UNIVERSITÀ DI PISA

Laurea Specialistica in
Ingegneria dell'Automazione

AUTOMATIZZAZIONE DI PROCESSI PRODUTTIVI

Relatore accademico:
Prof. Sergio Saponara

Relatore aziendale:
Dott. Emmanuele Angione

Candidato:
Antonino Crivello

ANNO ACCADEMICO 2012/2013

Et tota spes mea non nisi in magna valde misericordia tua.
Da quod iubes et iube quod vis

Le confessioni, Sant'Agostino

ABSTRACT

Nell'ambito della gestione della catena produttiva di dispositivi biomedicali, è di fondamentale importanza l'automazione delle fasi di test e di collaudo dei dispositivi stessi, al fine di diminuire gli errori e risparmiare tempo/uomo.

In questa tesi è descritto un insieme di software progettati e implementati per il dispositivo biomedicale denominato Winpack, sviluppato e prodotto in outsourcing dalla WINMedical S.r.l. Il Winpack si compone di un modulo base che si occupa della comunicazione tra i vari moduli e uno o più moduli accessori che realizzano diverse funzionalità quali, ad esempio, la misurazione della temperatura corporea o dell'ossigenazione del sangue. La necessità di sviluppare questi software deriva dal fatto che WINMedical mantiene al suo interno il controllo qualità al 100% di una produzione interamente realizzata in outsourcing.

Il lavoro consiste di tre software distinti, sviluppati in Java (JDK7): due per la fase di test e uno per la fase di collaudo del Winpack e di tutti i moduli accessori. L'approccio è di tipo *multi-threading* per migliorare le prestazioni e l'efficienza nell'utilizzo. I software di test hanno risolto le problematiche legate all'utilizzo di strumenti proprietari che ponevano limitazioni forti al tipo di piattaforma utilizzabile poiché era necessario scegliere piattaforme con licenze di utilizzo attive. Il risparmio ottenuto con il software di collaudo è stato, in termini di tempo, tra i 50% e il 70%, a seconda del modulo da collaudare e ha ridotto del 90% il numero totale degli errori commessi nei report di collaudo.

INDICE

1. Introduzione e obiettivo *pag. 6*
2. Analisi preliminare *8*
3. Soluzione proposta *12*
   3.1 Workflow del Kit *12*
   3.2 Test per l'integrità dei dati - 3LDbExtractor *15*
      3.2.1 Riprogrammazione firmware *19*
      3.2.2 Interfaccia utente di 3LdbExtractor *20*
   3.3 Test di stabilità del sistema - WincsTotalcon *22*
      3.3.1 Utilizzo del Syslog *22*
      3.3.2 Interfaccia utente di WincsTotalCon *24*
   3.4 Collaudo dei moduli - WinCollaudi *28*
      3.4.1 *Message Queue* *29*
      3.4.2 Estensione Winmultiserver *31*
      3.4.3 *Multi-thread* - Modularità - Scalabilità *33*
      3.4.4 Interfaccia utente di WinCollaudi *36*
4. Risultati ottenuti *45*
Appendice A *46*
Appendice B *62*
Appendice C *89*
Appendice D *97*

Bibliografia *103*
Ringraziamenti *104*

INDICE DELLE FIGURE

Figura 2.1 - Workflow sistema WINMedical *pag. 9*
Figura 2.2 - Esempio di verbale di collaudo *10*
Figura 2.3 - Esempio di report riassuntivo di una sessione di collaudo *11*
Figura 3.1 - Workflow del sistema dopo l'aggiunta del kit *12*
Figura 3.2 - Workflow del software 3LdbExtractor *13*
Figura 3.3 - Workflow del software WincsTotalCon *14*
Figura 3.4 - Workflow del software WinCollaudi *14*
Figura 3.5 - Grafico per le tracce dell'Ecg4 *16*
Figura 3.6 - Memorizzazione dell'Ecg4 in MongoDb *17*
Figura 3.7 - Esempio di report per 3LdbExtractor *19*
Figura 3.8 - Effetto della riprogrammazione del firmware di Ecg4 *20*
Figura 3.9 - Interfaccia utente di 3LdbExtractor *21*
Figura 3.10 - Syslog filtrato con grep "totalcon" *23*
Figura 3.11 - Interfaccia utente di WincsTotalCon *25*
Figura 3.12 - Interfaccia "Riepilogo" di WincsTotalCon *25*
Figura 3.13 - Interfaccia "Percentuali" di WincsTotalCon *26*
Figura 3.14 - Interfaccia "Andamento" di WincsTotalCon *28*
Figura 3.15 - Schema del *Message Queue* *29*
Figura 3.16 - Schema del Request - Reply *31*
Figura 3.17 - Struttura di WinCollaudi *36*
Figura 3.18 - Interfaccia principale di WinCollaudi *37*
Figura 3.19 - Grafico dell'heart rate *39*
Figura 3.20 - Frame di collaudo di Ecg *41*
Figura 3.21 - Report di collaudo di CaricaBatteria *44*
Figura 3.22 - Report di sessione *44*

1. INTRODUZIONE E OBIETTIVO

La WINMedical sviluppa e produce dispositivi integrati per il monitoraggio wireless di alcuni dei principali parametri fisiologici, tra cui: Heart Rate, Pulsossimetria, ECG a 4 derivazioni, temperatura corporea, posizione.

Il dispositivo che raccoglie e trasmette wireless i parametri vitali è modulare ed il suo "corpo" centrale è denominato wincard. Esso presenta sei slot per i moduli aggiuntivi (tutti plug and play), i quali, collegandosi al modulo base, rilevano i dati e li trasmettono ad un server centrale. Poiché WINMedical mantiene al suo interno il controllo qualità al 100% di una produzione interamente realizzata in outsourcing, nell'ambito della gestione della catena produttiva dei moduli è nata la necessità di automatizzare due importanti aspetti della catena di produzione: la fase di test e la fase di collaudo dei dispositivi biomedicali.

È doveroso ricordare inoltre che i dispositivi in questione sono dispositivi biomedicali CE0434 secondo la dir. 93/42/EEC. L'efficacia e l'efficienza dei test e dei collaudi sono quindi fondamentali per mantenere alti gli standard di qualità di un'azienda, la WINMedical, avente una sistema di qualità conforme ISO13485 e ISO9001.

Il processo aziendale antecedente allo sviluppo effettuato in questo lavoro di tesi prevedeva alcuni test volti a determinare la stabilità delle connessioni dei Winpack verso il server con cui dialogano. Particolare attenzione va rivolta all'integrità dei dati contenuti nel database interno al server, cioè di tutti i dati raccolti dal modulo base e dal firmware con cui è configurato e inviati, attraverso il modulo bluetooth e in ottemperanza ad un interno protocollo di comunicazione, al data collector.

I test sulla stabilità delle connessioni dei Winpack al server e sull'integrità dei dati in arrivo dal Winpack sfruttavano routine create ad hoc con l'ausilio di Matlab®. L'obiettivo posto per il miglioramento di questi due importanti test da effettuare sul dispositivo è di evitare l'uso di un software con licenza su un unico pc aziendale e creare una soluzione che permetta di svolgere gli stessi test con un software interno, open-source e cross-platform, che possa eseguito da qualunque terminale aziendale.

