delle varie parti e gli algoritmi utilizzati.

Nel sesto capitolo verranno mostrate le scelte implementative fatte basandosi su quelle progettuali.

Infine nel settimo ed ultimo capitolo verrà descritta la fase di testing.

Capitolo 1

Elements s.r.l.

1.1 Elements: l'azienda [1]

Elements s.r.l. è un'azienda, nata come start-up nel Dicembre del 2013, nel campo della microelettronica. L'azienda produce strumenti di misurazione miniaturizzati ad alta precisione per analisi sui nano-sensori, specialmente in elettrofisiologia. L'elettrofisiologia si occupa dello studio delle correnti ioniche che attraversano la membrana cellulare degli organismi viventi. L'azienda grazie ad anni di ricerca nel campo della microelettronica applicata ai nano-sensori ed alla collaborazione con centri di ricerca e compagnie leader nel campo dell'elettrofisiologia, può vantare di contribuire all'innovazione dei relativi strumenti di misurazione presenti sul mercato.

1.2 I dispositivi Elements

I prodotti Elements, come accennato nell'introduzione, sono amplificatori miniaturizzati in grado di leggere correnti ioniche a bassa intensità. Le correnti ioniche sono segnali elettrici prodotti dal passaggio di ioni attraverso i canali ionici presenti sulla membrana cellulare o su BLM, Lipid Bilayer Membrane, ossia membrane fosfolipidiche ricreate artificialmente. Con il termine canale ionico si identifica una proteina trans-membrana, che se sottoposta a stimolazioni elettriche permette (stato di apertura) o impedisce (stato di chiusura) agli ioni di attraversare la membrana.

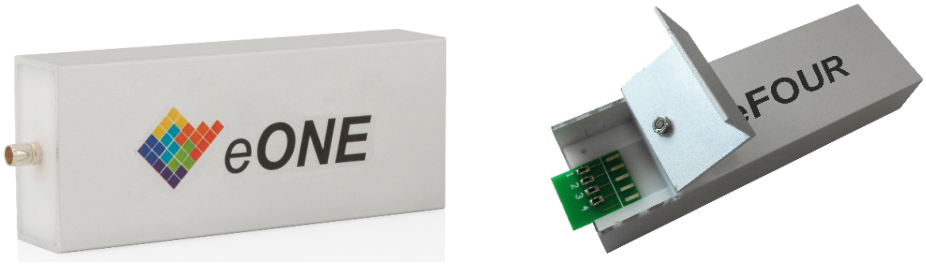
Oltre ai canali ionici, i dispositivi Elements trovano utilizzo nel settore emergente dei nanopori solid-state, ossia nanopori (fori dalle dimensioni nanometriche) realizzati su substrati artificiali quali ossido di silicio. Anche in questo caso le correnti che transitano attraverso i nanopori solid-state sono dell'ordine di grandezza delle correnti ioniche nelle proteine di membrana. Dalla

letteratura scientifica emergono applicazioni di nanopori nella translocazione di DNA e molecole di vario tipo, rendendo i nanopori solid state utilizzabili come sensori per eseguire sia sequenziamento genetico che spettroscopia di massa.

I dispositivi disponibili in casa Elements sono l'eOne, l'eFour e l'eSixteen.

eONE ed eFOUR

Questi due modelli sono quelli con cui Elements ha pian piano conquistato il mercato delle analisi su nano-sensori, essi infatti oltre a mantenere altissime prestazioni, sono anche i più piccoli amplificatori al mondo per l'analisi di segnali prodotti da nanopori. Questi modelli inoltre hanno alcune versioni che si differenziano in base alle funzionalità disponibili. L'eOne permette il monitoraggio di un singolo canale per volta, mentre con l'eFour il numero di canali supportati in simultanea arriva a quattro.



(a) Singolo canale (b) Quattro canali

Figura 1.1: Dispositivi distribuiti dall'azienda

eSIXTEEN

Il modello più recente è l'eSixteen e come suggerisce il nome utilizza fino a 16 canali in parallelo per la lettura di dati. Questo aumento esponenziale del numero di canali ha obbligato l'azienda alla creazione di un modello leggermente più grande rispetto ai due precedenti, ma esso rimane comunque di dimensioni molto piccole per un amplificatore di questa portata.

1.3 Caratteristiche dei dispositivi

Una delle caratteristiche innovative dei dispositivi distribuiti da Elements s.r.l., citati nel capitolo precedente, è la loro dimensione estremamente ridotta (30x15x74 mm) per essere amplificatori di altissima qualità e precisione. Altre caratteristiche sono: estrema semplicità di utilizzo, autoalimentato via USB, set di protocolli in tensione parametrizzabili, basso rumore ed alta banda.

1.3.1 Funzionalità principali

Le funzionalità principali offerte dai dispositivi Elements sono:

- Possibilità di scegliere la frequenza di campionamento dati: partendo da un minimo di 1,25 KHz fino ad un massimo di 200 KHz
- Possibilità di selezione tra due range in corrente: 20 nA o 200 pA
- Risoluzione ADC a 14 bit
- Possibilità di applicare stimoli in tensione tra ± 384 mV
- Possibilità di impostare o meno un filtro digitale sui dati letti per ridurne il rumore

1.3.2 Protocolli di tensione

Tramite l'applicazione distribuita dall'azienda, EDR, è possibile usufruire dei protocolli di tensione più utilizzati nell'ambito dell'elettrofisiologia configurando i relativi parametri in base alle proprie esigenze. La maggior parte degli esperimenti fatti dall'utente finale (generalmente un elettrofisiologo), sono possibili grazie a questa funzionalità.

1.3.3 Connessione al computer

Un'altra delle caratteristiche rivoluzionarie dei dispositivi Elements è la estrema semplicità di connessione con un PC. Questo è dovuto al fatto che i dispositivi utilizzano una connessione tramite l'USB sia per la comunicazione con il software sia per l'alimentazione. L'utilizzo dell'interfaccia USB per la connessione permette un'altissima portabilità, in quanto basta avviare il software EDR su un qualsiasi PC con una porta USB 2.0 ed infine collegare il dispositivo per iniziare ad utilizzarlo.

Figura 1.2: Connessione eONE al PC

1.4 Applicazioni nel campo dell'Elettrofisiologia

I dispositivi prodotti da Elements sono quindi utilizzati per lo studio e l'analisi dei canali ionici, ovvero studi che si effettuano in ambito elettrofisiologico. Questa scienza è una branca della fisiologia che studia i fenomeni elettrici dei tessuti organici o quelli provocati su un organismo dall'applicazione di un potenziale elettrico.

1.4.1 I canali ionici [2]

I canali ionici sono proteine di membrana di grandi dimensioni legate sul versante esterno a gruppi di carboidrati. Essi sono formati da svariate subunità che circoscrivono un poro acquoso il quale permette il passaggio selettivo degli ioni da una parte all'altra della membrana, cioè i canali ionici sono in grado di riconoscere e selezionare specie ioniche. Quando un canale permette il passaggio di ioni entra in uno stato di apertura, mentre quando impedisce il passaggio passa ad uno stato di chiusura. Le cinetiche dei flussi ionici attraverso i canali sono caratterizzati dalla conduttanza in risposta ad una certa forza elettrochimica, la quale è determinata dalla differenza di potenziale e dal gradiente di concentrazione dello ione considerato presenti ai capi della membrana.

Tramite l'utilizzo di appositi elettrodi il flusso di ioni può essere convertito in corrente elettrica ed infine analizzato dai dispositivi precedentemente descritti. L'analisi e l'interpretazione dei dati così ottenuti è l'oggetto di questa tesi, in quanto grazie a specifiche analisi è possibile studiare il comportamento dei canali ionici sotto diverse condizioni.

Capitolo 2

Tecnologie utilizzate

In questo capitolo verranno descritte nel dettaglio le tecnologie utilizzate per il completamento del modulo software richiesto dall'azienda.

2.1 Framework Qt

Per lo sviluppo del modulo software di cui tratta la tesi è stato utilizzato il framework Qt, in quanto già in uso nell'azienda. Questa scelta, fatta in precedenza dall'azienda, è dovuta dall'esigenza di avere software utilizzabile su diverse piattaforme e con ottime prestazioni. Qt, infatti, è un framework multi-piattaforma per lo sviluppo di software scritto in C++: grazie a piccole modifiche al codice esso è in grado di essere eseguito su sistemi diversi mantenendo però l'efficienza del linguaggio C++. Le piattaforme supportate sono molteplici: sistemi operativi per personal computer come Windows, OS X o Linux, quelli per sistemi mobile come Android, iOS e Window Phone ed infine piattaforme embedded quali QNX o VxWorks.

2.1.1 Architettura Software [3]

I concetti sul quale si basa Qt sono i seguenti:

- **Astrazione della GUI:** Nelle prima versione Qt aveva un proprio motore grafico in grado di astrarre la GUI rispetto alla piattaforma sulla quale stava girando. Per migliorare il framework sul fronte performance, nelle versioni più recenti i componenti si basano su API native.
- **Signals & Slots:** Questi due elementi vengono utilizzati per la comunicazione tra oggetti. Il meccanismo definito dall'utilizzo di signals e slots può essere utilizzato per semplificare al programmatore l'implementazione del design pattern Observer. Le signals, infatti, sono una sorta di

notifiche emesse da un oggetto quando il proprio stato interno è cambiato in qualche modo, tale per cui possa risultare interessante per altri oggetti. Le slot, invece, sono semplici funzioni in C++ con una particolarità, ovvero quella di poter essere invocate alla ricezione di una signal connessa con esse. Una signal può essere connessa a più slot e una slot può essere invocata da diverse signal, l'importante è che la signature delle signal e degli slots siano compatibili tra loro. Per poter permettere il corretto funzionamento di questo meccanismo è necessario un metodo per connettere una signal di un oggetto ad uno slot di un altro: questo è possibile grazie alla funzione connect offerta dalla libreria di Qt.

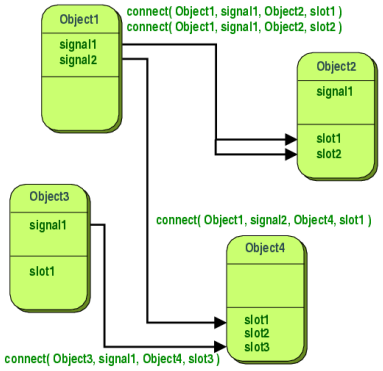


Figura 2.1: Meccanismo di Signal e Slot

- **Utilizzo di un compilatore di Meta-Oggetti (MOC):** Il compilatore di Meta-Oggetti è il programma che gestisce l'estensione Qt del linguaggio C++. Esso, infatti, quando legge un file header in C++ controlla se contiene la macro Q_OBJECT, in caso positivo produce un file sorgente c++ contenente le meta-informazioni relative a quella classe. Le meta-informazioni vengono utilizzate da Qt per aggiungere funzionalità non native agli oggetti, un esempio sono le sopracitate signal & slot.

2.1.2 Qt Creator [3]

L'IDE di sviluppo offerto da Qt è Qt Creator, un ambiente di sviluppo multi-piattaforma per linguaggi quali C++, Javascript e QML. Qt Creatore su macchine Linux e FreeBSD utilizza il compilatore C++ GNU Compiler Collection. Su Windows, invece, si può utilizzare come compilare MinGW oppure MSVC con l'installazione di default. Oltre ai compilatori appena citati è supportato anche Clang.
Le caratteristiche principale dell'editor sono:

- Un rapido sistema di navigazione del codice.
- Evidenziazione della sintassi e completamento del codice.
- Controllo statico del codice e suggerimenti sullo stile mentre si scrive.
- Support per rifattorizzazione del codice sorgente.
- Aiuto sensibile al contesto.
- Code folding, ovvero la possibilità di nascondere porzioni di codice mentre si lavora ad altre parti dello stesso file.
- Match delle parentesi e selezione del testo racchiuso.

Qt Creator al suo interno vanta un debugger visuale ed un designer per interfaccia grafica integrato.
Il debugger visuale per C++ è a conoscenza della struttura di molte classi Qt, aumentando così la capacità di visualizzare chiaramente i dati di Qt. Inoltre, Qt Creator visualizza le informazioni grezze da GDB in modo chiaro e conciso. Di seguito un elenco delle caratteristiche del debugger visuale:

- Esecuzione di interrupt programmati.
- Step del programma line-by-line or instruction-by-instruction.
- Impostare breakpoint.
- Esaminare le chiamate al contenuto dello stack, watcher, variabili locali e globali.

Il GUI (Graphical User Interface) designer utilizzabile nei progetti C++, invece, consente di progettare e creare rapidamente widget e finestre di dialogo sullo schermo, il tutto tramite l'utilizzo degli stessi widget che verranno inseriti nell'applicazione. I form sono completamente funzionanti e non necessitano di compilazioni per avere un'anteprima, in modo da poter controllare se l'aspetto

corrisponde a ciò che desideriamo.
Qt Creator non include un debugger per codice nativo. Per questo motivo fornisce un debugger plug-in che funziona come un'interfaccia tra il core di Qt Creator e un debugger nativo esterno per il linguaggio C++.
I debugger supportati sono:

- GNU Symbolic Debugger (GDB)
- Microsoft Console Debugger (CDB)
- Internal Javascript debugger
- LLVM debugger (LLDB)

2.2 Libreria grafica Qwt [4]

Qwt è l'acronimo di Qt Widgets for Technical applications, infatti questa libreria offre componenti grafici e classi di utilità per programmi in ambiti tecnici. Essa offre classi per grafici 2D, slider, scale e molto altro per il controllo e la visualizzazione di valori, array o intervalli di tipo double.

2.3 ALGLIB [5]

ALGLIB è una libreria multi-piattaforma per analisi numerica e elaborazione dati. La libreria è scritta in pseudocodice appositamente progettato per essere tradotto in automatico in vari linguaggi di programmazione. ALGLIB è un progetto relativamente nuovo in quanto lo sviluppo è iniziato nel 2008, mentre la GNU Scientific Library ha circa 20 anni di storia. ALGLIB, inoltre, è utilizzata da parecchie librerie e applicazione sia open source che commerciali.
ALGLIB fornisce facilitazioni per:

- Algebra lineare (algoritmi diretti, risolutori, EVD/SVD)
- Fast Fourier transforms (FFT)
- Integrazione numerica
- Interpolazione
- Fitting lineare e non lineare tramite metodo dei minimi quadrati
- Ottimizzazione

- Equazioni differenziali ordinarie
- Funzioni speciali
- Statistiche (Statistiche descrittive, test di verifica d'ipotesi)
- Analisi dati (classificazione/regressione, comprese le reti neurali)
- Versioni di precisione multipla di algebra lineare, interpolazione e algoritmi di ottimizzazione (usando MPFR per calcoli in virgola mobile)

I vantaggi che offre questa libreria sono:

- Supporta vari linguaggi di programmazione, quali C++, C#, FreePascal, Delphi, VBA.
- Funzionalità identiche per ogni linguaggio di programmazione.
- Facilità di installazione.
- Portabilità.
- Supporto per calcoli a precisione multipla.

Questa libreria però possiede anche vari svantaggi:

- Solo l'edizione commerciale supporta il multi-threading.
- Non può utilizzare il SSE (Streaming SIMD Extensions)per velocizzare le operazioni in virgola mobile.
- Sebbene alcuni algoritmi di algebra lineare sono implementati in cache oblivious manner, alcune sottofunzioni (specialmente risolutori SVD) non possono lavorare efficientemente con matrici che non stanno dentro la cache della CPU.

2.4 Clampfit [6]

Clampfit è un software molto potente che permette di effettuare svariati tipi di analisi su dati. Questo software viene utilizzato in ambito elettrofisiologico per analizzare i dati salvati precedentemente attraverso i relativi dispositivi, come per esempio tutti i dispositivi Elements. Clampfit fa parte di una suite di software denominata pCLAMP, di cui vi fanno parte anche le seguenti applicazioni:

- Clampex: utilizzato per acquisizione dati e produzione di segnali di stimolo.
- AxoScope: utilizzato per la registrazione di grafici di segnali.
- MiniDigi: digitalizzatore a due canali.

Le caratteristiche fornite da Clampfit sono:

- **Utilizzo di finestre di lavoro:** Clampfit si basa su l'utilizzo di finestre di lavoro, le quali hanno diverse funzionalità in base alla loro tipologia. I tipi di finestre di lavoro disponibili sono:
  - ***Analysis:*** una volta aperto un file dati, in questa finestra verranno visualizzati graficamente i dati. Se un file contiene più tracciati allora nella stessa finestra verranno mostrati tutti i tracciati. Non c'è un numero massimo di file che possono essere aperti e mostrati nello stesso momento.
  - ***Data File Index:*** Questa finestra offre la possibilità di utilizzare un tool per la gestione dei file, il quale permette di costruire un indice di file dati. I file dati possono essere raggruppati insieme in file DFI separati, all'interno dei quali vengono ordinati secondo un insieme di parametri riportati per ciascun file.
  - ***Graph:*** Questa finestra permette di visualizzare i dati selezionati su di un grafico a due dimensioni. Ci sono vari grafici disponibili come grafici a dispersione, a linea e istogrammi. Nella stessa finestra possono essere visualizzati più grafici, in tal caso vi sarà una legenda che li identifica.
  - ***Lab Book:*** Questa finestra è un editor di testo per la registrazione degli eventi che si verificano mentre Clampfit è in esecuzione.
  - ***Layout:*** Questa finestra è un editor di impaginazione per la formattazione di fogli di calcolo e grafici pronti per la presentazione.