Per la fase di collaudo invece la riposta dei moduli a determinati segnali d'ingresso, generati all'occorrenza da opportuni generatori di funzione e con l'ausilio degli oscilloscopi, veniva elaborata da un software sviluppato internamente all'azienda che si limitava a visualizzali, mentre era compito dell'operatore preposto ai collaudi analizzare

e interpretare i dati visualizzati per produrre la documentazione necessaria all'approvazione o al fallimento di un Winpack o di uno dei suoi moduli accessori.

Analizzando l'esigenza aziendale di trovare una soluzione per rendere più snelle e accurate le fasi di test e collaudo, si è scelto di sviluppare strumenti software che assicurino un più alto margine di confidenza del risultato ed un più basso effort dell'operatore che esegue il collaudo.

Il lavoro di tesi proposto ha quindi l'obiettivo di fornire all'azienda un set di software capaci di automatizzare le procedure, migliorando sia l'efficienza che l'efficacia del processo.

2. ANALISI PRELIMINARE

Per fase di test si intende la fase in cui vengono svolte le prove da parte del gruppo di sviluppo durante la realizzazione del progetto. In questo caso il progetto è in continua evoluzione poiché viene migliorata sia la componente server preposta a ricevere i dati lato server Winmultiserver sia il firmware interno installato sul modulo base (Winpack). Durante il collaudo invece l'azienda, che "diventa" cliente del suo fornitore dato che i moduli vengono fisicamente realizzati in outsourcing, deve garantire la conformità del prodotto finale rispetto ai requisiti espressi. L'esito di ogni collaudo è formalizzato da un apposito verbale di collaudo, automatizzato, come vedremo, proprio dal software WinCollaudi. Il collaudo produce i seguenti prodotti di fase:

- Piano di collaudo
- Specifiche di test
- Prodotto collaudato secondo il piano
- Verbale di collaudo

Dall'analisi preliminare delle procedure di collaudo e delle strumentazioni utilizzate si è evinta la necessita di uno sviluppo multipiattaforma, perché il software WINMedical è sviluppato per sistemi Unix (in questo sistema operativo è configurato il database che contiene i dati in arrivo dal Winpack e molti dei server utilizzati internamente hanno quindi installata una versione di Ubuntu Server con Xfce), ma i collaudi invece, eseguiti da diversi operatori, vengono effettuati indistintamente su sistemi operativi Windows o distribuzioni Unix. Per i motivi sopracitati si è scelto di utilizzare Java come linguaggio di programmazione [1].

La figura 2.1 mostra l'architettura del sistema WINMedical. Il dispositivo Winpack comunica, attraverso il protocollo Bluetooth, con il Winmultiserver, che è un modulo software capace appunto di comunicare con il modulo wireless inserito nel Winpack. È inoltre preposto, d'accordo con il protocollo di comunicazione, al corretto scambio dei segnali di Ack/Nack per sincronizzare e regolare i dati provenienti dal Wincard (che è, come già detto, il blocco base del Winpack), oltre che ad inoltrarli al modulo software preposto alla memorizzazione all'interno del database (MongoDB). È compito dell'applicazione Web-client restituire i dati al personale sanitario che ha in uso i dispositivi telemetrici.

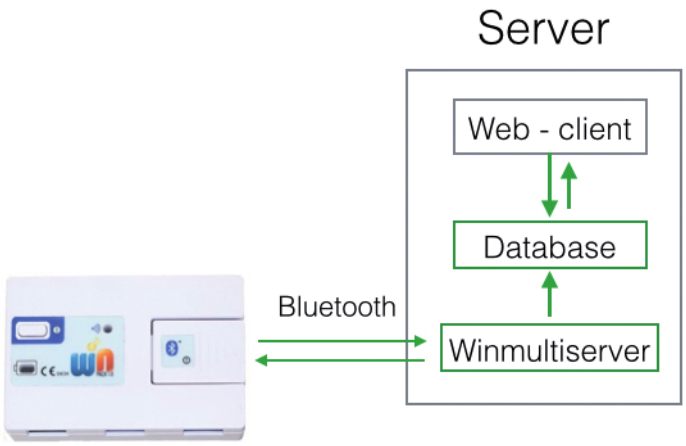


Figura 2.1 - Workflow sistema WINMedical

Nei sistemi Unix molte delle informazioni prodotte dai software vengono scritte sfruttando il log di sistema: sfruttando questa possibilità, Winmultiserver memorizza sul log i seriali dei Winpack collegati correttamente con il Server.

Esposti ed analizzati gli strumenti software già esistenti all'interno dell'architettura WINMedical sono stati individuati tre elementi essenziali per il raggiungimento degli obiettivi prefissati:

- attraverso l'analisi del syslog è possibile capire quale sia stato l'andamento temporale delle connessioni tra i Winpack e il server;

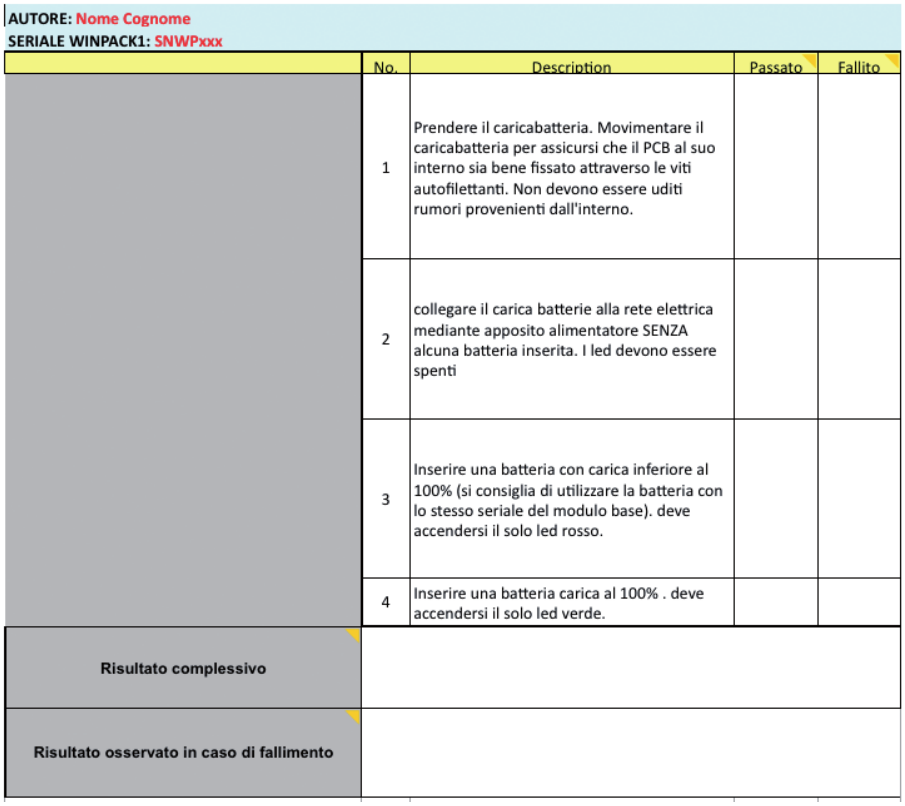
- attraverso l'utilizzo di query al database dove vengono memorizzati i dati è possibile capire se i dati in esso contenuti sono corretti o se alcuni pacchetti sono andati irrimediabilmente perduti;

- le procedure di collaudo sono già state dettagliate: sarà quindi necessario automatizzarle sviluppando un'interfaccia capace di aiutare l'operatore a non commettere errori, valutando autonomamente, quando possibile, i dati provenienti dal Winpack; ciò può avvenire ponendosi a valle del modulo Winmultiserver e comunicando con esso in tempo reale.

Descriviamo ora nel dettaglio quali siano le procedure di collaudo sopracitate. La documentazione da produrre a cui si fa riferimento è così composta:

- Spreadsheet contenente: dati del collaudatore, data del test, elenco degli step da eseguire e per ogni step l'esito (passato o fallito). Infine il risultato complessivo che

esprime se il collaudo è superato o meno ed un campo note per dettagli aggiuntivi dell'operatore o del responsabile di qualità designato. Il nome del file deve inoltre avere una ben precisa struttura, per meglio essere organizzato e conservato in archivio. Più nel dettaglio, il nome deve avere una struttura TipoModulo_Numerodilotto_seriale_data_collaudatore. In figura 2.2 è riportato un esempio del report di collaudo (che è diverso per ogni modulo).

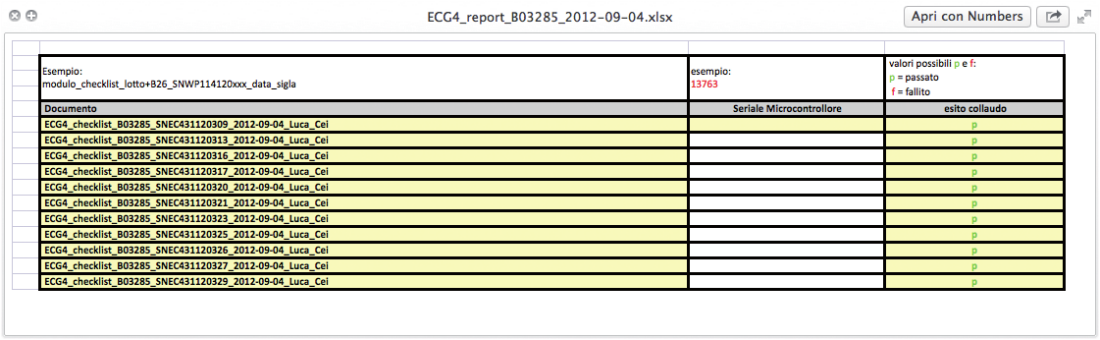


AUTORE: Nome Cognome
SERIALE WINPACK1: SNWPxxx

	No.	Description	Passato	Fallito
	1	Prendere il caricabatteria. Movimentare il caricabatteria per assicursi che il PCB al suo interno sia bene fissato attraverso le viti autofilettanti. Non devono essere uditi rumori provenienti dall'interno.		
	2	collegare il carica batterie alla rete elettrica mediante apposito alimentatore SENZA alcuna batteria inserita. I led devono essere spenti		
	3	Inserire una batteria con carica inferiore al 100% (si consiglia di utilizzare la batteria con lo stesso seriale del modulo base). deve accendersi il solo led rosso.		
	4	Inserire una batteria carica al 100% . deve accendersi il solo led verde.		
Risultato complessivo				
Risultato osservato in caso di fallimento				

Figura 2.2 - Esempio di verbale di collaudo

- Spreadsheet contenente un riepilogo di tutti i collaudi effettuati in una determinata sessione, che possa quindi dare, in maniera immediata, un feedback sull'esito delle operazioni su ciascun modulo. Esso infatti contiene: il riferimento al file di collaudo per il singolo modulo, il seriale del microcontrollore con cui è stato collaudato e l'esito dell'operazione di collaudo. In figura 2.3 è riportato un esempio del file che riassume una sessione di collaudo: ogni entry del file riporta il riferimento ad un report di collaudo ed esprime, per ogni modulo collaudato, l'esito finale.



Esempio: modulo_checklist_lotto+B26_SNWP114120xxx_data_sigla	esempio: 13763	valori possibili p e f: p = passato f = fallito
Documento	**Seriale Microcontrollore**	**esito collaudo**
ECG4_checklist_B03285_SNEC431120309_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120313_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120316_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120317_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120320_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120321_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120323_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120325_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120326_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120327_2012-09-04_Luca_Cei		p
ECG4_checklist_B03285_SNEC431120329_2012-09-04_Luca_Cei		p

Figura 2.3 - Esempio di report riassuntivo di una sessione di collaudo

3. SOLUZIONE PROPOSTA

In questo capitolo verranno descritti nel dettaglio i tre software sviluppati come oggetto di questo lavoro di tesi. Per ciascuno è descritto il workflow e sono dettagliate le relative funzionalità. In figura 3.1 è mostrato come la soluzione proposta si instaura con gli elementi già presenti nel sistema. In rosso sono evidenziati i moduli software introdotti, in verde invece i moduli esistenti con cui i moduli scambiano informazioni e processi.

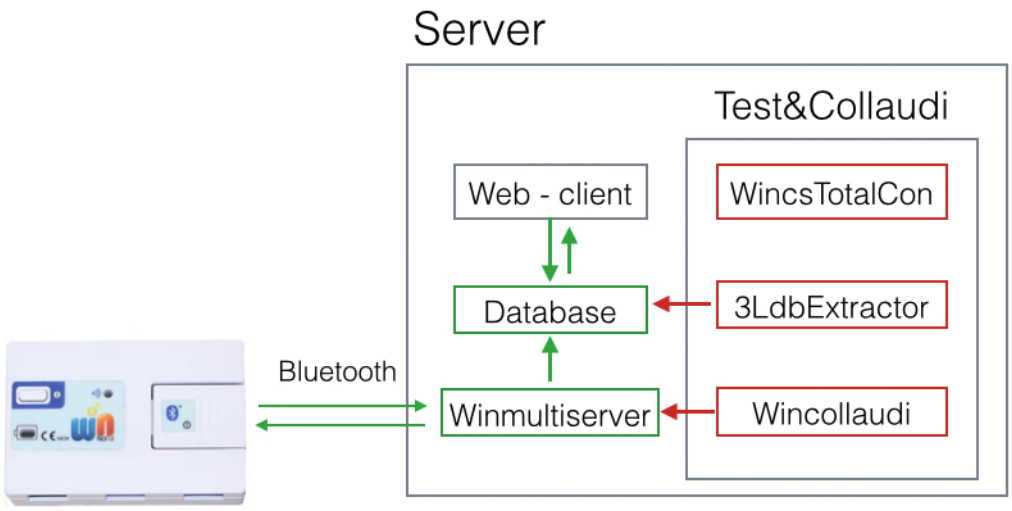


Figura 3.1 - Workflow del sistema dopo l'aggiunta del kit

L'insieme dei software prende il nome di kit "Test&Collaudi".