  - ***Results:*** Questa finestra contiene 20 fogli di calcolo, 13 che ricevono i risultati da determinate funzioni e 7 per uso generale.
  - ***Statistics:*** Questa finestra viene utilizzata per aprire file statistici creati con Clampfit.

- **Importazione di file:** nell'ambiente Windows Clampfit permette il trasferimento di dati ed immagini in modo semplice e rapido. Tramite le semplici operazioni di copia ed incolla è possibile importare risultati analitici da altri programmi in un foglio di calcolo della finestra Result. Infatti, impostando un delimitatore di colonna e altre opzioni, è possibile aprire qualsiasi file di testo in unoo di questi fogli di calcolo.

- **Condizionamento dati:** Una volta che i dati sono visualizzati nella finestra Analysis, Clampfit permette di usare vari tipi di condizionamento dati, tra il quali il filtraggio e la correzione della baseline. In aggiunta ai convenzionali filtri passa alto e passa basso, sono inclusi filtri notch e anti-interferenza elettrica.
  Riguardo alla correzione della baseline, è possibile effettuarlo applicando varie funzionalità fornite dal software, ma il metodo più utile per questo genere di operazione è quello manuale. Clampfit, infatti, una volta attivata la modalità manuale, permette di definire dei punti di flessione e di decidere come deve cambiare la pendenza della baseline.

- **Rilevazione eventi:** La rilevazione di eventi è una funzionalità che lega l'utilizzo dei grafici, la finestra Result e altre specifiche caratteristiche per rilevare gli eventi con la finestra Analysis dove è visualizzato il file desiderato. Un file dati per volta può essere analizzato, cioè vengono ricercati particolari caratteristiche del tracciato per essere contrassegnati come eventi biologici, come l'apertura di un canale ionico. Clampfit permette tre tipologie differenti per la rilevazione di errori e sono Single-channel search, Template searches e Threshold-based searches.

- **Analisi su singolo canale:** il software permette di effettuare una vasta quantità di analisi, tra i quali creazione di istogrammi, grafici I/V, spettro di potenza e molti altri.

- **Fitting e analisi statistiche:** Clampfit può vantare un vasto insieme di funzioni per il fitting di dati visualizzati nelle finestre Analysis, Graph e Result. Inoltre permette di definire una propria funzione per il fitting dei dati. La finestra di dialogo del fitting permette di selezionare il metodo di fitting e altri parametri relativi alla funzione e al metodo selezionati. Dopo aver eseguito il fitting sui dati verrà visualizzato il risultato nella

relativa finestra.
Per quanto riguarda le analisi statistiche, Clampfit possiede un insieme di analisi statistiche parametriche e non-parametriche applicabili ai dati nelle finestre Analisi e Risultato. In entrambi i casi è possibile effettuare le analisi sia su tutti i dati sia su un sotto insieme definito da cursori (finestra Analysis) o selezionati (finestra Result).

- **Creazione di immagini nella finestra Layout:** La riduzione e presentazione dei dati a volte implicano diversi passaggi, dal semplice fit dei dati all'estrapolazione della funzione complessa basata su di essi. Per questo motivo Clampfit è configurato con un vasto insieme di funzioni di fitting che permettono di avere sia i dati iniziali sia il risultato delle analisi nella stessa immagine. La finestra Layout può essere salvata su file e aperta solo tramite Clampfit, oppure si può esportare l'immagine con i comandi di copia ed incolla.

2.5 Bitbucket

Bitbucket è un servizio di hosting web-based per progetti che utilizzano Mercurial o Git. Bitbucket è scritto in Python usando il Framework per applicazioni web Django. Questo servizio permette di creare repository online pubblici o privati: il primo tipo viene utilizzato soprattutto nello sviluppo di software open source, mentre il secondo quando lo sviluppo del software deve rimanere privato ad un team di persone.

Capitolo 3

Analisi Dei Requisiti

Descrizione dei requisiti richiesti dal sistema.

3.1 Descrizione del Sistema

3.1.1 Prospettiva del Sistema

Il software progettato e sviluppato è un plug-in, da integrare nel software in uso dall'azienda, che permetta l'elaborazione e l'analisi di dati acquisiti e registrati dai dispositivi dell'azienda. L'azienda ha richiesto il modulo software in esame perché necessitava di un'applicazione in post-processing mirato alle analisi nel campo elettrofisiologico oltre all'analisi su nanopori solid state.

3.1.2 Funzionalità Principali

Il software prevede:

- Diverse tipologie di analisi sui dati
- Salvataggio delle singole analisi in diversi formati
- Importazione delle singole analisi effettuate con EDR
- Fitting dei dati secondo curve date
- Applicazione di filtri digitali sui dati

3.1.3 Requisiti espressi dall'azienda

L'azienda ha espresso di progettare e sviluppare un modulo software in grado di elaborare ed analizzare dati acquisiti nell'ambito dell'elettrofisiologia, ossia lo studio delle proprietà elettriche dei canali ionici. Il modulo prevede la possibilità di eseguire una serie di analisi statistiche quali istogrammi, fitting, identificazione di tempi medi di eventi di blocking molecolari, determinazione della conduttanza, cinetica del canale, visualizzazione di spettri del segnale, FFT, filtraggio digitale ed altre misure specifiche. Il modulo software sviluppato sarà integrato nel software esistente denominato EDA (Elements Data Analyzer) già sviluppato dall'azienda. Il modulo software richiesto deve essere scalabile in previsione di cambiamenti futuri e aggiunta di nuove funzionalità e analisi, in modo da limitare il codice da modificare ed impedire dipendenze inutili tra le classi. Inoltre, deve mantenere l'interoperabilità con il software EDR sviluppato dall'azienda e una buona portabilità, ovvero la possibilità di utilizzare il software su diversi sistemi operativi.

Le prestazioni minime richieste sono quelle di poter analizzare una mole di dati dell'ordine di un centinaio di MB in tempistiche adeguate. Inoltre, bisogna gestire la correttezza dei dati, ovvero confrontare i risultati ottenuti con software di terze parti specializzate nell'analisi di dati.

3.1.4 Utente Finale

Il tipico utente finale è *l'elettrofisiologo* che effettua operazioni di lettura di dati e analisi su di essi. Per questo motivo è necessario offrire funzioni specifiche all'ambito elettrofisiologico, mantenendo un'interfaccia utente semplice ed intuitiva.

3.1.5 Ambiente d'uso e Limitazioni

Il sistema dovrà essere multi-piattaforma, ovvero essere eseguibile su macchine Windows e Mac di fascia media mantenendo buone prestazioni e affidabilità. Essendo software post-processing dovrà essere gestito tramite algoritmi e strutture dati intelligenti per mantenere un'ottima efficienza e tempistiche di elaborazione ridotte.

3.1.6 Linguaggio e piattaforma di sviluppo

Per soddisfare i requisiti richiesti dall'azienda in fatto di portabilità ed efficienza si è utilizzato il framework Qt e il linguaggio ad oggetti C++ 11. Si è preferito il C++ ad altri linguaggi di programmazione ad oggetti, in quanto non lavorando su macchine virtuali le prestazioni sono molto migliori.

3.2 Interfaccia utente

Il software deve fornire un'interfaccia utente completa, intuitiva e funzionale mantenendo caratteristiche simili alle applicazioni maggiormente utilizzate nel campo dell'elettrofisiologia. L'interfaccia deve essere reattiva ed affidabile anche durante l'elaborazione di una elevata mole di dati.

3.3 Suddivisione in moduli

L'azienda ha richiesto che il software sia il più scalabile possibile, di conseguenza il plug-in dovrà mantenere una struttura modulare.

3.3.1 EDA Core

Con EDA Core si identifica il software preesistente fornito all'azienda e nel quale verrà integrato il plug-in oggetto di tesi. Il Core presenta le seguenti funzionalità:

- Lettura file dati con estensione .dat e .abf
- Plotting dei dati letti da file, possibilità di effettuare zoom ed impostare un offset sui dati
- Possibilità di esportare i grafici in formati quali .pdf e .svg

3.3.2 Analisi

Il plug-in deve permettere la possibilità di effettuare vari tipi di analisi. Questo sotto-modulo, infatti, contiene algoritmi e strutture dati relative alle analisi richieste dall'azienda. Le analisi attualmente fornite sono le seguenti:

- **Istogramma:** definito dall'utente la dimensione del bin, viene creato un istogramma sui dati selezionati.
- **Event Detection:** l'utente dopo aver definito due soglie e la tipologia di evento (impulso positivo o negativo) potrà visualizzare il numero medio di eventi, ovvero quante volte il segnale passa dalla prima alla seconda soglia.
- **Dwell Time e Inter-event Time:** basandosi sull'event detection si possono calcolare altri due dati di estrema importanza, ovvero il dwell time e l'inter-event time. Il primo consiste nella durata media di un evento e il secondo nel tempo medio che trascorre tra un evento e il successivo.

- **FFT:** definito la dimension del bin dall'utente viene calcolata la trasformata discreta di Fourier tramite l'algoritmo denominato Fast Fourier Transform (FFT). I dati visualizzati corrispondono al modulo.
- **Grafico I/V:** rappresentazione grafica della corrente in corrispondenza dei valori di tensione allo scopo di stimare la conduttanza del canale ionico sotto esame.
- **Current Step Detection:** l'utente dopo aver definito una tolleranza in tempo e corrente potrà visualizzare il numero di salti (step) in corrente rilevati dai dati in esame, la dimensione media degli step e la durata media delle baseline. Questo tipo di esame viene utilizzato per determinare quando avviene l'apertura (salto con valore positivo) o la chiusura (salto con valore negativo) di un canale ionico, la quantità media di ioni che vi passano attraverso e la durata media di apertura.

3.3.3 Fitting

Questo sotto-modulo è riservato alle funzionalità di curve fitting, ovvero quel processo di costruzione di una funzione matematica che abbia la migliore corrispondenza con una serie di punti dati. Le funzioni richieste dall'azienda sono:

- **Funzione lineare:** $f(x) = mx + b$
- **Funzione esponenziale:** $f(x) = Ae^{-\frac{x}{\tau}}$
- **Sommatoria di gaussiane:** $f(x) = \sum_{i=1}^{N} A_i \frac{e^{-(x-\mu_i)^2/2\sigma_i^2}}{\sigma_i\sqrt{2\pi}} + C$

3.3.4 Filtri digitali

Questo sotto-modulo contiene le classi relative ai filtri digitali applicati ai dati. I filtri permetto di compiere alcune funzioni su campioni di segnali discreti nel tempo e vengono utilizzati per eliminare il rumore e poter rilevare meglio i dati di interesse. La tipologia di filtri digitali attualmente richiesta dall'azienda sono i FIR (Finite Impulse Response).

Capitolo 4

Analisi

In questo capitolo verrà fatta un'analisi approfondita sui sotto-moduli del software oggetto di tesi.

4.1 Suddivisione in moduli

Per mantenere una buona scalabilità e manutenibilità, come già affermato nel capitolo 3, si è deciso di dividere il plug-in in più moduli.

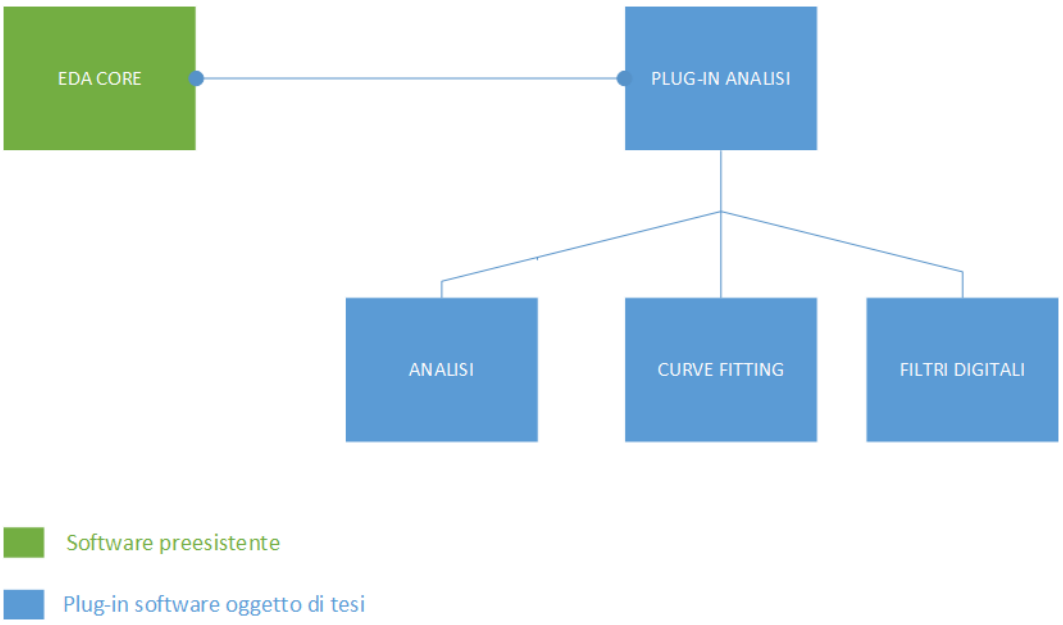


Figura 4.1: Suddivisione del plug-in software in più moduli

4.2 Panoramica generale del Core di EDA

Come già introdotto precedentemente, con EDA Core si definisce il software di base preesistente nel quale verrà integrato il plug-in oggetto di tesi. Il software permette la lettura di file dati con formato .abf e .dat. Per avere una maggiore comprensione riguardo la struttura del software preesistente, è necessaria una breve descrizione sui due formati supportati:

- **.dat:** formato binario dove i dati letti devono essere salvati secondo un formato specifico per mantenere l'interoperabilità con altri software. I dati relativi ad una acquisizione[1] possono essere suddivisi su più file. I dati al suo interno rispettano il seguente formato:

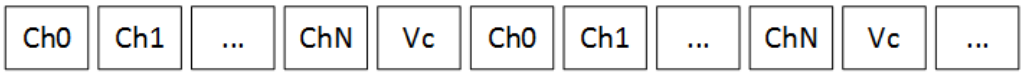


Figura 4.2: Struttura dei dati di un file *.dat*

- **.abf:**[7] l'ABF (AXON BINARY FILE) è un formato proprietario, può quindi essere letto e creato dai sviluppatori di terze parti usando la libreria ABFFIO. Il formato del file ABF è il seguente:

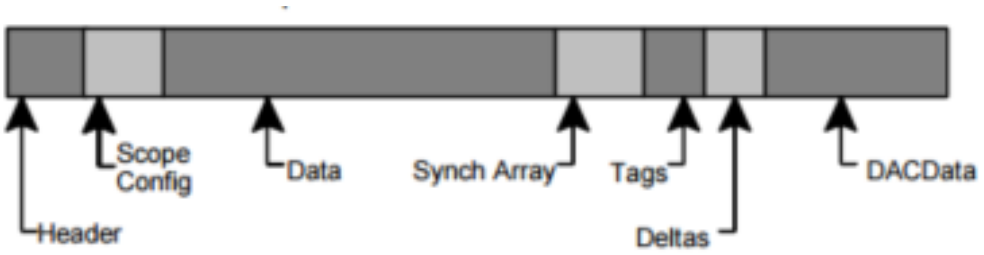


Figura 4.3: Struttura dei dati di un file *.abf*

Di seguito sono illustrate le componenti di EDA Core utilizzate per l'integrazione del modulo software richiesto:

[1] Le acquisizioni è possibile effettuarle tramite il software EDR (Elements Data Reader) dell'azienda, utilizzando i dispositivi descritti nel capitolo 1

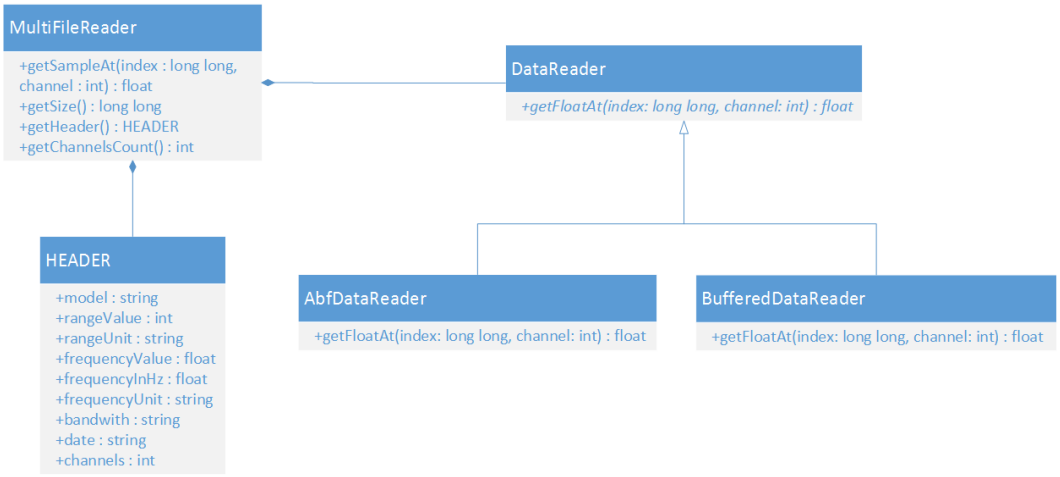


Figura 4.4: Diagramma UML delle classi relativo ad alcuni componenti di EDA Core

DataReader è un'interfaccia per la lettura di dati da file. Questa interfaccia viene implementata nelle classi *BufferedDataReader*, utilizzata per la lettura di file .dat, e *AbfDataReader*, la quale permette di importare dati dal formato standard della Axon (.abf). I dati in entrambi i formati possono essere suddivisi su più file, per questo motivo è stata creata la classe *MultiFileReader* la quale utilizza più *DataReader* per simulare un singolo flusso di dati.