3.1 Workflow del Kit

Le figure di seguito proposte mostrano in dettaglio i workflow delle tre applicazioni software che compongono il kit. Osservando le figure da sinistra verso destra è possibile vedere, in sequenza, input, core d'elaborazione e output previsti.

Ogni workflow verrà descritto e analizzato più nel dettaglio nei paragrafi seguenti, dove verranno analizzati gli aspetti principali del software, le scelte effettuate nell'implementare le soluzioni necessarie e i risultati ottenuti sia nel front-end sia nel back-end.

Nelle figure successive il verde è usato per i moduli già esistenti, mentre il rosso per i moduli creati come oggetto di questo lavoro di tesi.

In figura 3.2 è mostrato il workflow del software 3LdbExtractor.

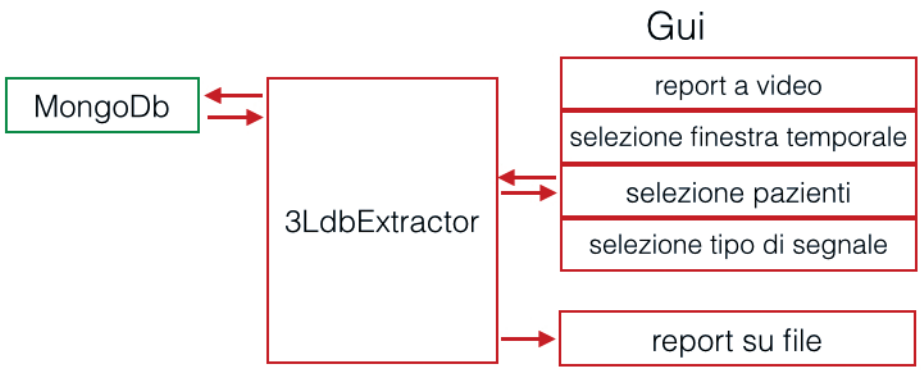


Figura 3.2 - Workflow del software 3LdbExtractor

L'input del modulo 3LdbExtractor è costituito dai dati contentuti nel database al quale l'interno sistema si appoggia. Si tratta di un database non relazionale, implementato sfruttando le potenzialità di MongoDB. A partire dagli input forniti al software dal tecnico che effettua il test attraverso un'apposita nterfaccia grafica 3LdbExtractor costruisce la query e interroga il DB per ottenere i dati di interesse. I dati così ottenuti sono scritti su un file così da poter essere consultati anche successivamente e poi visualizzati all'interno della GUI.

La figura 3.3 mostra il workflow del software WincsTotalCon. I dati di input in questo caso vengono forniti dall'operatore attraverso la GUI. Essi sono rappresentati da un file di testo, risultato di un'opportuna operazione di filtraggio sul file di Log che si vuole utilizzare e popolato per tutta la durata del test di cui si vuole analizzare a posteriori la stabilità in termini di connessioni totali dei Winpack al server.

Il file di Log già filtrato deve essere creato dall'operatore prima di utilizzare WincsTotalcon. Il software analizza questo file di testo e, riga dopo riga, ricava il numero di connessioni totali e il tempo in cui ogni connessione si è presentata rispetto alla durata totale del test.

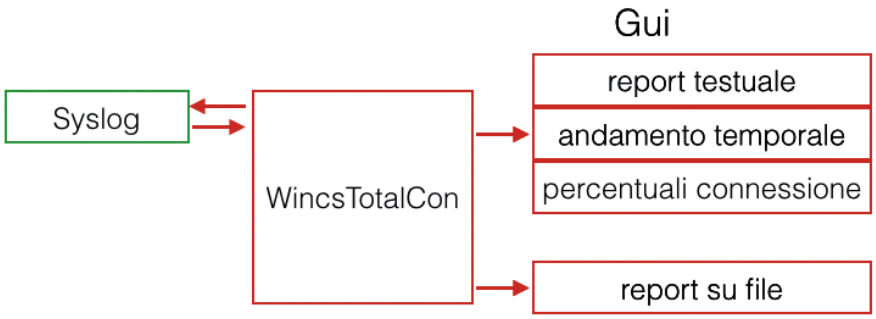


Figura 3.3 - Workflow del software WincsTotalCon

In figura 3.4 è mostrato il workflow del software WinCollaudi.

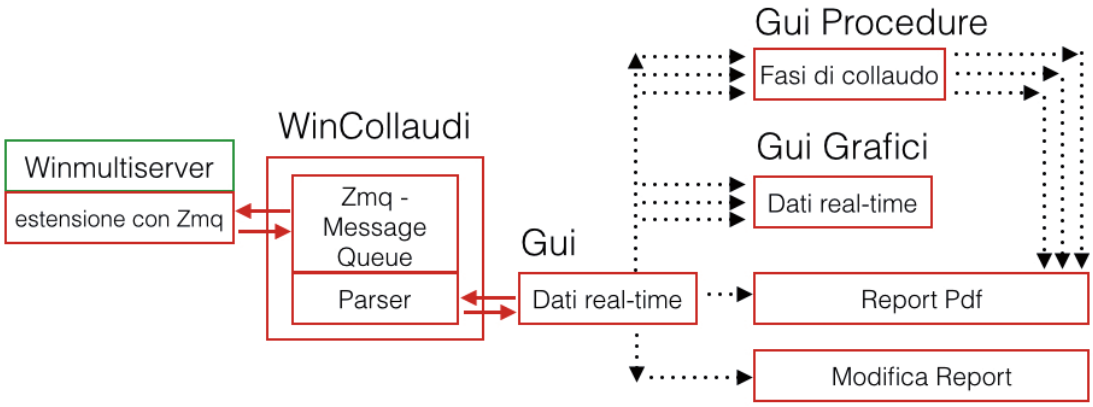


Figura 3.4 - Workflow del software WinCollaudi

Gli input sono i dati provenienti da Winmultiserver, il server bluetooth con il quale il Winpack si autentica e mantiene attiva la connessione. WinCollaudi si occupa di dialogare continuamente con Winmultiserver sfruttando l'estensione creata ad hoc che permette la comunicazione su *socket*. Inoltre, WinCollaudi implementa un *parser* che si occupa di popolare la GUI che espone il front-end principale. Attraverso l'interazione con questo front-end l'operatore può svolgere diverse operazioni: visualizzare grafici che mostrano l'andamento, in tempo reale, dei dati collezionati dal WinPack, creare o modificare i file di report, eseguire le procedure di collaudo sui moduli, sfruttando la GUI dedicata.

Tutte queste operazioni sono state implementate sfruttando i vantaggi della programmazione *multi-threading*. WinCollaudi che rappresenta il core del software, implementa tutte e leoperazioni comuni agli altri *thread*.

L'output di WinCollaudi è costituito dai grafici in *real-time* dei dati provenienti dal Winpack e dai file di report sia per i singoli moduli sia per la sessione di collaudo.

3.2 Test per l'integrità dei dati - 3LdbExtractor

Viene presentato il software, sviluppato e pensato per effettuare efficacemente test d'integrità sui dati memorizzati all'interno del database: 3LdbExtractor.

I dati, provenienti dal Winpack, sono memorizzati all'interno di un database non relazionaleI test vengono effettuati programmando il firmware del winpack affinchè essi mandino segnali fake e aventi forse a rampa, capaci cioè di svariare per tutto il *range* di dati ammissibili per ciascun modulo. 3LdbExtractor, conoscendo quali segnali sono stati inviati dal Winpack, può analizzare, attraverso opportune query, l'integrità dei dati memorizzati. Il software crea dei file di testo dove memorizza i dati estratti e aggiunge in fondo un report sui risultati ottenuti. Lo stesso report viene riportato anche a video, così che l'utente del software ha subito visione del buono o cattivo esito dell'operazione di analisi sul Db. Il DB utilizzato è MongoDb.

MongoDB è un sistema gestionale di basi di dati non relazionale. Il linguaggio utilizzato per la gestione dei dati sfrutta la notazione JSON. È inoltre un gestore di database schemaless, cioè i dati, anche all'interno di una stessa tabella, possono essere strutturati in qualsiasi modo e non devono rispettare alcuna regola. MongoDB raggiunge prestazioni estremamente elevate, in termini di velocità di risposta a singole query, anche con grandi quantità di dati: è questo uno dei principali motivi per cui è stato adottato per la gestione dei dati provenienti dal Winpack. L'elevata frequenza di campionamento e il quantitativo di dati raccolti sono infatti piuttosto elevati; basti pensare, ad esempio, che il modulo Ecg4 è capace di rilevare quattro derivazioni contemporaneamente e ciò avviene calcolando cinquecento campioni al secondo. Ogni due millisecondi quindi ogni derivazione deve essere memorizzata, in ottemperanza all'architettura prevista per la memorizzazione.

Sempre in riferimento all'esempio del modulo Ecg4 è possibile vedere in figura 3.5 come i dati in arrivo al server vengano visualizzati dall'applicazione web-client sfruttando i dati del database.

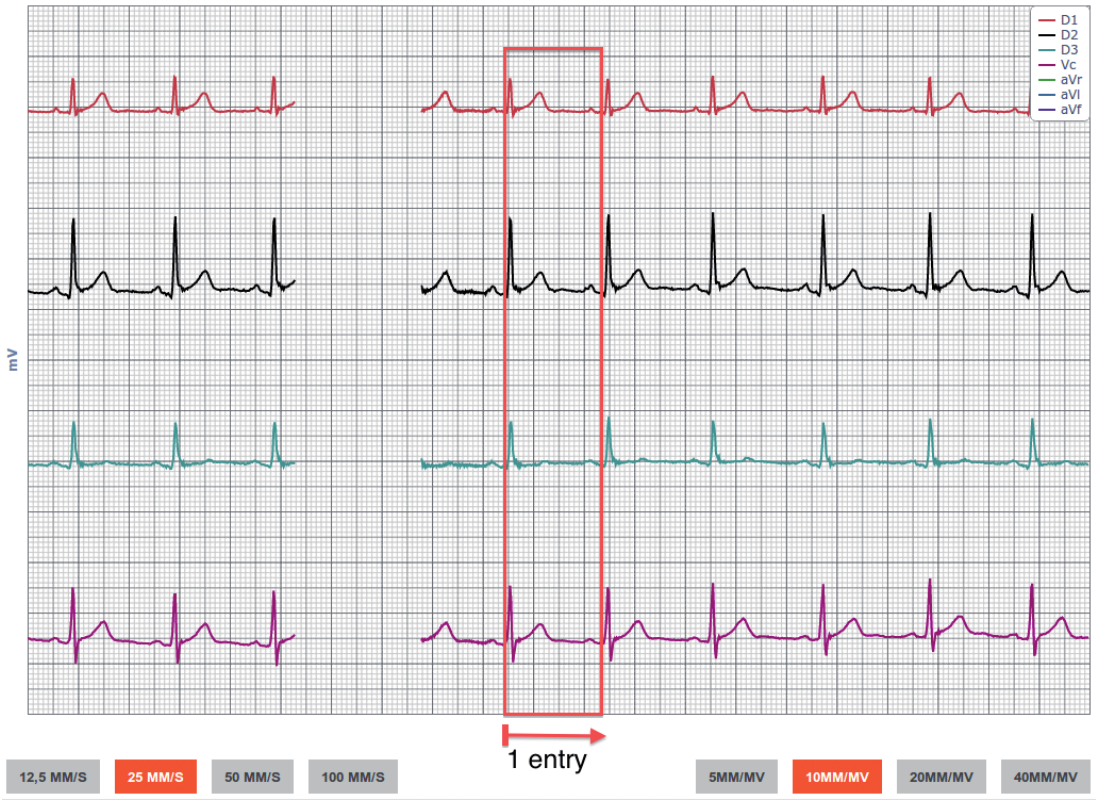
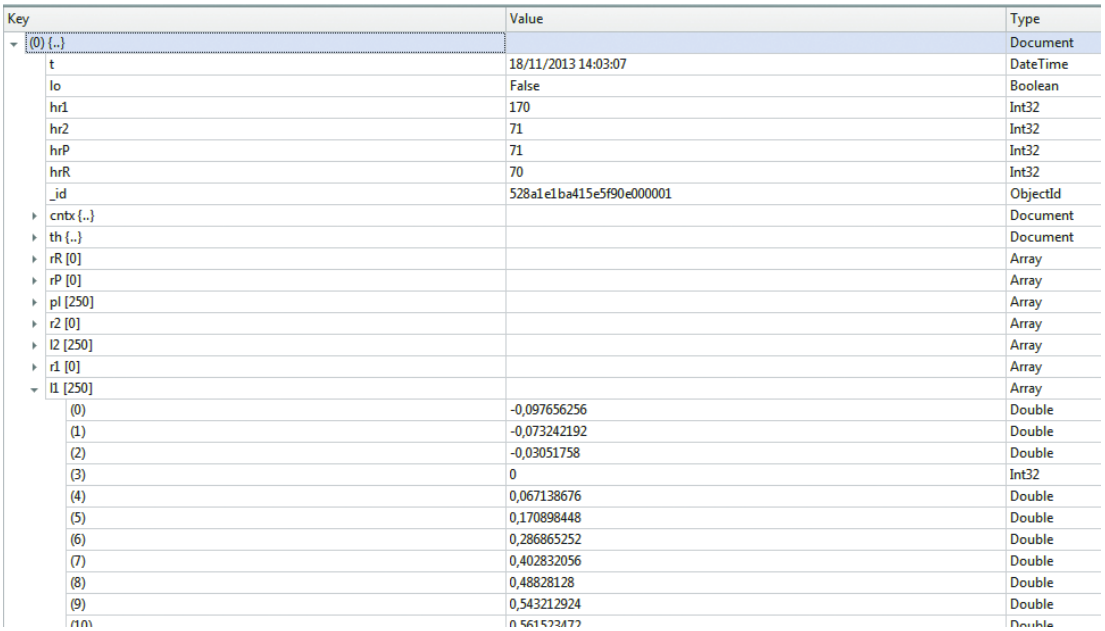


Figura 3.5 - Grafico per le tracce dell'Ecg4

La porzione di segnale visualizzata in figura rappresenta una entry del database.