4.3 Comunicazione tra i moduli

Per mantenere il sotto-modulo di analisi più indipendente possibile e allo stesso tempo utilizzabile dal software preesistente (EDA Core), si è deciso di utilizzare un'interfaccia intermedia. Questa interfaccia serve per adattare i dati provenienti dal *MultiFileReader*, appartenente al core di EDA per la lettura dei dati, al tipo di dati richiesti per le singole analisi. Inoltre, il modulo di analisi avrà il compito di convertire i vari risultati in un formato tale per cui il modulo di fitting possa funzionare correttamente. Il modulo dei filtri digitali invece va ad aggiungere funzionalità al *MultiFileReader* senza però modificarne la struttura.

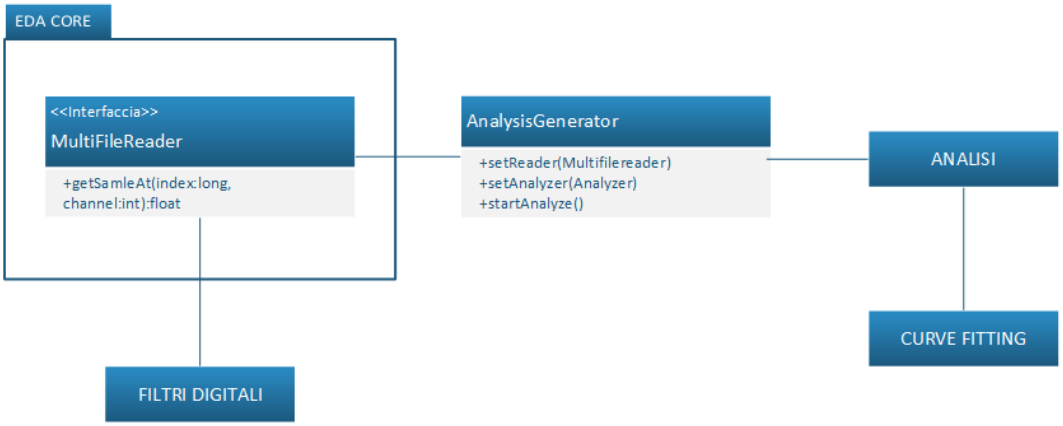


Figura 4.5: Suddivisione del plug-in software in più moduli

4.4 Processo di produzione

Data le specifiche richieste ed essendo un software sviluppato singolarmente, è stato deciso di basarsi sui principi della metodologia agile. I principi di questo metodo di sviluppo del software prediligono l'interazione costante tra cliente e sviluppatore, una maggiore flessibilità rispetto ai requisiti iniziali e software funzionante rispetto alla documentazione.

4.5 Modulo analisi

4.5.1 Specifiche del modulo

Realizzare un modulo che permetta di effettuare una serie di analisi, richieste dall'azienda, dati in input un'insieme di dati. Il modulo inoltre deve essere progettato in modo tale da prevedere la creazione di altri tipi di analisi oltre a quelle sviluppate in questo software. Le analisi richieste sono:

- **Istogramma:** creazione di un istogramma sui dati ricevuti e rilevare graficamente il picco massimo. L'utente deve avere la possibilità di decidere la dimensione del bin.

- **Event detection:** rilevazione di tutti gli eventi presenti nei dati acquisiti, visualizzazione dei valori medi del dwell time ed inter-event time e dei singoli valori istogrammi. Deve essere permesso all'utente di scegliere le due soglie per rilevare gli eventi, la dimensione del bin per gli istogrammi e il tipo di evento da rilevare (impulso positivo o negativo).

- **FFT:** Costruire la trasformata discreta di Fourier dei dati acquisiti e mostrare graficamente il modulo. L'utente deve avere la possibilità di selezionare i bin (multiplo di 2) su cui fare l'analisi, maggiore è il numero e più accurata sarà l'analisi.

- **Grafico I/V:** visualizzazione grafica dei dati su un piano cartesiano in cui l'ascissa corrisponde alla tensione applicata e l'ordinata alla corrente rilevata. Inoltre l'utente deve avere la possibilità di calcolare la conduttanza sui dati, ovvero la retta calcolata tramite interpolazione lineare dei punti medi di ogni singola ascissa.

- **Current step detection:** rilevazione di tutte le baseline (step) presenti nei dati ricevuti. Inoltre, deve essere visualizzato il numero di baseline, la media delle differenze tra una baseline e l'altra (current jump) e la media delle differenze tra il tempo in cui finisce una baseline e la sua successiva ed infine la costruzione di istogrammi sui i singoli valori delle medie appena citate.

4.5.2 Diagramma UML casi d'uso

Figura 4.6: Diagramma UML dei casi d'uso del modulo di analisi

4.5.3 Descrizione dello scenario

Scenario base

1. Selezione da parte dell'utente di tutti i dati in esame

2. Selezione del tipo di analisi da effettuare

3. Inserimento dei parametri richiesti dall'analisi scelta
4. L'analisi richiede al core di EDA un buffer di dati e li elabora
5. Finché ci sono dati da analizzare ripetere il punto 4
6. Visualizzazione numerica o grafica del risultato dell'analisi scelta

Varianti

1.a Selezione da parte dell'utente di una porzione di dati

6.a Nel caso di grafico I/V, calcolo e visualizzazione della conduttanza

4.5.4 Diagramma UML di sequenza

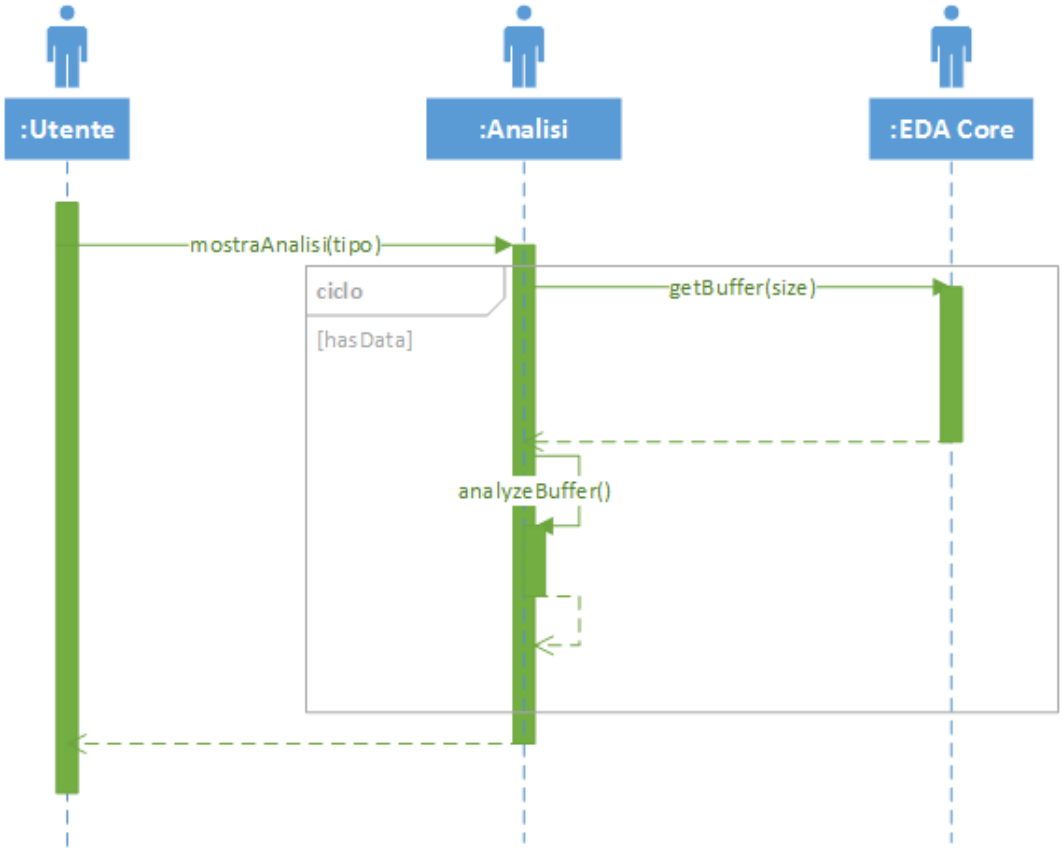


Figura 4.7: Diagramma UML di sequenza del modulo di analisi

4.5.5 Diagramma UML delle classi

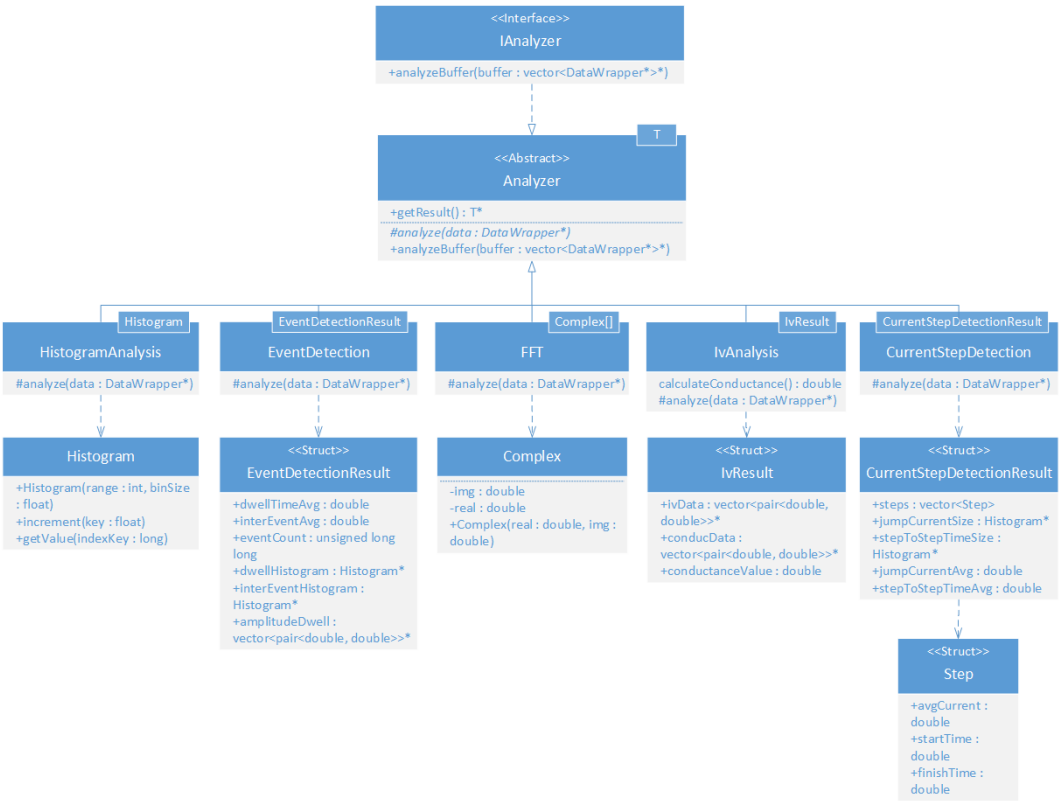


Figura 4.8: Diagramma UML delle classi del modulo di analisi

Il *DataWrapper* è una semplice interfaccia utilizzata dal modulo di analisi per effettuare le singole analisi. Questo permette di rendere le analisi il più indipendente possibile dal resto del software, infatti per poter integrare questo modulo basta semplicemente implementare un *SimpleDataWrapper* il quale tramite i metodi *getValueAt(channel)* e *getVc()* permette di recuperare i dati da analizzare.

4.6 Analisi modulo di curve fitting

4.6.1 Specifiche del modulo

Realizzare un modulo per la costruzione di una funzione matematica che abbia la migliore corrispondenza ad una serie di dati forniti. Le funzioni richieste dall'azienda sono quelle descritte nel capitolo 3.

4.6.2 Diagramma UML casi d'uso

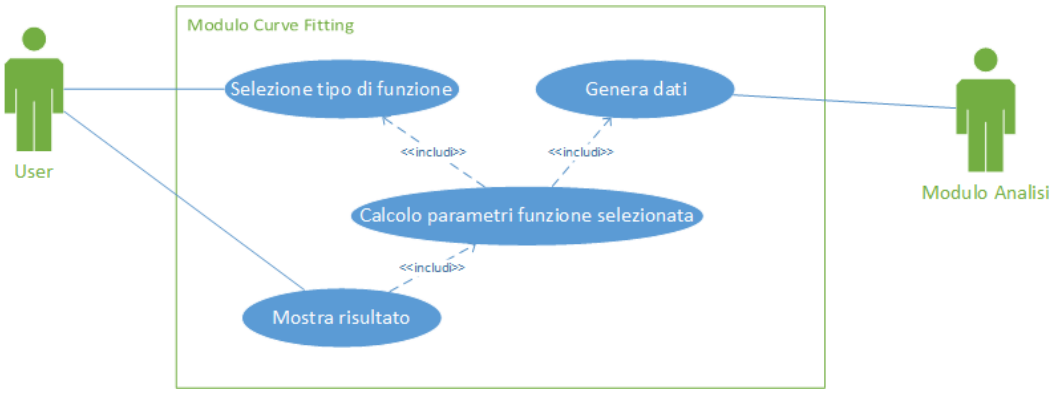


Figura 4.9: Diagramma UML dei casi d'uso del modulo di curve fitting

4.6.3 Descrizione dello scenario

Scenario base

1. Selezione del tipo di funzione per il fitting
2. Recupero dei dati dal modulo di analisi
3. Calcolo dei parametri della funzione selezionata
4. Visualizzazione del valore dei parametri calcolati e del grafico relativo alla curva risultante

Varianti

1.a Impostazione del numero di gaussiane da calcolare, solo se la funzione selezionata è quella Gaussiana

1.b Possibilità di forzare i parametri stimati iniziali o di calcolarli in automatico, solo se la funzione selezionata è quella Gaussiana

4.6.4 Diagramma UML di sequenza

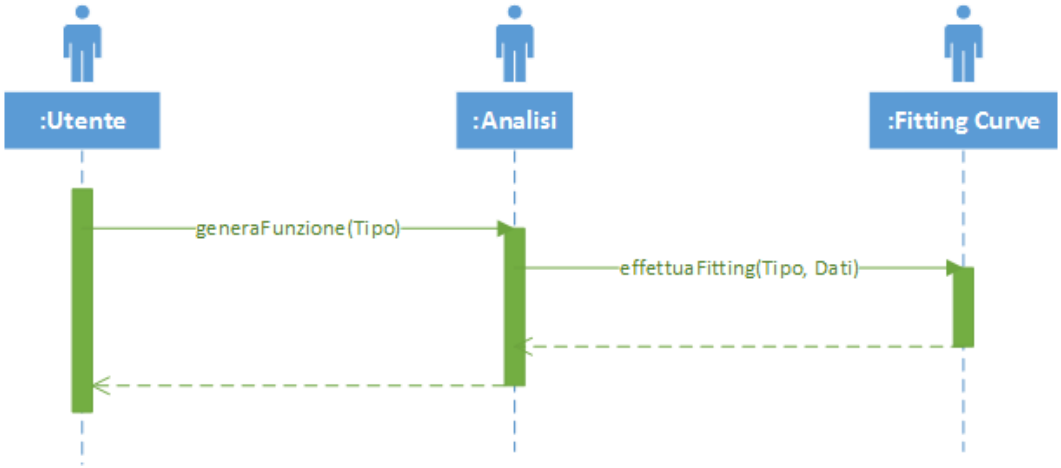


Figura 4.10: Diagramma UML di sequenza del modulo di curve fitting

4.6.5 Diagramma UML delle classi

Fitting funzioni lineari

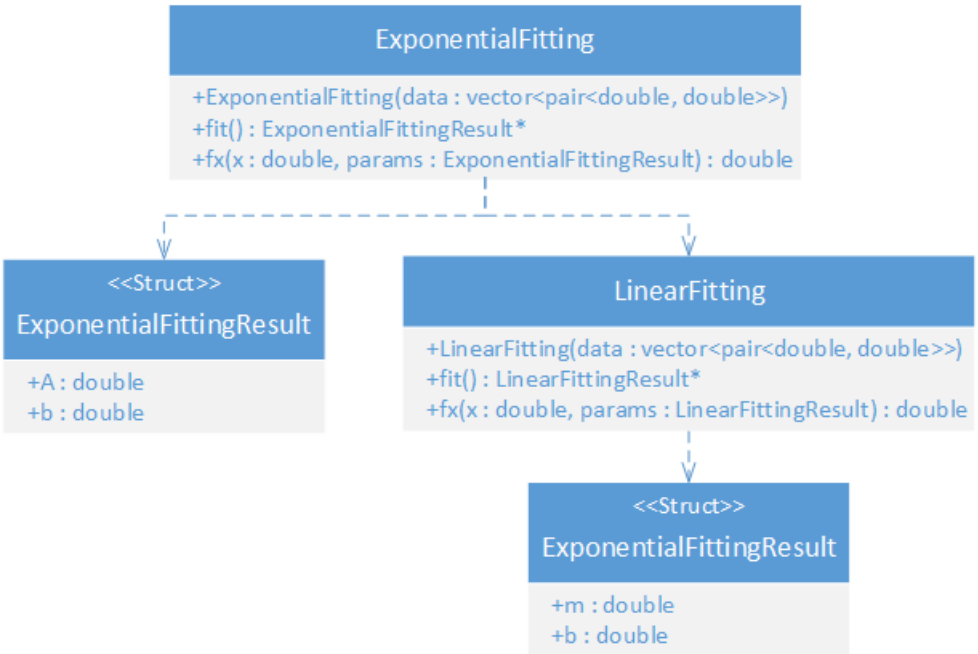


Figura 4.11: Diagramma UML delle classi per il fitting di funzioni lineari del modulo di curve fitting

Il fitting esponenziale è stato effettuato tramite una linearizzazione della funzione e la relativa interpolazione lineare, per questo motivo nel diagramma possiamo notare la dipendenza tra le due classi. I due tipi di fitting però sono stati gestiti separatamente nell'evenienza si decida di gestire in maniera differente il fitting esponenziale.