I dati vengono memorizzati seguendo una precisa struttura Json, come è possibile vedere in figura 3.6.



Key	Value	Type
(0) {..}		Document
t	18/11/2013 14:03:07	DateTime
lo	False	Boolean
hr1	170	Int32
hr2	71	Int32
hrP	71	Int32
hrR	70	Int32
_id	528a1e1ba415e5f90e000001	ObjectId
cntx {..}		Document
th {..}		Document
rR [0]		Array
rP [0]		Array
pI [250]		Array
r2 [0]		Array
I2 [250]		Array
r1 [0]		Array
I1 [250]		Array
(0)	-0,097656256	Double
(1)	-0,073242192	Double
(2)	-0,03051758	Double
(3)	0	Int32
(4)	0,067138676	Double
(5)	0,170898448	Double
(6)	0,286865252	Double
(7)	0,402832056	Double
(8)	0,48828128	Double
(9)	0,543212924	Double
(10)	0,561523472	Double

Figura 3.6 - Memorizzazione dell'Ecg4 in MongoDb

Avendo chiarito quali dati e come vengono i memorizzati nel database è possibile spiegare il funzionamento del software 3LdbExtractor. In riferimento ad una *query* per prelevare i dati dell'Ecg4 memorizzati è possibile procedere come segue (il codice completo dell'applicazione 3LdbExtractor è presente nell'appendice B del presente documento):

1) Accedere al database

```
1.  Mongo m = new Mongo( jTextField_dbaddress.getText());
2.  db = m.getDB( "winmed_db" );
3.  boolean auth = db.authenticate(user,passwd.toCharArray());
```

2) Selezionare i record di interesse

```
1.  BasicDBObject query = new BasicDBObject();
2.
3.  // query che seleziona i record solo in base agli intervalli introdotti nella form
4.  query.put("t", BasicDBObjectBuilder.start("$gte", jSpinner_from.
    getValue()).add("$lte", jSpinner_to.getValue()).get());
5.
6.  // table contiene la stringa del singolo modulo di cui si vogliono estrarre i dati
7.  String table = jComboBox_db_collections.getSelectedItem().toString
    ();
8.
9.  Date datefromfile = (Date) jSpinner_from.getValue();
10. String dateform =(DateFormat.getDateTimeInstance(DateFormat.
SHORT, DateFormat.LONG).format(datefromfile));
```

```
11.
12. Date datetofile = (Date) jSpinner_to.getValue();
13. String dateformto =(DateFormat.getDateTimeInstance(DateFormat.
SHORT, DateFormat.LONG).format(datetofile));
14.
15. String app2 = app.replaceAll("AllItem", "ecg5s");
16. coll = db.getCollection(app2);
17.
18. // pl,l1,l2 saranno tre array, ognuno da 250 valori, in accordo con il protocollo
19. fields = new BasicDBObject
    ("pl",true).append("l1",true).append("l2",true).append("t",true).
    append("_id",false);
20. cur = coll.find(query, fields).sort(new BasicDBObject("t", 1));
```

3) Scorrere i record selezionati contando gli errori

```
1.  while(cur.hasNext())
2.  {
3.    pl = (ArrayList)cur.next().get("pl");
4.    ArrayList l1 = (ArrayList)cur.curr().get("l1");
5.    ArrayList l2 = (ArrayList)cur.curr().get("l2");
6.
7.    date = (Date) (cur.curr().get("t"));
8.    long dateunix = date.getTime() + 7200000;
9.    dateprev = dateunix - 500;
10.   if ((pl.size()!=250)||(l1.size()!=250)||(l2.size()!=250))
11.   {
12.     s_err += 1;
13.     errorextractall = true;
14.   }
15.
16.   i_pl = pl.iterator();
17.   Iterator i_l1 = l1.iterator();
18.   Iterator i_l2 = l2.iterator();
19. ...
```

4) Stampare i risultati a video e su file

```
1.  writer.append("REPORT for ECG5 \r\n"
2.    + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
errors #: "+t_err+"\r\nTimestamp Errors perc: "+date_err+"% \r\n"
3.    + "Packets extracted (L1,L2,PL): "+tot_smpl+"\r\n"
4.    + "PL Errors #: "+pl_v_err+"\r\nPL values Errors perc: "+pl_
      value_err+"% \r\n"
5.    + "L1 Errors #: "+l1_v_err+"\r\nL1 values Errors perc: "+l1_
      value_err+"% \r\n"
6.    + "L2 Errors #: "+l2_v_err+"\r\nL2 values Errors perc: "+l2_
      value_err+"% \r\n”;
```

L’analisi dei risultati consiste essenzialmente in tre aspetti fondamentali:

1) Analisi della distanza temporale tra un dato e il successivo.

2) Analisi del numero totale di sample che, a seconda del modulo e del protocollo di comunicazione, devono essere memorizzati all’interno del database.

3) A seconda del tipo di modulo occorre valutare che il dato che viene letto sia quello che si vuole, cioè sapendo quali dati vengono dati in ingresso e con quale sequenza, è possibile a partire dal primo valore capire se il successivo è o non è un dato corretto.

È possibile vedere un risultato di un'estrazione dei dati usando il software 3LdbExtractor nella figura sottostante.

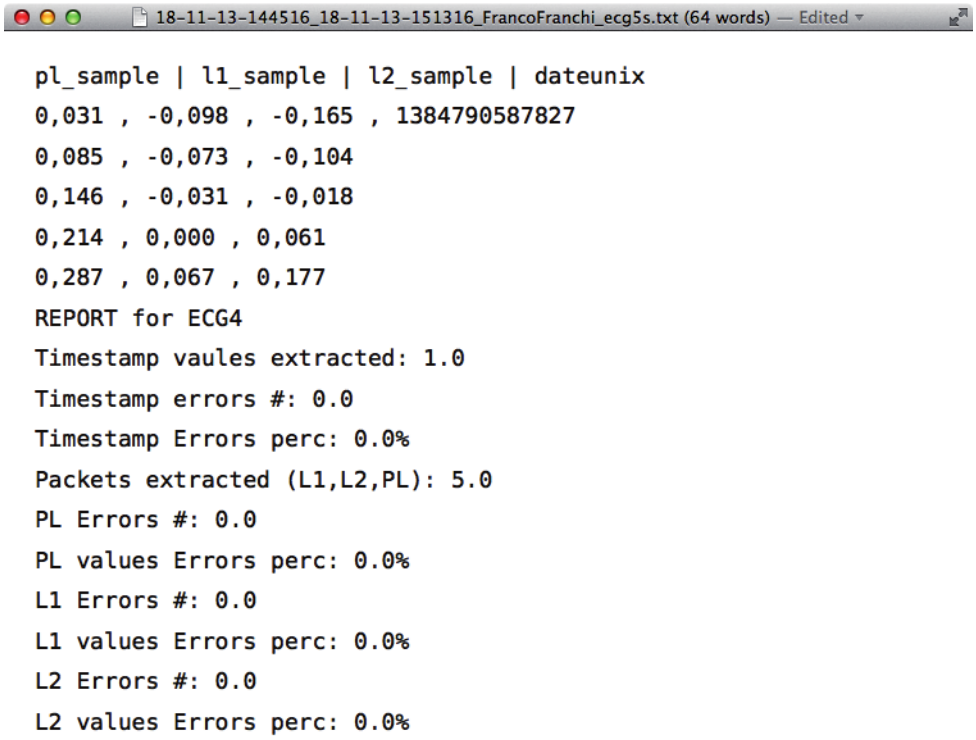


Figura 3.7 - Esempio di report per 3LdbExtractor

3.2.1 Riprogrammazione firmware

Per rendere più efficiente la fase di test sul database si è cercato una soluzione che assicurasse che tutti i valori previsti dal sistema venissero spediti correttamente dal Winpack e soprattutto venissero memorizzati nelle varie entry del database.