Fitting funzioni non lineari

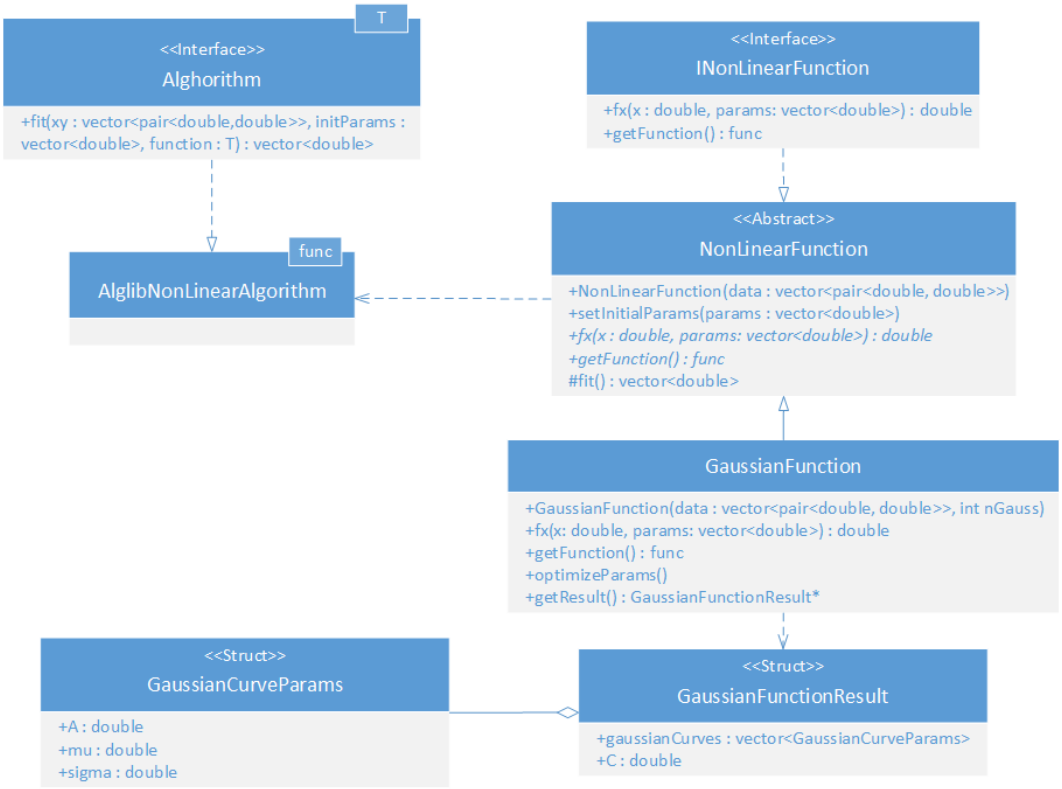


Figura 4.12: Diagramma UML delle classi per il fitting di funzioni non lineari del modulo di curve fitting

I fitting non lineari necessitano di algoritmi iterativi per trovare i parametri ottimi della funzione che interpoli i punti dati nel miglior modo possibile. L'algoritmo utilizzato è quello fornito dalla libreria ALGLIB. E' stata scelta questa struttura per rendere la funzione indipendente dall'algoritmo di ottimizzazione e fornire la possibilità di sostituire l'algoritmo senza troppe modifiche al codice sorgente.

4.7 Analisi modulo dei filtri digitali

4.7.1 Specifiche del modulo

Realizzare un modulo per la creazione, gestione ed applicazione di filtri digitali sui dati letti dal core di EDA. Il modulo dovrà essere gestito in modo da prevedere l'inserimento di altri tipi di filtri digitali. La tipologia di filtri richiesta dall'azienda è quella FIR (Finite Pulse Response).

4.7.2 Diagramma UML casi d'uso

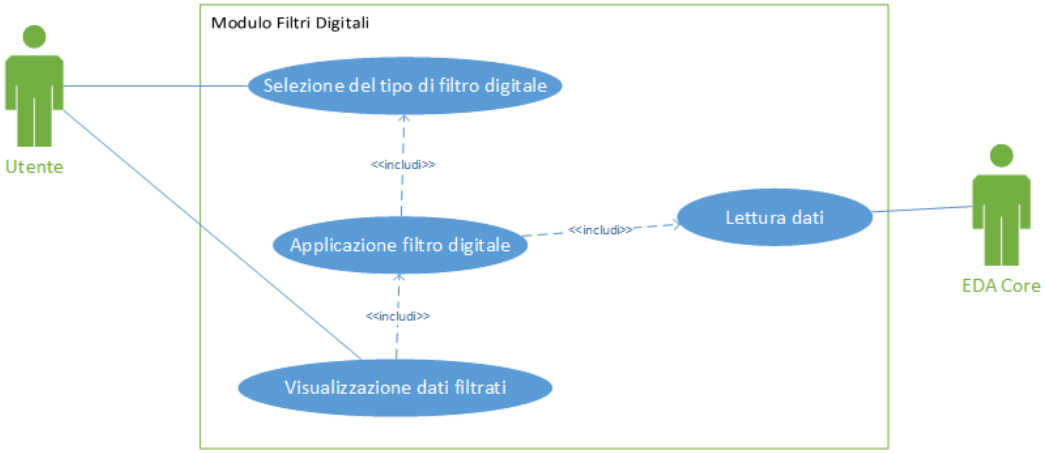


Figura 4.13: Diagramma UML dei casi d'uso del modulo dei filtri digitali

4.7.3 Descrizione dello scenario

Scenario base

1. Selezione del tipo di filtro FIR da applicare ai dati
2. Richiesta di un buffer dati di dimensione variabile in base al filtro scelto
3. Finché non sono stati filtrati tutti i dati ripetere il punto 2
4. Visualizzazione del tracciato iniziale filtrato secondo il filtro selezionato

4.7.4 Diagramma UML di sequenza

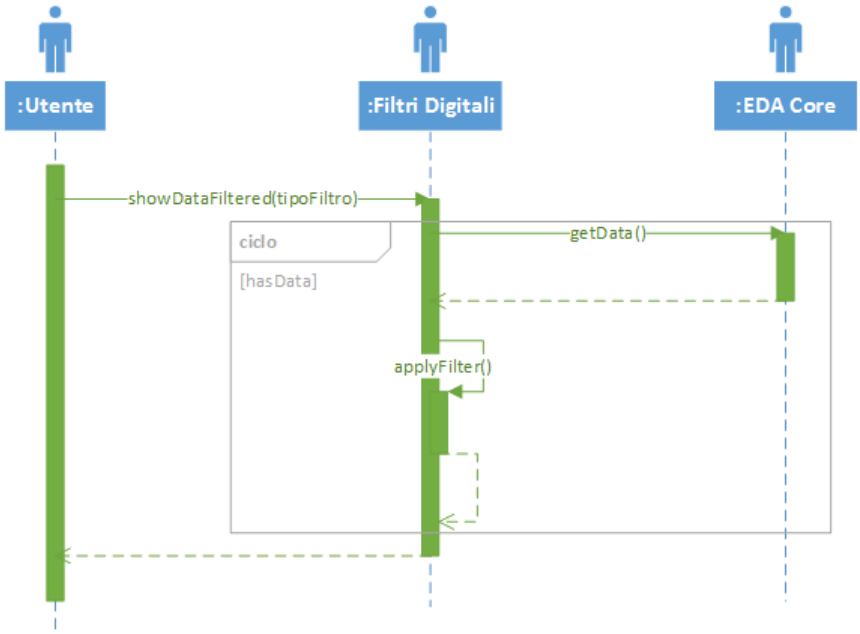


Figura 4.14: Diagramma UML di sequenza del modulo dei filtri digitali

4.7.5 Diagramma UML delle classi

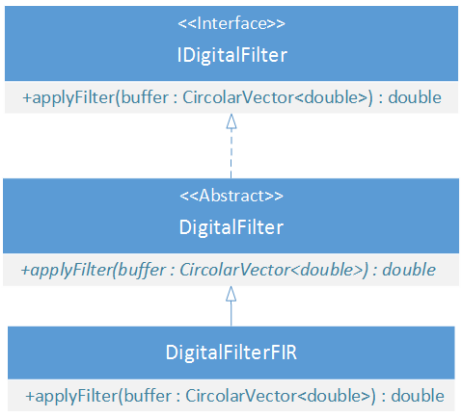


Figura 4.15: Diagramma UML delle classi del modulo dei filtri digitali

La struttura dei filtri è stata progettata in questo modo per permettere massima estendibilità. In questo modo l'aggiunta di una nuova tipologia di

filtri è facile, immediata ed evita modifiche a livello di codice. Nel prossimo capitolo analizzeremo nel dettaglio il funzionamento del modulo dei filtri digitali.

4.8 Qualità del software

Il software per poter essere distribuito ai clienti di Elements s.r.l. deve possedere alcune caratteristiche fondamentali. Le qualità richieste sono:

- **Correttezza:** Il software dovrà avere un comportamento corretto, ovvero in linea con quanto previsto dalla specifica dei requisiti.
- **Affidabilità:** Il software dovrà essere affidabile, cioè i malfunzionamenti devono avvenire il più raramente possibile o nel migliore dei casi mai.
- **Efficienza:** Il software dovrà fare un utilizzo intelligente delle risorse, data la possibile presenza di una mole consistente di dati da elaborare.
- **Usabilità:** Il software dovrà presentare un'interfaccia utente snella ed intuitiva.
- **Portabilità:** Il plug-in dovrà essere integrato in un software cross-platform e di conseguenza esserlo a propria volta.
- **Manutenibilità:** Il software dovrà essere sviluppato in modo tale da semplificare le operazioni di manutenzione. La modularità del sistema software è una caratteristica che permette di semplificare questa operazione, in quanto un errore corrisponde ad un minor numero di modifiche a livello di codice sorgente.
- **Interoperabilità:** I dati utilizzati per le analisi sono stati registrati da EDR, di conseguenza bisogna mantenere una struttura dati compatibile con quella utilizzata nel software di analisi in real time dell'azienda.

Capitolo 5

Progettazione

In questo capitolo verranno descritte le scelte progettuali relative ai vari moduli finora analizzati, quindi verranno definiti gli algoritmi e i design pattern che verranno utilizzati.

5.1 Descrizione tecnica dei design pattern utilizzati

In questo sottocapitolo verranno elencati e descritti dal punto di vista tecnico i vari design pattern utilizzati nel modulo software oggetto di tesi, i quali verranno poi citati nei capitoli seguenti per comprendere a fondo il contesto nel quale sono stati utilizzati. I design pattern sono schemi che definiscono una soluzione progettuale generale ad un problema ricorrente nella fase di progettazione. I pattern utilizzati sono i seguenti[2]:

Pattern strutturali

- **Adapter:** questo design pattern utilizzato in informatica nella programmazione orientata agli oggetti. A volte viene chiamato wrapper (ovvero involucro) per il suo schema di funzionamento. Il fine dell'adapter è di fornire una soluzione astratta al problema dell'interoperabilità tra interfacce differenti. Il problema si presenta ogni qual volta nel progetto di un software si debbano utilizzare sistemi di supporto (come per esempio librerie) la cui interfaccia non è perfettamente compatibile con quanto richiesto da applicazioni già esistenti. Invece di dover riscrivere parte del sistema, compito oneroso e non sempre possibile se non si ha a disposizione il codice sorgente, può essere comodo scrivere un adapter che faccia da tramite. L'Adapter è un pattern strutturale che può essere basato sia

su classi che su oggetti. Questo design pattern è parte fondamentale della programmazione a oggetti ed è stato formalizzato per la prima volta da Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides - la cosiddetta gang of four - nel libro Design Patterns.

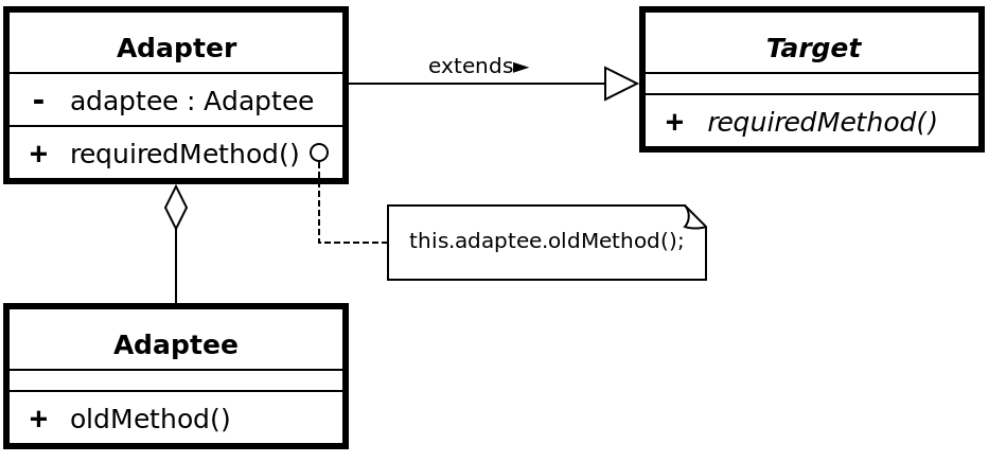


Figura 5.1: Diagramma UML delle classi del pattern Adapter

- **Decorator:** nella programmazione ad oggetti, il decorator è uno dei pattern fondamentali, definiti originariamente dalla gang of four. Il design pattern decorator consente di aggiungere durante il run-time nuove funzionalità ad oggetti già esistenti. Questo viene realizzato costruendo una nuova classe decoratore che "avvolge" l'oggetto originale. Al costruttore del decoratore si passa come parametro l'oggetto originale. È altresì possibile passarvi un differente decoratore. In questo modo, più decoratori possono essere concatenati l'uno all'altro, aggiungendo così in modo incrementale funzionalità alla classe concreta (che è rappresentata dall'ultimo anello della catena). La concatenazione dei decoratori può avvenire secondo una composizione arbitraria: il numero di comportamenti possibili dell'oggetto composto varia dunque con legge combinatoriale rispetto al numero dei decoratori disponibili. Questo pattern si pone come valida alternativa all'uso dell'ereditarietà singola o multipla. Con l'ereditarietà, infatti, l'aggiunta di funzionalità avviene staticamente secondo i legami definiti nella gerarchia di classi e non è possibile ottenere al run-time una combinazione arbitraria delle funzionalità, né la loro aggiunta/rimozione.

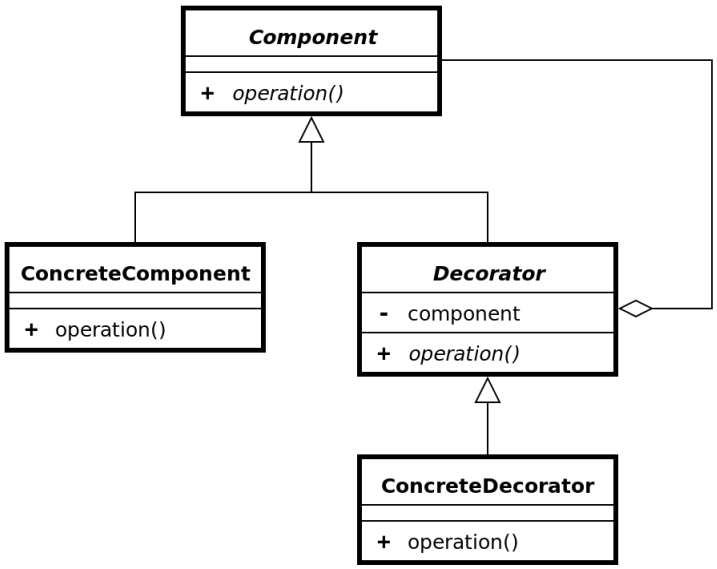


Figura 5.2: Diagramma UML delle classi del pattern Decorator

Pattern comportamentali

- **Template method:** il template method è un pattern comportamentale basato su classi, utilizzato in informatica nell'ambito della programmazione orientata agli oggetti. Questo pattern permette di definire la struttura di un algoritmo lasciando alle sottoclassi il compito di implementarne alcuni passi come preferiscono. In questo modo si può ridefinire e personalizzare parte del comportamento nelle varie sottoclassi senza dover riscrivere più volte il codice in comune. Template method è uno dei design pattern fondamentali della programmazione orientata agli oggetti definiti originariamente dalla cosiddetta gang of four, ovvero gli autori del libro Design Patterns.

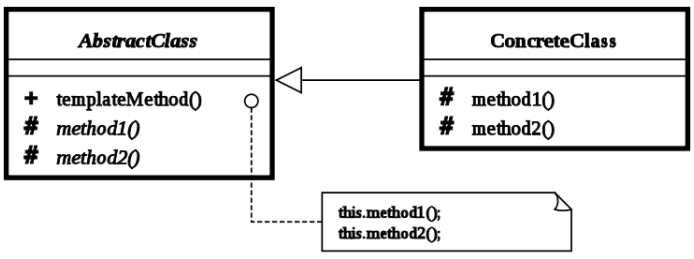


Figura 5.3: Diagramma UML delle classi del pattern Template Method

- **Observer:** l'Observer pattern è un design pattern utilizzato per tenere sotto controllo lo stato di diversi oggetti. È un pattern intuitivamente utilizzato come base architetturale di molti sistemi di gestione di eventi. Molti paradigmi di programmazione legati agli eventi, utilizzati anche quando ancora non era diffusa la programmazione ad oggetti, sono riconducibili a questo pattern. È possibile individuarlo in maniera rudimentale nella programmazione di sistema Windows, o in altri framework di sviluppo che richiedono la gestione di eventi provenienti da diversi oggetti, come ad esempio la funzione "OnMsgProc" per la gestione delle code di messaggi windows. Sostanzialmente il pattern si basa su uno o più oggetti, chiamati osservatori o observer, che vengono registrati per gestire un evento che potrebbe essere generato dall'oggetto "osservato", che può essere chiamato soggetto. Oltre all'observer esiste il concrete Observer che si differenzia dal primo perché implementa direttamente le azioni da compiere in risposta ad un messaggio; riepilogando il primo è una classe astratta, il secondo no. Uno degli aspetti fondamentali è che tutto il funzionamento dell'observer si basa su meccanismi di callback, implementabili in diversi modi, o tramite funzioni virtuali o tramite puntatori a funzioni passati quali argomenti nel momento della registrazione dell'observer, e spesso a questa funzione vengono passati dei parametri in fase di generazione dell'evento. Questo pattern, nel modulo software richiesto da Elements, non è stato implementato ma solamente utilizzato tramite il meccanismo di signal e slot concesso da Qt (vedi capitolo 2.1).

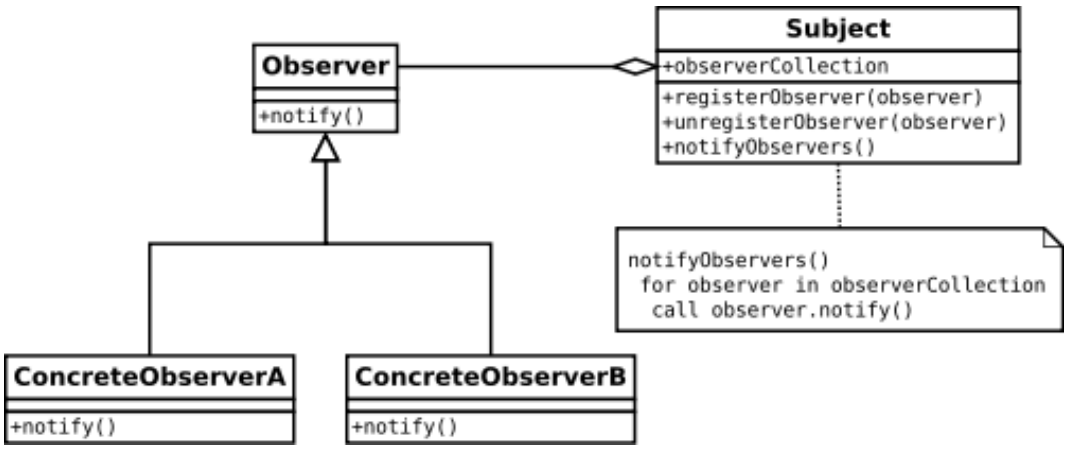


Figura 5.4: Diagramma UML delle classi del pattern Observer

Pattern architetturali

- **Model-View-Controller (MVC):** il MVC è un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati, ovvero dell'interfaccia utente, dalla logica di business, cioè quella dei dati veri e propri. Il ruolo dei componenti di questo pattern è il seguente:
  - **Model:** fornisce i metodi per accedere ai dati utili all'applicazione.
  - **View:** visualizza i dati contenuti nel model e si occupa dell'interazione con l'utente.
  - **Controller:** riceve i comandi dall'utente attraverso alla view e li attua modificando lo stato del Model e della View.

  La gestione delle notifiche tra i componenti quando avviene un passaggio di stato d'interesse (compito del Controller) verrà affidata al meccanismo di signal e slot presenti in Qt (vedi capitolo 2.1).

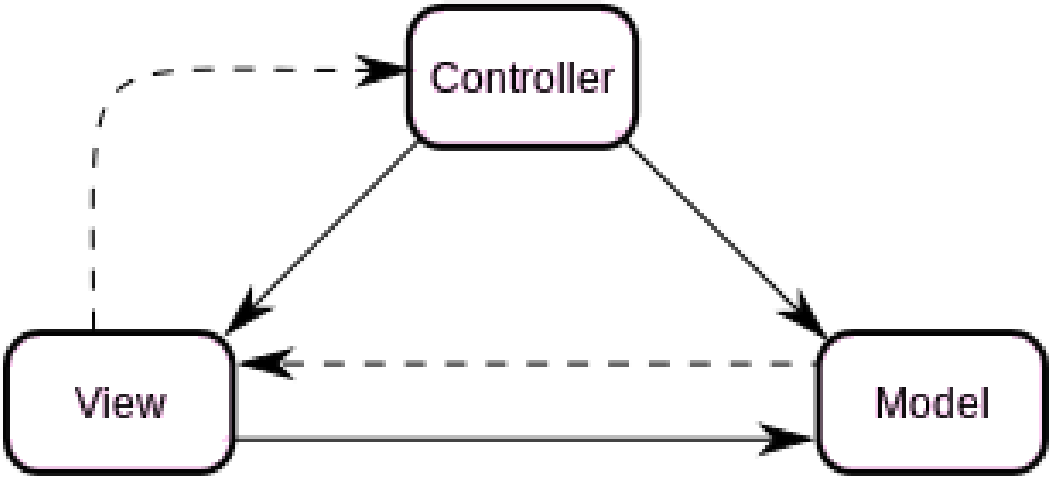


Figura 5.5: Schema relativo alla struttura del pattern Model-View-Controller

5.2 Interfaccia grafica

L'interfaccia grafica deve essere progettata per risultare intuitiva all'utente finale, ovvero deve permettere di utilizzare funzionalità complesse in modo semplice e rapido. Il tutto è stato progettato tramite il pattern architetturale MVC (citato nel capitolo precedente) affidando la logica del Controller al meccanismo di signal e slot fornito da Qt. La parte di interfaccia grafica che ha richiesto maggior attenzione durante la fase di progettazione è quella relativa alle finestre delle singole analisi.

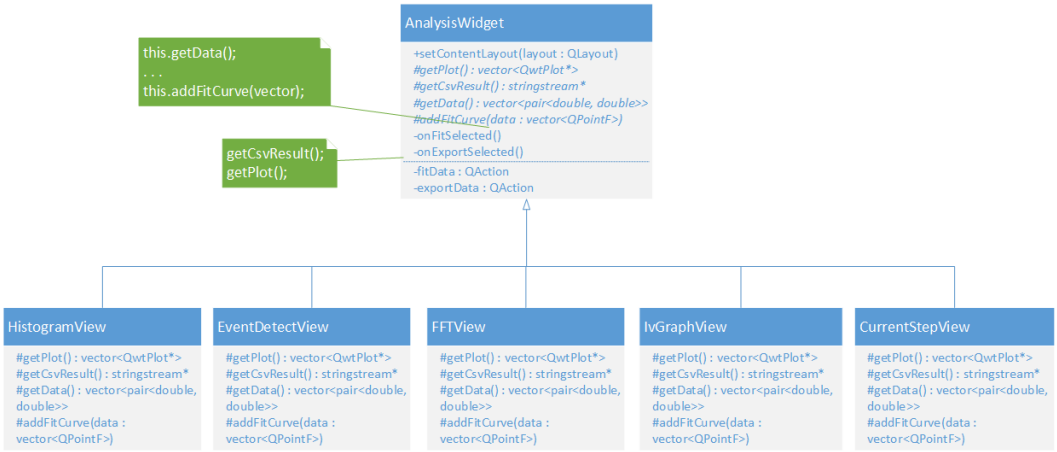


Figura 5.6: Diagramma UML delle classi relativo all'interfaccia utente usata per le analisi

Come si può notare da questo diagramma si è utilizzato più volte il pattern comportamentale Template Method (vedi capitolo 5.1), ovvero con i metodi privati *onFitSelected()* e *onExportSelected()*. Solitamente il metodo che implementa questo design pattern è pubblico in modo tale che possa essere invocato dall'esterno ma grazie al meccanismo di signal e slot non è necessario, questo perché i *QAction* (classi appartenenti alla libreria di Qt che permettono di inserire pulsanti nella barra dei menu della propria applicazione) quando vengono clickati emettono una signal che può essere connessa ad un metodo, purché questo metodo sia definito come slot. In questo modo alla pressione del QAction fitData verrà eseguito il metodo privato *onFitSelected()* e nel caso del QAction exportData verrà eseguito *onExportSelected()*.

5.3 Modulo analisi

Le analisi richieste sono varie e in futuro potrebbero esserne necessarie altre, per questo motivo si è scelto di utilizzare una struttura come quella descritta nel capitolo 4.5.5 in modo da renderle facilmente estendibili. Per questo motivo è stato necessario creare una classe astratta *Analyzer* da cui estendere le singole analisi, la quale utilizza il pattern comportamentale Template method per analizzare i dati. Infatti, la classe possiede un metodo pubblico (*analyzeBuffer(buffer)*) che permette di analizzare un buffer di dati passati come parametro, il quale internamente richiama su ogni dato del buffer il metodo protetto *Analyze(DataWrapper)*. Questo secondo metodo deve essere implementato da ogni classe di analisi che estenda la classe *Analyzer*.
Per rendere la struttura delle analisi indipendente si è deciso di creare un'interfaccia *DataWrapper*, la quale viene utilizzata come pattern Adapter. Infatti, chiunque voglia usufruire di queste analisi, basta che crei un'implementazione dell'interfaccia *DataWrapper* che adatti il formato originale dei propri dati in dati analizzabili da questo modulo.
Un'altra classe fondamentale per l'analisi dei dati è l'*AnalysisGenerator*. Questa classe è designata al passaggio di dati dal *MultiFileReader* all'analisi scelta. Di seguito verranno elencati i vari passaggi dell'algoritmo utilizzato da questa classe:

- Creazione di un buffer di *DataWrapper*.
- Leggi il dato alla i-esima posizione tramite il *MultiFileReader*.
- Creazione di un'oggetto *SimpleDataWrapper* per adattare il dato appena letto.
- Inserimento dell'oggetto appena creato nel buffer.
- Se il buffer è pieno, passare il buffer come parametro al metodo *analyzeBuffer(buffer)* dell'analisi scelta.
- Se ci sono ancora dati da leggere e il buffer non è ancora pieno allora incrementare la posizione e ripetere dal punto 2. nel caso ci siano ancora dati da leggere e il buffer è pieno allora ripetere dal punto 1.
- Notificare di aver concluso l'analisi dei dati.

5.3.1 HistogramAnalysis

Questa analisi costruisce un'istogramma sui dati ricevuti. Per mantenere efficiente la struttura dell'istogramma si è creata una classe che alloca al suo interno un vettore di dimensione prefissata in base al range e al bin size passati come parametri al costruttore. Questa struttura è stata pensata per una maggiore efficienza in tempo rispetto a quella in spazio.
Sia B la dimensione del bin e R il range, allora la dimensione D del vettore è:
$D = \frac{2R}{B}$
Sia x un valore con la virgola da inserire nell'istogramma, allora il suo indice i nel vettore è:
$i = round(\frac{x}{B})$

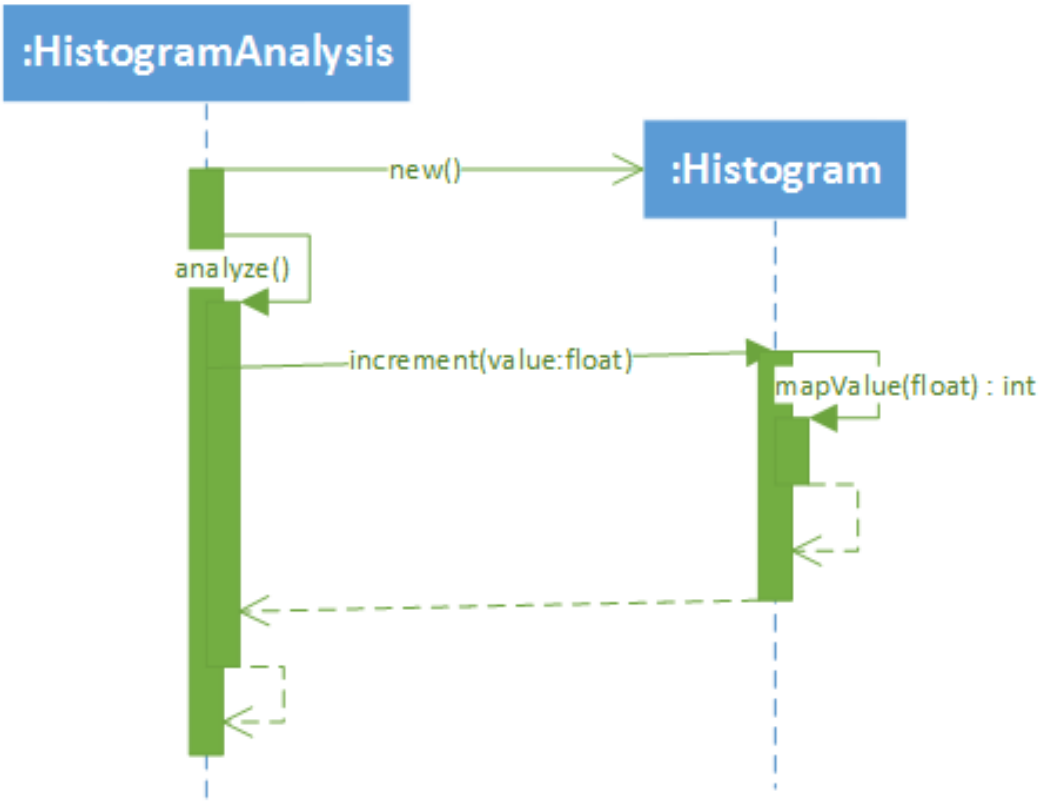
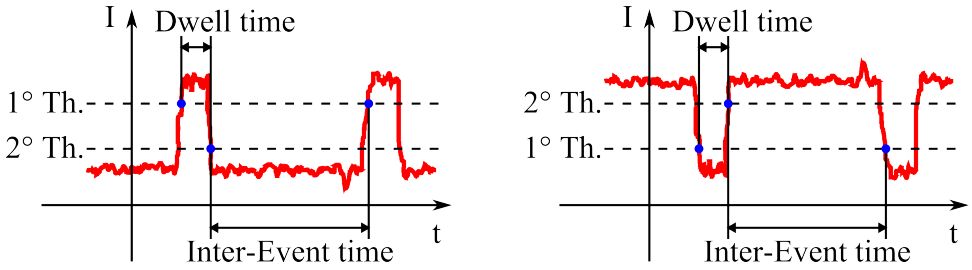


Figura 5.7: Diagramma UML di sequenza relativo all'algoritmo utilizzato per la creazione di un istogramma

5.3.2 EventDetection

Questa analisi permette, definite due soglie ed una tipologia, di rilevare gli eventi presenti nel segnale elettrico in esame. Le tipologie di eventi sono due, ovvero positive pulse e negative pulse.



(a) Tipologia positive pulse (b) Tipologia negative pulse

Figura 5.8: Tipi di eventi rilevabili dall'analisi EventDetection

Come viene illustrato dalle immagini la durata di un evento è definito tramite il Dwell time, mentre il tempo tra un evento e il successivo si chiama l'Inter-event time. Grazie a questi due dati e la ampiezza media di ogni singolo evento si è in grado di costruire i grafici richiesti dalla azienda.

5.3.3 FFT

Questa analisi calcola la DFT (Discrete Fourier Transform), ovvero un particolare tipo della trasformata di Fourier che converte una collezione finita di campioni equispaziati di una funzione in una collezione di coefficienti di una combinazione lineare di sinusoidi complesse, ordinate al crescere della frequenza.
Data la sequenza di numeri complessi $x[n], n = 0, 1, ..., N-1$
La DFT è definita come:

$$X[k] = DFTx[n] = \sum_{n=0}^{N-1} x[n] e^{-ik\frac{2\pi}{N}n}$$

L'algoritmo utilizzato per calcolare la DFT è la FFT (Fast Fourier Transform). Questo algoritmo permette ridurre il costo computazionale del calcolo della DFT da $O(n^2)$ a $O(nlogn)$. Il risultato della DFT, come già descritto prima, è un vettore di numeri complessi quindi necessita di un ulteriore operazione per il calcolo del modulo:

$$||X[k]|| = \sqrt{Re(X[k])^2 + Im(X[k])^2}$$

5.3.4 Grafico I/V

Questa analisi permette di calcolare la conduttanza a partire dai campioni del segnale elettrico in esame. Per calcolare la conduttanza per prima cosa è necessario mappare i campioni relativi al segnale elettrico in esame, ovvero dato un campione creare un punto nel piano cartesiano con ordinata corrispondente alla corrente ed ascissa uguale alla differenza di potenziale applicata. Una volta ottenuto l'insieme dei dati, è necessario calcolare per ogni valore di differenza di potenziale (ascissa) il relativo valor medio di corrente (ordinata). Dopo aver ottenuto i punti medi, si effettua un'interpolazione lineare (vedi capitolo 5.4) e la pendenza della retta corrisponde al valore di conduttanza.

5.3.5 CurrentStepDetection

Questa analisi ha come scopo quella di rivelare l'apertura e la chiusura dei canali ionici. Questa caratteristica è rilevabile in quanto ad ogni apertura avviene un passaggio di corrente, quindi un aumento di ampiezza del segnale, mentre ad ogni chiusura la corrente smette di attraversare la membrana quindi avviene una riduzione dell'ampiezza. Per evitare che il rumore del segnale dia risultati errati, l'utente può impostare due parametri δT e δI, il primo corrisponde a quanto tempo deve rimanere stabile il segnale per essere considerata un'apertura e il secondo a quanto deve essere la differenza di corrente tra una baseline e l'altra.
Il diagramma seguente mostra l'algoritmo utilizzato per questa analisi.