Per questo motivo, d'accordo interni con il team di sviluppo dell'azienda, si è pensato a modificare il firmware così da avere moduli fake, in grado cioè di produrre dei segnali pilota.

In particolare si è pensato ad una soluzione che creasse dei segnali a rampa. Così facendo infatti si è assicurato che il Winpack invii segnali capaci di spaziare su tutto il *range* ammesso per ogni modulo. Il modulo base e i moduli accessori hanno installato micro ARM di ST a 32bit. In figura è mostrato il risultato della riprogrammazione del firmware di un modulo Ecg4.

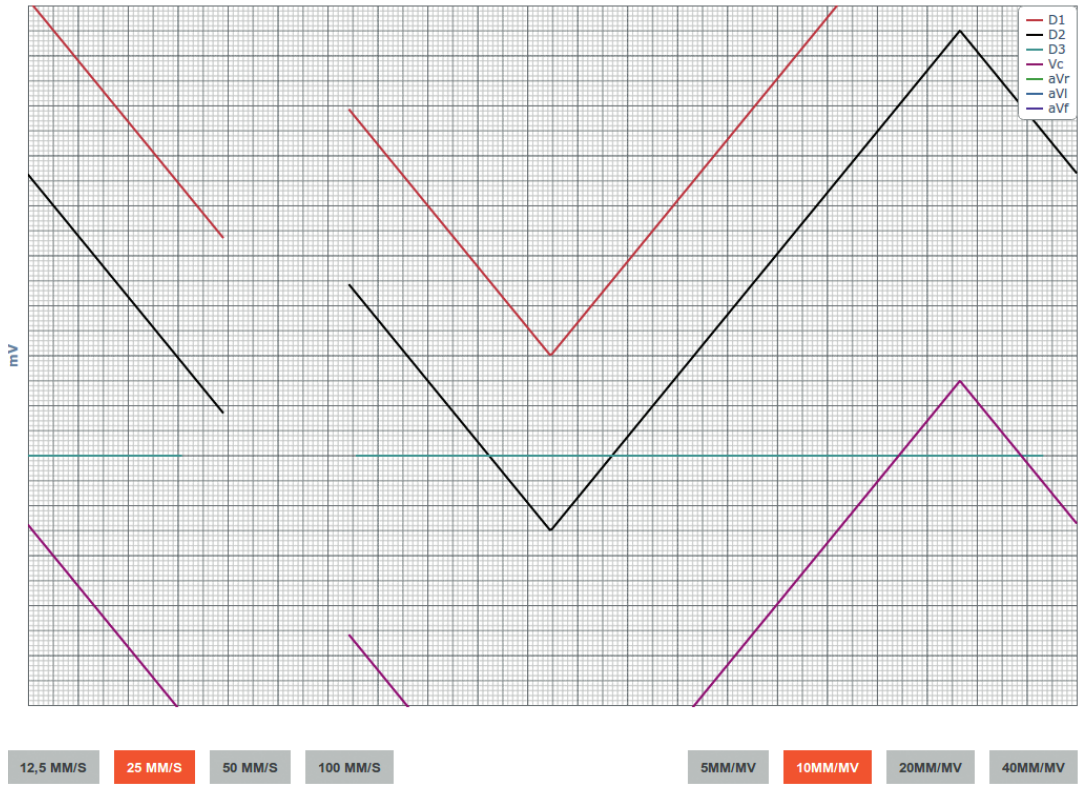


Figura 3.8 - Effetto della riprogrammazione del firmware di Ecg4

3.2.2 Interfaccia utente di 3LdbExtractor

L'interfaccia (figura 3.9) prevede un campo denominato DB address attraverso cui è possibile digitare un indirizzo IP di una macchina server dove è presente il datacollector. Dopo averlo digitato sarà possibile usate il tasto "Connect". Se l'operazione è andata a buon fine il tasto rimarrà selezionato. Inoltre le combo box previste per i pazienti e per i segnali cominceranno a popolarsi dei dati trovati all'interno del database.

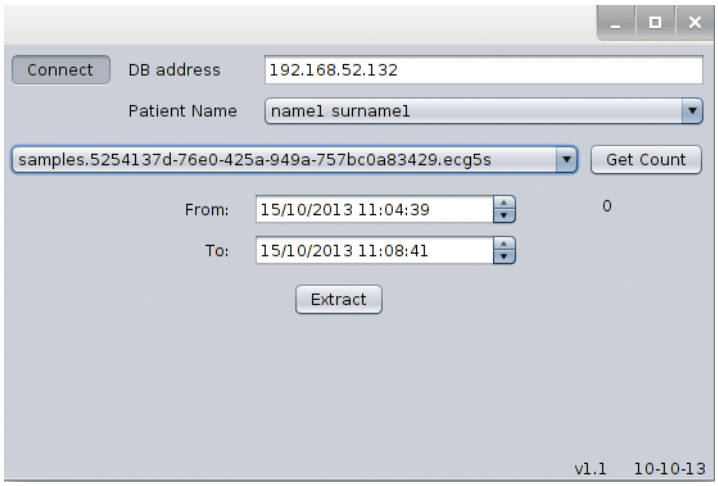


Figura 3.9 - Interfaccia utente di 3LdbExtractor

Attraverso le due combo box di cui sopra è possibile selezionare la riga corrispondente al paziente scelto (vengono indicati nome e cognome) e il modulo telemetrico di cui si vogliono analizzare i dati.

In quest'ultima combo box è presente anche una voce "all items" che permette di analizzare tutti i moduli telemetrici disponibili nel sistema.

Un bottone "Get Count" permette di avere un rapido conteggio del numero di righe del database, che verranno analizzate.

Svolte queste prime operazioni sarà possibile scegliere una finestra temporale di cui si vogliono analizzare i dati. Poi è sufficiente cliccare sul pulsante "Extract" per iniziare l'analisi.

Al termine di questa operazione verranno creati due tipi di feedback: uno visivo, con i risultati stampati direttamente all'interno della Gui, poco sotto il pulsante Extract, e un file di testo, diverso per ogni modulo, dove vengono riportati tutti i dati analizzati e, in fondo ad ogni file, un report dei risultati con annesse percentuali d'errore.