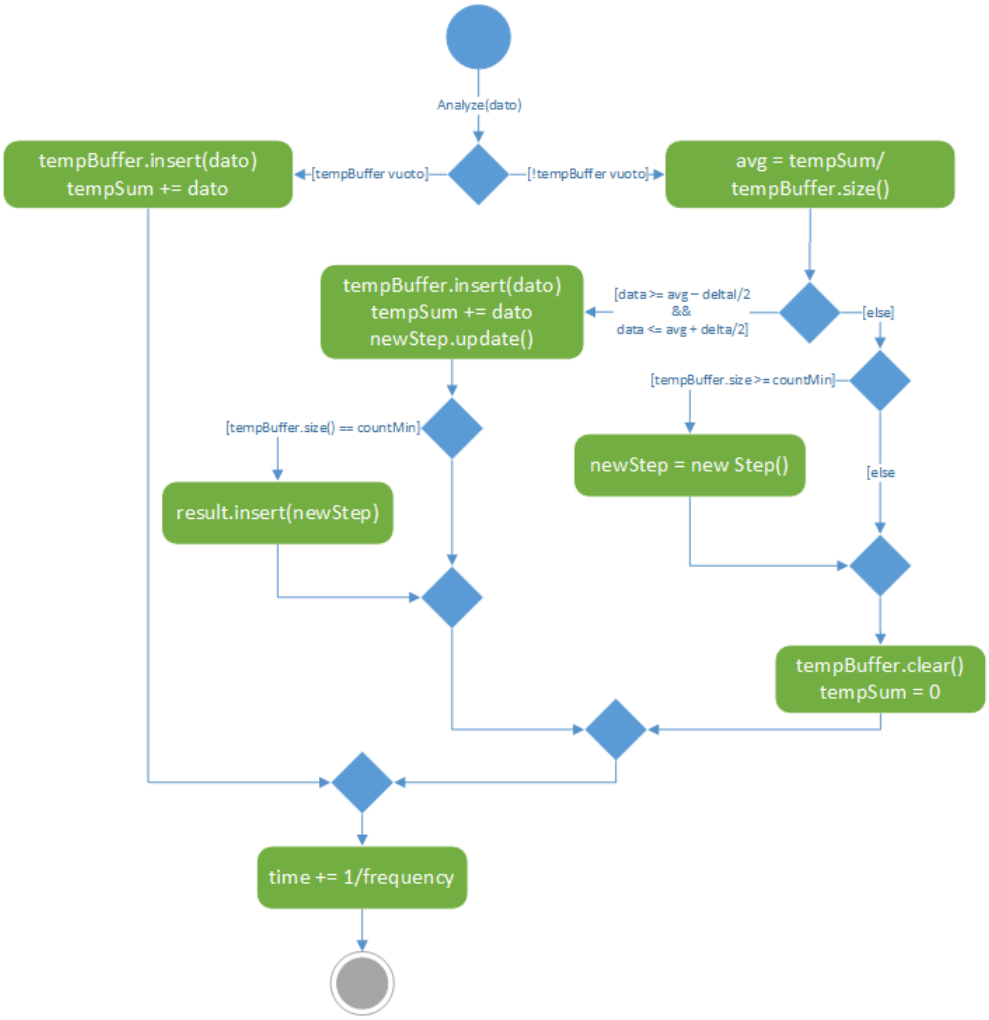


Figura 5.9: Diagramma UML delle attività relativo all'algoritmo utilizzato nell'analisi CurrentStepDetection

Nel costruttore dell'analisi vengono creati gli oggetti *tempBuffer* e *newStep* ed inizializzata la variabile *tempSum* a zero. La struttura *Step* rappresenta una baseline ed è descritta da tre parametri, ovvero corrente media, tempo di inizo e tempo di fine. Tramite l'annotazione *newStep.update()* si indicano le operazione di modifica dei parametri dello step. La variabile time definisce il momento temporale in cui il campione è stato registrato, questo avviene tramite il passaggio del tempo del primo campione da analizzare al costruttore e tramite la frequenza di campionamento (anch'essa passata come parametro) si ricava il tempo di campionamento di quelli successivi.

5.4 Modulo curve fitting

Il modulo di curve fitting deve prevedere di calcolare i parametri di diversi tipi di funzione. Il calcolo dei parametri si basa sulla costruzione di una funzione la quale interpoli il meglio possibile l'insieme di punti dati. Di seguito verranno descritti nel dettaglio gli algoritmi utilizzati per le tre funzioni richieste dall'azienda:

- **Funzione lineare:** per calcolare i due parametri m e b della retta che interpola un insieme di punti dati viene utilizzato l'algoritmo per la regressione lineare:
  $Sxx = \sigma^2(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$
  $Sxy = \sigma(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$
  $m = Sxy/Sxx$
  $b = \bar{y} - m\bar{x}$

- **Funzione esponenziale:** il calcolo dei due parametri A e b della funzione esponenziale, si effettua tramite la linearizzazione della funzione. Il primo passaggio dell'algoritmo è sostituire l'ordinata di ogni punto appartenente ai dati da interpolare con il logaritmo del valore iniziare. Dopo aver applicato il primo passaggio a tutti i dati si avrà un nuovo insieme di dati su cui effettuare l'interpolazione lineare.
  Trovati i valori M e B della retta che interpola il nuovo insieme di dati, si risolvono queste equazione per trovare i parametri della funzione esponenziali:
  $b = M$
  $A = e^B$

Le funzioni non lineari necessitano di algoritmi molto complessi per il calcolo dei parametri di una funzione, quindi per mancanza di tempo sufficiente all'approfondimento di questo campo si è deciso di utilizzare gli algoritmi forniti dalla libreria ALGLIB.

5.5 Modulo filtri digitali

I filtri digitali sono filtri utilizzati nello studio di segnali elettrici, essi permettono di applicare funzioni matematiche sul campioni del segnale per poterne evidenziare determinate caratteristiche. La tipologia di filtri digitali richiesta dall'azienda sono i FIR e corrispondono all'applicazione di una convoluzione discreta sui vari campioni del segnale.
Questa è l'equazione che descrive il funzionamento di un filtro FIR:

$$y[n] = \sum_{i=0}^{N} h_i x[n-i]$$

Per risolvere questa richiesta dell'azienda si è deciso di utilizzare il design pattern Decorator, questo per poter aggiungere la funzionalità di applicare un filtro digitali ai campioni del segnale. Il decorator da implementare è relativo alla classe MultiFileReader (appartenente al Core di EDA) per evitare di modificare la classe originale ma aggiungendole funzionalità.

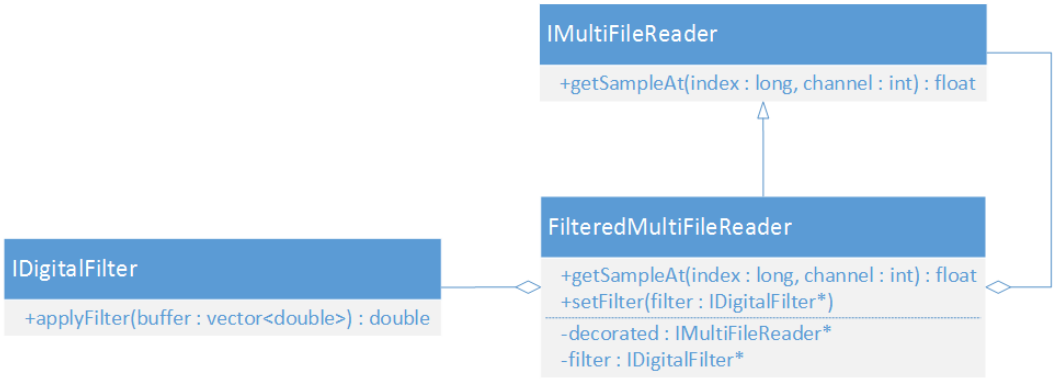


Figura 5.10: Diagramma UML delle classi del pattern Decorator utilizzato nel modulo dei filtri digitali

Capitolo 6

Implementazione

In questo capitolo verranno descritte le scelte implementative di maggiore rilievo. Come già definito nei primi capitoli il linguaggio di riferimento sarà C++.

6.1 AnalysisGenerator

Come indicato nel capitolo 5.3 la classe AnalysisGenerator è la responsabile dell'esecuzione delle varie analisi. Ovviamente la mole di dati da analizzare può variare e di conseguenza quando questa classe viene eseguita su insieme molto grande di dati l'interfaccia utente non è più reattiva. Per evitare che questo accada si è deciso di effettuare l'analisi dei dati su un thread separato in modo tale da non bloccare l'interazione tra utente ed applicazione.

```
#define BUFFER_SIZE 200000

class AnalysisGenerator : public QObject, public QRunnable {
    Q_OBJECT

public:
    AnalysisGenerator():AnalysisGenerator(NULL, NULL, 0, 0){}
    AnalysisGenerator(IAnalizer* analyzer, MultiFileReader* reader, double start,
        double finish, int bufferSize = BUFFER_SIZE) {
        this->start = start*reader->getHeader().frequencyInHz;
        this->finish = finish*reader->getHeader().frequencyInHz;
        this->analyzer = analyzer;
        this->reader = reader;
        this->bufferSize = bufferSize;
    }
 [...]
```

Listato 6.1: Costruttore della classe

```
[...]
public:
    void setAnalyzer(IAnalizer* analyzer) {this->analyzer = analyzer;}
    void setReader(MultiFileReader* reader) {this->reader = reader;}
    void setStart(double start) {this->start = start;}
    void setFinish(double finish) {this->finish = finish;}
    void setBufferSize(int size) {this->bufferSize = size;}

    template<class T>
    T* getAnalyzer() {return dynamic_cast<T*>(analyzer);}
    MultiFileReader* getReader() {return reader;}
    double getStart() {return start;}
    double getFinish() {return finish;}
    int getBufferSize() {return bufferSize;}

    //metodo per far partire il thread con l'analisi
    void startAnalyze(){
        this->setAutoDelete(false);
        QThreadPool::globalInstance()->start(this);
    }
[...]
```

Listato 6.2: Metodi pubblici della classe

```
[...]

private:
    MultiFileReader* reader;
    IAnalizer* analyzer;
    double start;
    double finish;
    float beanSize;
    int bufferSize;

    void run() {
        int count = 0;
        int channelNumber = reader->getchannelsCount() - 1;
        vector<DataWrapper*>* buffer = new vector<DataWrapper*>();
        //se il buffer e' stato riempito almeno una volta
        bool filled = false;

        for(long long i=start;i<finish;i++){
            if(count >= bufferSize) {
                analyzer->analyzeBuffer(buffer);
                count = 0;
                filled = true;
                double percent = (i - start)/(finish - start);
```

```
                //notifica della percetuale di analisi effettuata
                emit onProgress(percent * 100);
            }
            SimpleDataWrapper* sdw;
            if (! filled ) {
                sdw = new SimpleDataWrapper();
                buffer->push_back(sdw);
            } else {
                sdw = dynamic_cast<SimpleDataWrapper*>(buffer->at(count));
            }
            vector<float> values;
            for (int j = 0; j < channelNumber; j++) {
                values.push_back(reader->getSampleAt(i,j));
            }
            sdw->setValues(values);
            //la differenza di potenziale e' l'ultimo canale
            sdw->setVc(reader->getSampleAt(i,channelNumber));

            count++;
        }
        buffer->resize(count);
        if (!buffer->empty()) {
            analyzer->analyzeBuffer(buffer);
        }
        emit onProgress(100);
        emit onFinish(this);
    }

signals:
    void onFinish(AnalysisGenerator*);
    void onProgress(int);
```

Listato 6.3: Metodi e campi privati della classe

6.2 CircularVector

Come è stato definito nel capitolo 5.5 per calcolare il dato filtrato rispetto ad un campione bisogna effettuare una convoluzione discreta, ovvero effettuare una sommatoria di prodotti tra i coefficienti noti e una finestra di dimensioni prefissate di dati precedenti al campione in esame. Le finestre di due campioni adiacenti differiscono per un campione, ovvero si elimina il campione in testa e si aggiunge un nuovo campione in coda. Grazie a questa caratteristica si può rappresentare la finestra di dati con un vettore circolare, ottenendo così un miglioramento in efficienza.

La classe è stata mantenuta generica per poter essere utilizzata con diversi tipi di dati, ma è limitata ai tipi numerici.

```
template <class T>
class CircularVector {
public:
    CircularVector(unsigned int size) {
        vect.resize(size);
      // inizializzazione del vettore
        for(unsigned i = 0; i < size; i++) {
            vect[i] = 0;
        }
      // inizializzazione index
        firstIndex = 0;
    }

    void insert(T value) {
        vect[firstIndex] = value;
      //aggiornamento index
        firstIndex = (firstIndex + 1) % vect.size();
    }

    T at(unsigned int index) {
      //controllo che l'index sia minore della dimensione del vettore
        if (index >= vect.size()) {
            throw runtime_error("out of bound exception");
        }
        unsigned int i = (firstIndex + index) % vect.size();
        return vect[i];
    }

    T operator[](unsigned int index) {
        return at(index);
    }

   //metodo che restituisce la dimensione del vettore circolare
    unsigned int size() {
        return vect.size();
    }
private:
    vector<T> vect;
    unsigned int firstIndex;
};
```

Listato 6.4: Implementazione della classe CircularVector

La classe designata all'utilizzo del CircularVector è il *FilteredMultiFileReader*. Questa classe aggiunge al *MultiFileReader* (classe appartenente al Core di EDA)la funzionalità di impostare un filtro digitale ed applicarlo ai dati letti.

Di seguito verrà mostrata l'implementazione della classe e il relativo utilizzo del *CircularVector*.

```
FilteredMultiFileReader::FilteredMultiFileReader(IMultiFileReader *reader) {
    this->tempBuffer = NULL;
    this->filter = NULL;
    this->reader = reader;
    this->prevIndex = -5;
}
//metodo utilizzato per leggere un dato filtrato
float FilteredMultiFileReader::getSampleAt(long long index, int channel) {
    if ( filter == NULL) {
        throw runtime_error("no filter setted");
    }
   //se il dato in esame e' successivo a quello analizzato precedentemente
   // inserisci in coda ed elimina in testa altrimenti ricrea la finestra dati
    if (index - 1 == prevIndex) {
        tempBuffer->insert(reader->getSampleAt(index, channel));
    } else {
        unsigned int i = index < tempBuffer->size() ? 0 : index - tempBuffer->size()
            + 1;
        for (; i <= index; i++) {
            tempBuffer->insert(reader->getSampleAt(i, channel));
        }
    }
    prevIndex = index;
    return filter ->applyFilter(*tempBuffer);
}
//metodo per recuperare il numero totale di campioni
long long FilteredMultiFileReader::getSize(){
    return reader->getSize();
}
//metodo per recuperare dati presenti nell'header dei file
HEADER FilteredMultiFileReader::getHeader() {
    return reader->getHeader();
}
//metodo per recuperare il numero di canali
int FilteredMultiFileReader::getchannelsCount() {
    return reader->getchannelsCount();
}
//metodo per settare il filtro da applicare ai dati
void FilteredMultiFileReader:: setFilter ( IDigitalFilter * filter ) {
    this->filter = filter ;
    delete this->tempBuffer;
    this->tempBuffer = new CircularVector<double>(filter->windowSize());
}
```

Listato 6.5: Implementazione della classe FilteredMultiFileReader

Capitolo 7

Testing

In questo capitolo verranno mostrati i vari test effettuati sul plug-in sviluppato, per verificare gli algoritmi utilizzati. I test verranno eseguiti su segnali appositamente generati, i quali analizzati tramite una specifica analisi di EDA dovranno restituire un risultato noto a priori o confrontato con quello restituito da Clampfit. La fase di testing è stata prevalentemente incentrata sul modulo di analisi e di curve fitting.

7.1 Test sul modulo di analisi

7.1.1 Test su HistogramAnalysis

In questo capitolo si mostrerà la creazione di un istogramma sui dati acquisiti da file. Verranno messi a confronto Clampfit ed il software dell'azienda Elemnts, EDA. I due istogrammi verranno creati utilizzando gli stessi dati ed impostando lo stesso binSize (a 0,01 pA).

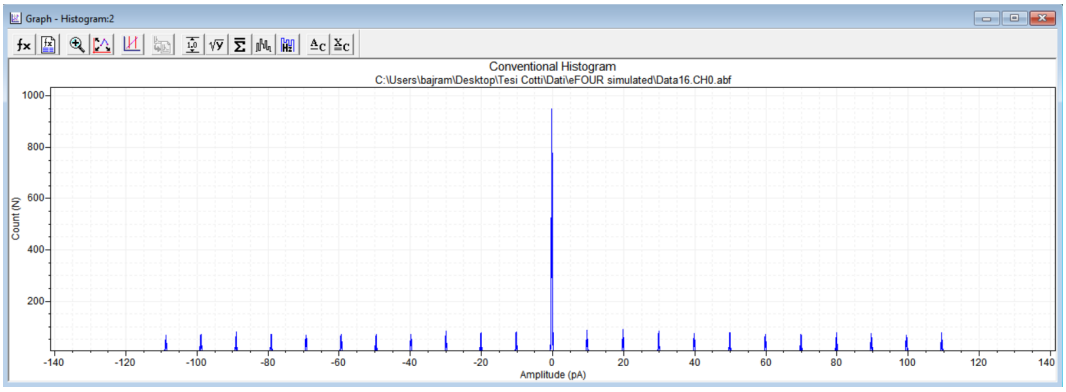


Figura 7.1: Creazione di un istogramma su Clampfit

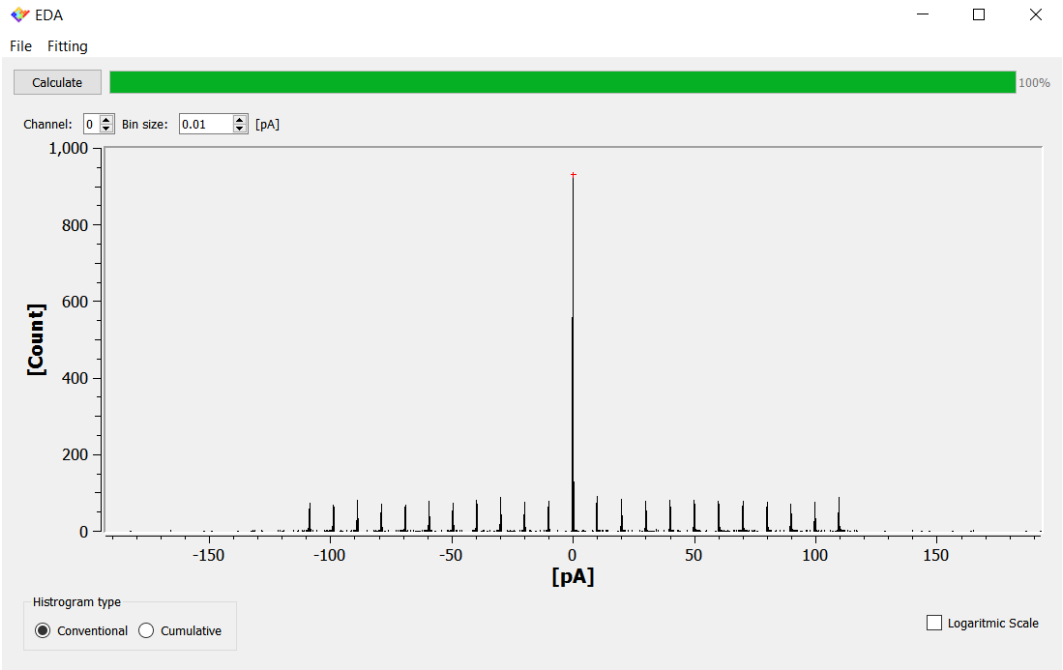


Figura 7.2: Creazione di un istogramma su EDA

Date le due immagini si può notare che l'istogramma generato da EDA coincide perfettamente con quello creato da Clampfit.

7.1.2 Test sull'EventDetection

I test sull'EventDetection non sono stati molteplici, infatti sono stati fatti alcuni controlli incrociati con i dati letti da EDR in real-time, salvati e poi rianalizzati su EDA. I dati sono risultati uguali di conseguenza non sono state fatte ulteriori analisi. Questo è dovuto al fatto di aver sfruttato lo stesso algoritmo di riconoscimento degli eventi presente nel software principale dell'azienda, EDR. Di seguito è mostrato un risultato ottenuto tramite l'analisi EventDetection.

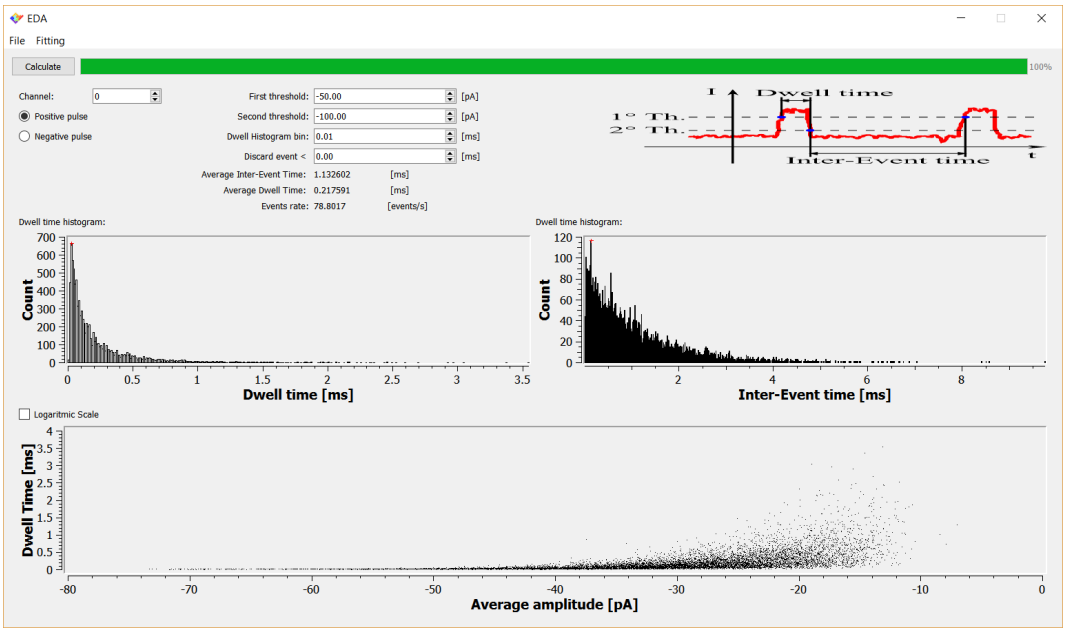


Figura 7.3: Risultato di dati letti da file analizzati tramite l'EventDetection su EDA

7.1.3 Test su FFT

Come descritto nei capitoli precedenti, l'analisi FFT è utilizzata per calcolare la Trasformata Discreta di Fourier. In questo capitolo verrà mostrato il test di correttezza eseguito su questa analisi, prima sottoponendo i dati al software Clampfit e successivamente ad EDA per confrontare i risultati. Entrambi i software analizzano i dati utilizzano un binSize di 32768.

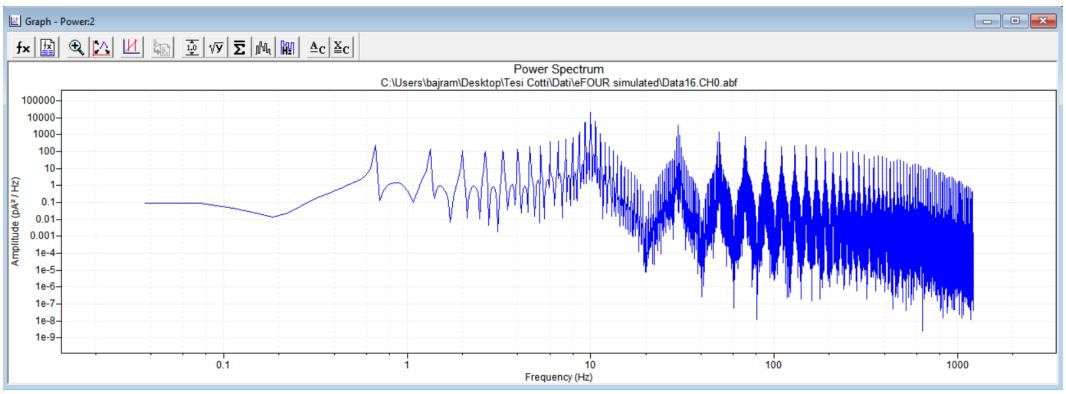


Figura 7.4: Grafito relativo allo spettro del segnale calcolato tramite FFT su Clampfit

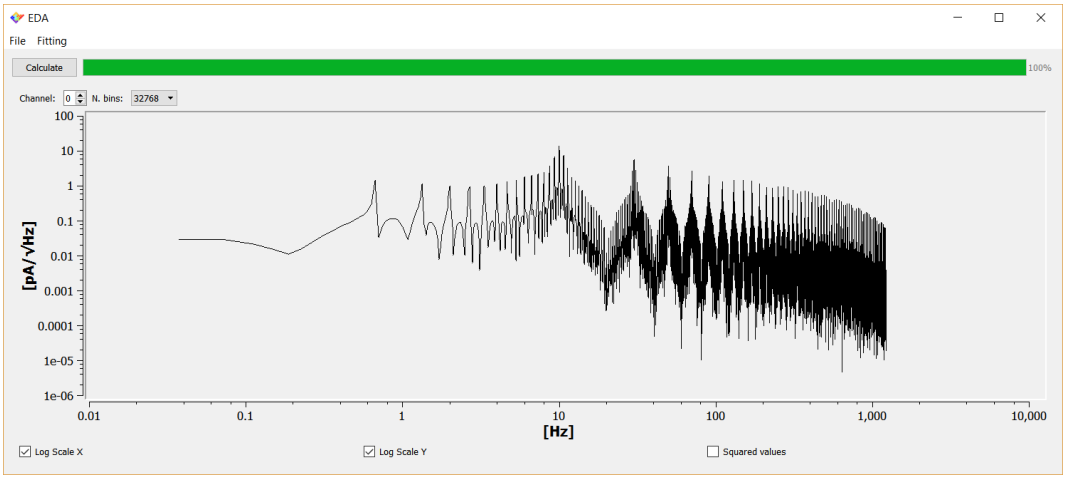


Figura 7.5: Grafito relativo allo spettro del segnale calcolato tramite FFT su EDA

7.1.4 Test sul grafico I/V

Il grafico I/V non è stato soggetto a particolari test di correttezza in quanto inserisce i dati letti da file in un diagramma cartesiano. Inoltre, il calcolo della conduttanza avviene tramite interpolazione lineare dei punti medi, la quale possiede un capitolo a parte (7.2.1) relativo al testing di questa funzionalità.

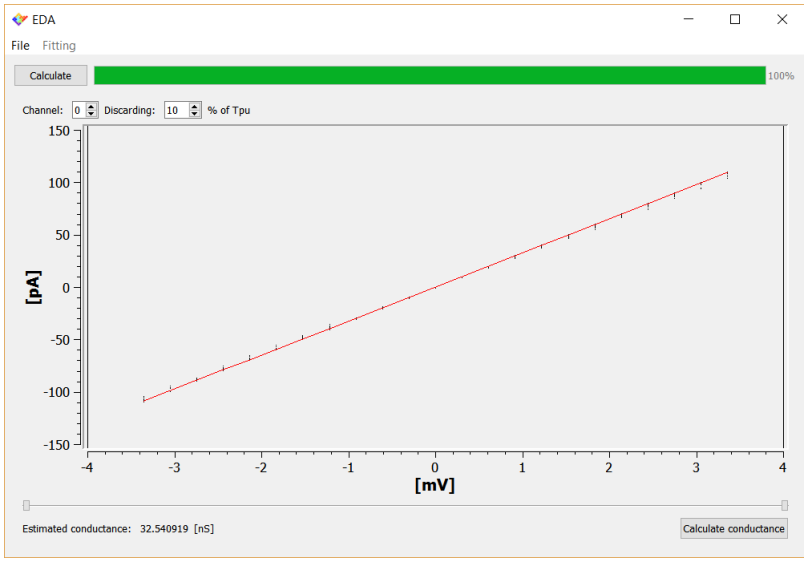


Figura 7.6: Grafico rappresentante il risultato dell'analisi I/V e relativo calcolo della conduttanza

7.1.5 Test su CurrentStepDetection

Il testing su l'analisi CurrentStepDetection è stata effettuata su segnali in input noti e di conseguenza il risultato giusto era verificato dall'utente, a differenza delle altre analisi che risultavano giuste dopo il confronto con Clampfit.
Il segnale generato è stato creato con l'ausilio di EDR, ovvero utilizzando un dispositivo eOne con una resistenza da un $G\omega$ ed applicandovi il protocollo numero 7.

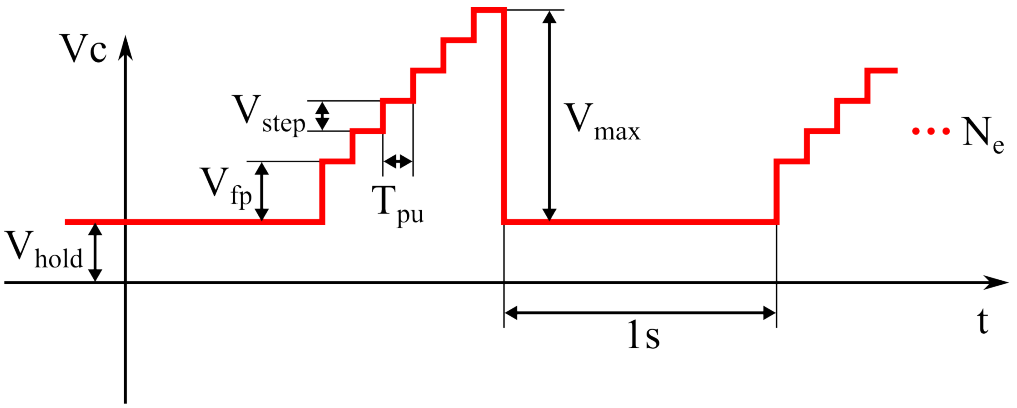


Figura 7.7: Immagine esplicativa del protocollo numero 7 di EDR

I dati relativi al protocollo utili per stimare il risultato finale di questa analisi sono $V_{fp}, V_{step}, T_{pu} e V_{max}$. Nel segnale utilizzato per il testing i parametri citati avevano il seguente valore:

- $V_{fp} = 20pA$
- $V_{step} = 20pA$
- $T_{pu} = 20ms$
- $V_{max} = 100pA$

In base a questi dati possiamo dedurre che presa in analisi una sola "scalinata" il risultato corretto sarà:

- $AvgCurrent \approx 20pA$
- $AvgCurrent \approx 20ms$
- $Baselinecount = 5$

I primi due dati potrebbero differire dal valore stimato fino ad un paio di pA perché il segnale generato presenta picchi dovuti alla presenza della resistenza, durante le analisi su dati rilevanti questi picchi non vi sono di conseguenza i valori saranno maggiormente precisi. Di seguito verranno mostrati il segnale principale su cui si è effettuata l'analisi e il relativo risultato.

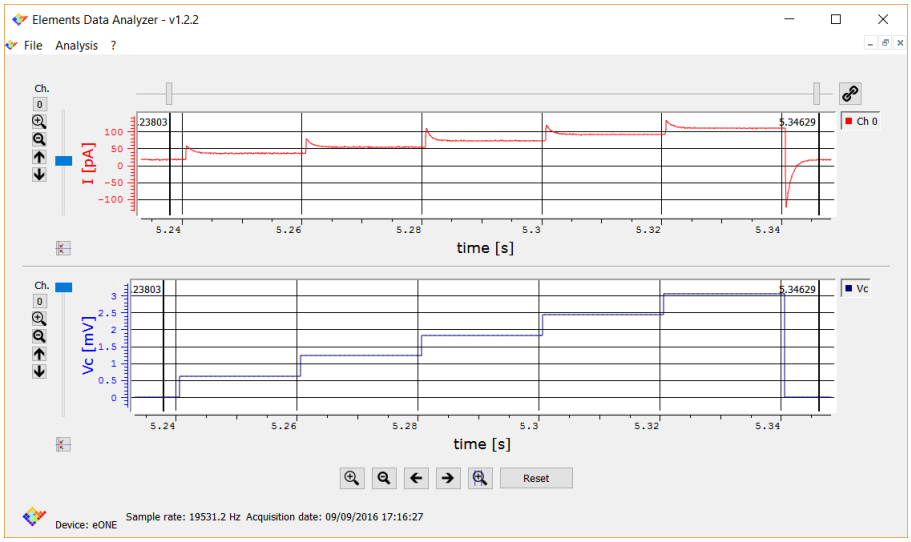


Figura 7.8: Segnale generato da EDR per il testing dell'analisi CurrentStepDetection

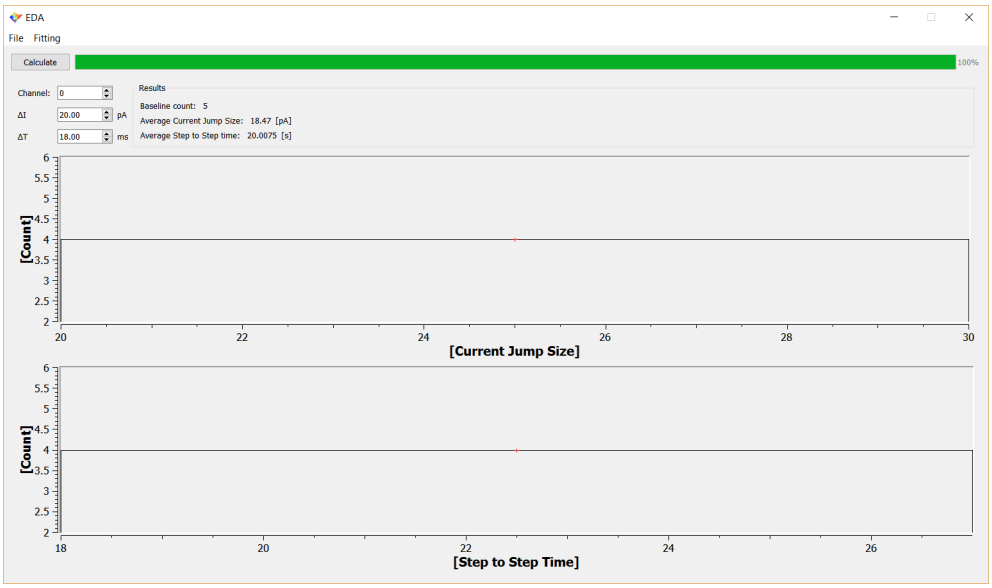


Figura 7.9: Risultato dell'analisi CurrentStepDetection sul segnale generato

Come era già stato previsto i dati differiscono leggermente per il motivo sopracitato, di conseguenza l'analisi risulta corretta. In questo caso gli istogrammi non sono molto elaborati ma confermano la correttezza del risultato in quanto tutti gli step presenti (n. baseline - 1) hanno la stessa durata in tempo e differenza di ampiezza.

7.2 Test sul modulo di curve fitting

7.2.1 Test funzione lineare

Di seguito è possibile visualizzare un test per il controllo della correttezza del risultato calcolato dall'applicazione di fitting lineare sui dati, ovvero interpolazione dei dati tramite funzione lineare. Il fitting è stato calcolato sugli stessi dati sia su Clampfit che su EDA.

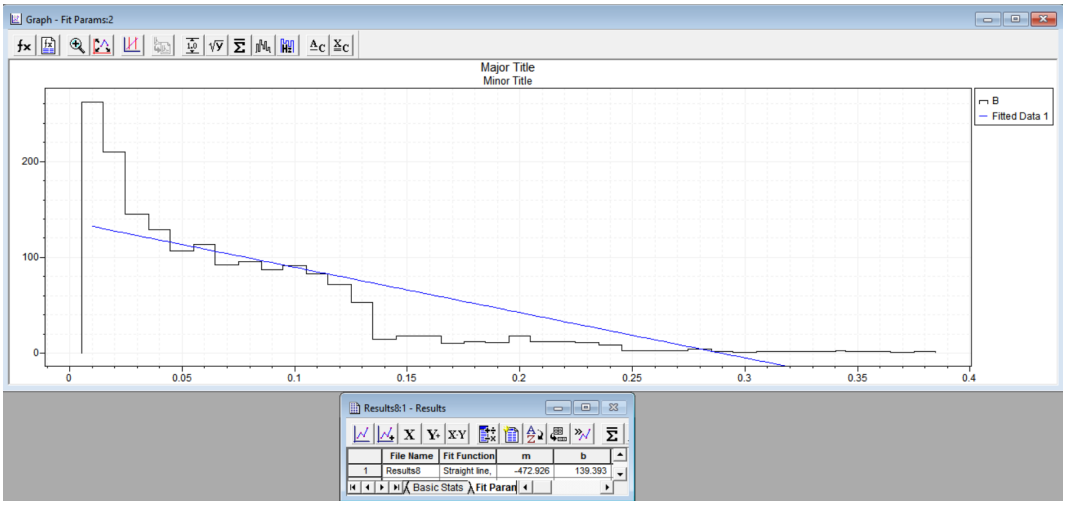


Figura 7.10: Applicazione di fitting lineare ad un istogramma su Clampfit

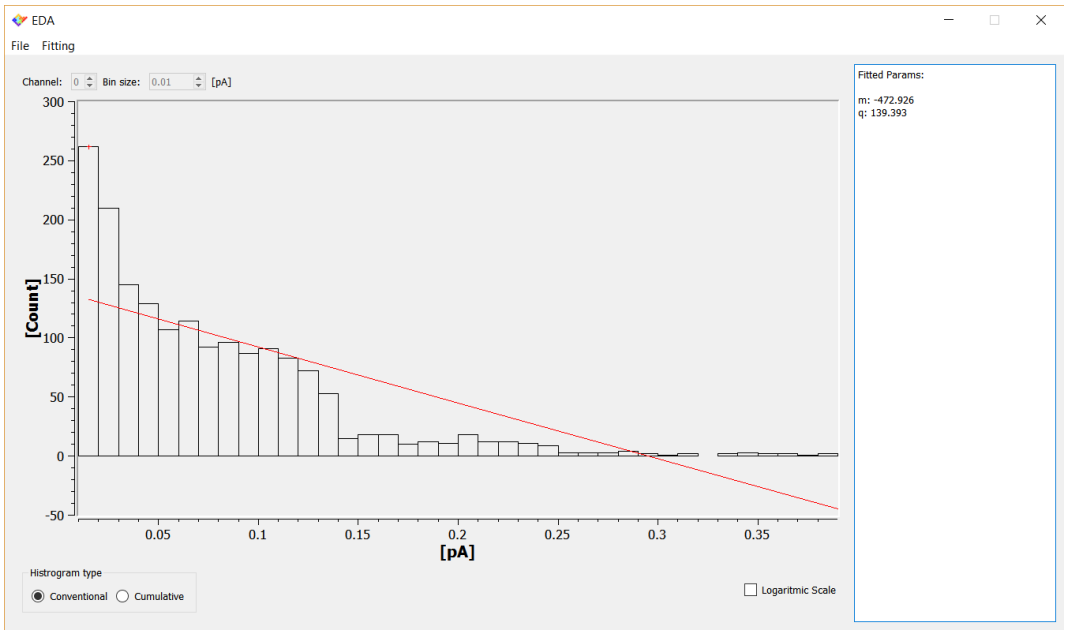


Figura 7.11: Applicazione di fitting lineare ad un istogramma su EDA

Come si può notare i risultati del fitting su i due software sono equivalenti, quindi l'algoritmo utilizzato può essere considerato corretto.
Il fitting esponenziale non è stato soggetto a particolari test di correttezza in quanto, come spiegato nel capitolo 5.4, si è implementato linearizzando la funzione ed applicando l'interpolazione lineare che abbiamo appena dimostrato essere corretta.

7.2.2 Test sommatoria di gaussiane

In questo capitolo verranno fatti test relativi al fitting che si basa una sommatoria di gaussiane come funzione da interpolare. I test che verranno mostrati sono stati effettuati su dati che presentano una gaussiana e successivamente su altri i quali ne contengono due. Come nel caso del fitting lineare gli stessi dati vengono elaborati prima da Clampfit e successivamente da EDA per confrontare i risultati.

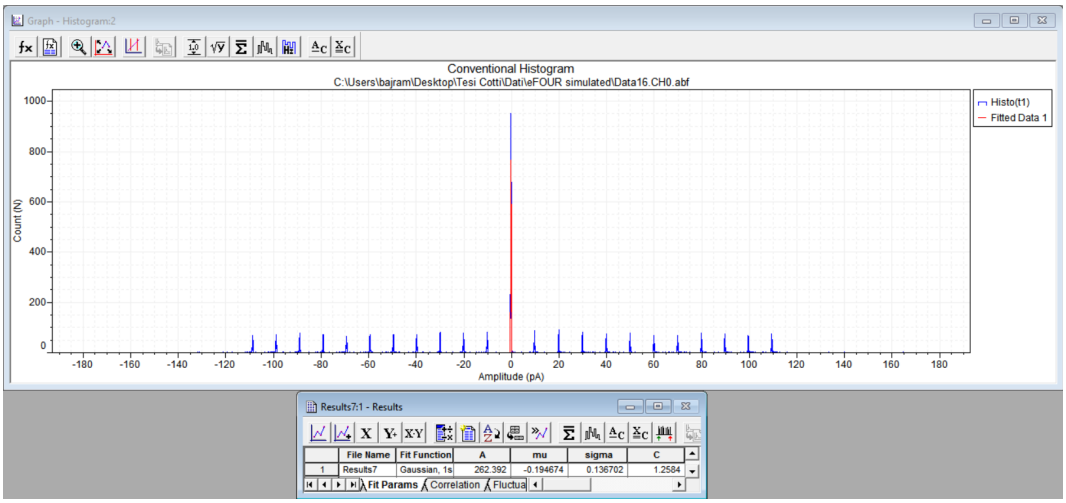


Figura 7.12: Applicazione di interpolazione gaussiana (una campana) ad un istogramma su Clampfit

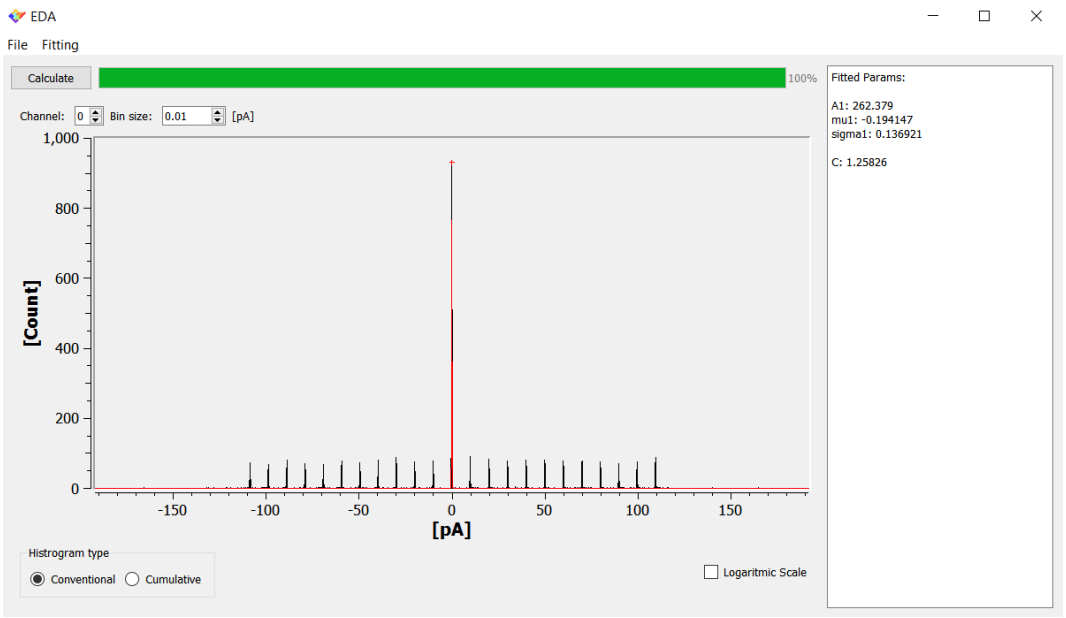


Figura 7.13: Applicazione di interpolazione gaussiana (una campana) ad un istogramma su EDA

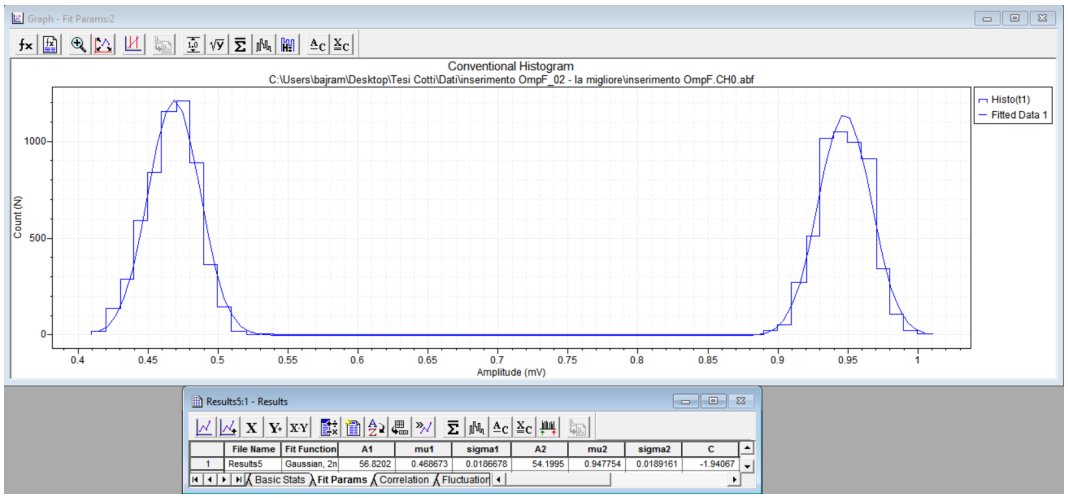


Figura 7.14: Applicazione di interpolazione gaussiana (due campane) ad un istogramma su Clampfit

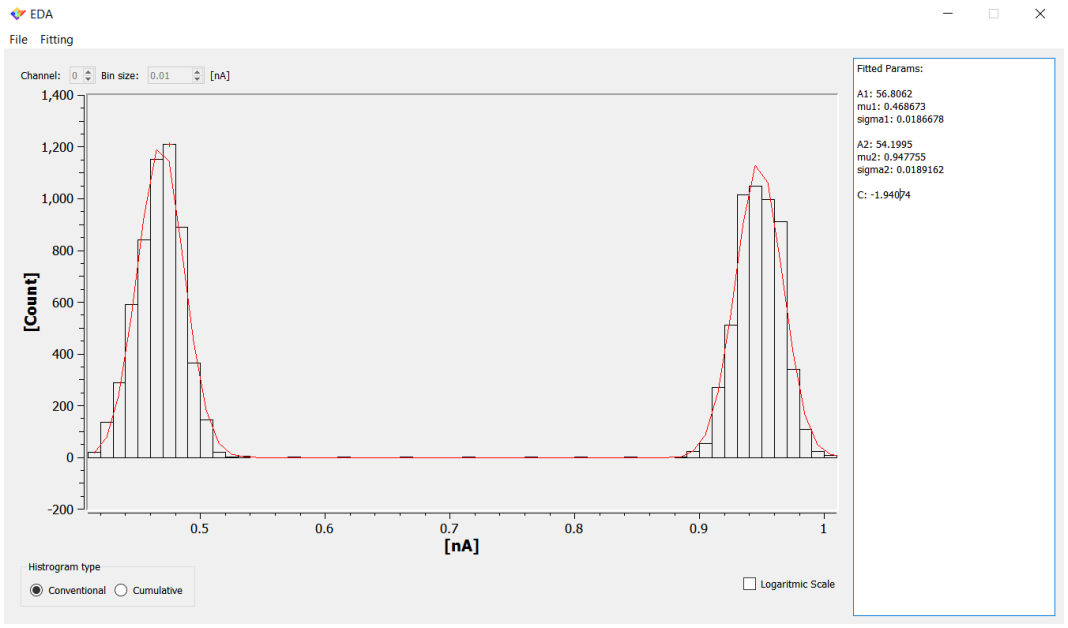


Figura 7.15: Applicazione di interpolazione gaussiana (due campana) ad un istogramma su EDA

Confrontando i risultati mostrati nelle immagini si può notare che i risultati coincidono, di conseguenza si può affermare che la funzionalità in esame rispetta le specifiche.

Conclusioni

Il progetto di tesi è stata un'esperienza davvero interessante e stimolante, inoltre questo primo approccio al mondo del lavoro in ambito informatico è stato molto formativo. Questa esperienza, infatti, mi ha portato a conoscenza di tematiche a me sconosciute (come per esempio cosa sono i canali ionici e quali sono le analisi maggiormente utilizzate nel campo dell'elettrofisiologia) e ad applicare le conoscenze apprese nei corsi universitari frequentati.

Il sostegno, la fiducia e la disponibilità dei tecnici di Elements sono stati fondamentali per la buona riuscita del plug-in da loro richiesto. Il continuo dialogo e confronto con l'azienda in tutte le fasi di sviluppo del software, mi ha aiutato ad effettuare correzioni sul progetto in tempi brevi realizzando così un software che soddisfi i requisiti da loro espressi.

I punti di forza del software sono la scalabilità e la correttezza, questo perché sono le caratteristiche principali su cui si è basata la progettazione dell'intero sistema software. Inoltre i test sono risultati di fondamentale importanza per verificare la correttezza dell'applicazione.

Le conoscenze acquisite durante le lezioni universitarie hanno svolto un ruolo importante per lo sviluppo del software, in particolare *Programmazione a Oggetti* per il paradigma a oggetti su cui si basa l'intero progetto e tecniche di programmazione avanzate come l'utilizzo dei vari design pattern ed infine *Algoritmi e strutture dati* per i calcoli dei vari costi computazionali, in modo da garantire tempi ragionevoli per l'elaborazione di una mole discretamente elevata di dati.

Il software di analisi in post-processing dell'azienda, Elements Data Analyzer (EDA), ora grazie al plug-in oggetto di questa tesi è in grado di usufruire di un discreto numero di funzionalità aggiuntive e possiede un modulo facilmente estendibile che permette di inserirne altre in maniera semplice e veloce.

Sviluppi Futuri

Il modulo software progettato è una versione base, come specificato sin dal principio dall'azienda. Possibili ampliamenti sono previsti nel sotto-modulo di analisi, quali inserimento di nuove tipologie di analisi e aggiornamento degli algoritmi utilizzati (questo dovuto alla continua ricerca nel campo dell'elettrofisiologia). Anche il sotto-modulo di curve fitting potrebbero subire ampliamenti futuri come aggiunta di nuove funzioni da calcolare o l'utilizzo di un'altra libreria per l'ottimizzazione dei parametri. Per questo momento il sotto-modulo dei filtri digitali sembra non prevedere grosse modifiche, ma la struttura scelta è già in grado di supportare futuri upgrade.

Ringraziamenti

I miei ringraziamenti vanno ai tecnici di Elements S.r.l., Federico, Marco e Michele per la bellissima esperienza offertami, la massima disponibilità e totale fiducia nei miei confronti. Un ringraziamento particolare va al Dott. Bajram Hushi, entrato da un anno a far parte dello staff di Elements, per la sua disponibilità e gli utilissimi consigli durante lo sviluppo del software. Vorrei ringraziare il Professor Mirko Viroli per gli ottimi consigli in fase di stesura della tesi. Inoltre, i miei ringraziamenti vanno a tutte le persone che mi sono state vicine durante il mio percorso di studio e mi hanno supportato durante l'esperienza di tesi.

Elenco delle figure

Bibliografia

[1] Elements http://elements-ic.com/

[2] Wikipedia https://www.wikipedia.org/

[3] QT https://wiki.qt.io/

[4] Qwt http://qwt.sourceforge.net/

[5] ALGLIB http://www.alglib.net/

[6] Axon Instruments Inc. pClamp10 User manual

[7] Axon Instruments Inc. ABF Help