3.3 Test di stabilità del sistema - WincsTotalcon

WincsTotalCon ha lo scopo di fornire un supporto all'operatore che deve effettuare itest sulla stabilità delle connessioni dei Winpack con il server. Precedentemente questo test veniva effettuato sfruttando opportuni script Matlab. Non tutte le macchine in dotazione all'azienda possiedono licenze attive per il software; questo crea una notevole dipendenza al pc da utilizzare.Per il raggiungimento di questo obiettivo è possibile sfruttare syslog. Tutte le informazioni riguardo le connessioni dei winpack vengono memorizzate all'interno del log di sistema dalla componente winmultiserver. WincsTotalcon prevede di avere in ingresso un file contenente il Log di sistema opportunamente filtrato, dove cioè vengono scritte solo le righe del Log di sistema che contengono informazioni sul numero totale di connessioni attive al momento della scrittura su file.

3.3.1 Utilizzo del Syslog

Syslog è un sistema di logging omnicomprensivo [2]. È abbastanza flessibile e permette che i messaggi vengano ordinati rispetto alla loro sorgente e alla loro gravità. Inoltre una delle caratteristiche migliori è che questo sistema centralizza il logging in un unico luogo.

Esso viene avviato in fase di boot e rimane costantemente in esecuzione. Per tale motivo si è certi di poter osservare e analizzare le informazioni opportune in ogni periodo desiderato.
Winmultiserver è capace di scrivere, sfruttando il meccanismo di syslog, le informazioni riguardanti il numero di connessioni attive verso il server.

Ogni entry del syslog inizia con il tempo in cui la entry viene registrata. Winmultiserver aggiunge successivamente una stringa contenente la dicitura "totalcon" seguita dal seriale dei dispositivi connessi. È necessario, per utilizzare correttamente il software WincsTotalcon, che il file di syslog venga filtrato sfruttando proprio la stringa "totalcon", inizio di ogni entry.

È possibile ottenere questi file filtrati ad esempio utilizzando il comando: "cat syslog.1 | grep totalcon > wms". Il file "wms" è l'output creato dall'operazione e contiene, di tutte le entry presenti in syslog.1, soltanto quelle che iniziano con la stringa preceduta dal comando grep. È possibile vederne un esempio nella figura sottostante.

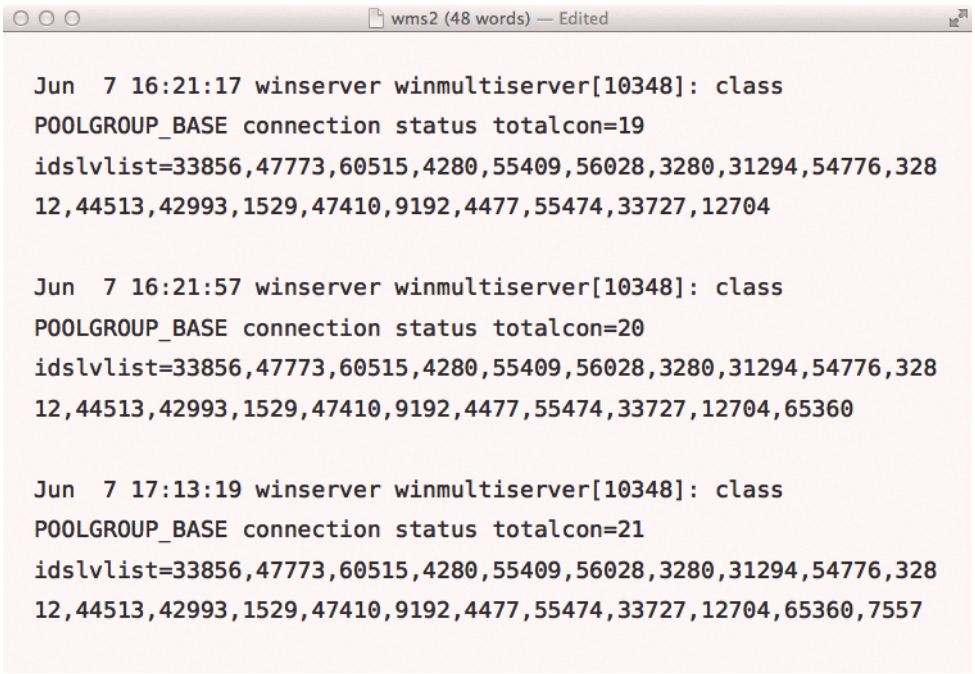
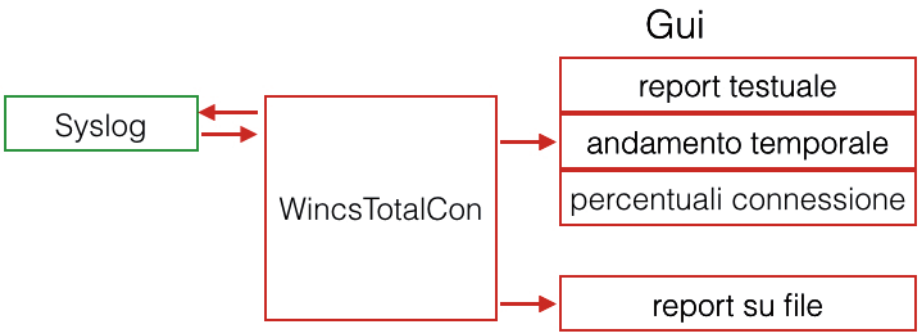


Figura 3.10 - Syslog filtrato con grep "totalcon"

Come nel caso di 3LdbExtractor si può spiegare nel dettaglio il workflow del software. Riconsiderando la figura 3.3 è possibile definire il file "wms" sopracitato come input del



Workflow del software WincsTotalCon

software. Si è già visto come ottenere il file attraverso l'uso di un semplice comando da eseguire da bash. Esso poi viene letto e analizzato, riga per riga, come segue:

1) Analizza tutte le righe attraverso l'uso di un ciclo while.

```
1.  while ((strLine = br.readLine()) != null) {
2.    String d = strLine.substring(0/*, 15*/);
3.    SimpleDateFormat df = new SimpleDateFormat("MMM dd HH:mm:ss",
    Locale.ENGLISH);
4.    dateCurr = df.parse(d);
```

2) Su ogni riga del file viene eseguito uno split per separare le informazioni d'interesse. In questo caso, la stringa totalcon viene seguita dall'elenco dei seriali dei microcontrollori connessi al server. Contando questi seriali è possibile capire il numero di connessioni attive per ogni entry.

```
5.  String[] tok = strLine.split("totalcon=");
6.  String[] tokens = tok[1].split(" ");
7.  totalcon_curr = Integer.parseInt(tokens[0]);
8.  if (counter > 0) {
9.    time = dateCurr.getTime() - datePrev.getTime();
10.   Lconnections.add(counter,totalcon_prev);
11.   Ltime_diff.add(counter,time);
12.   acquisition_time += time;
```

3) Memorizza in un *array* il tempo di connessione dell'attuale numero di dispositivi.

```
13.   total[totalcon_prev] += time;
14.   if (totalcon_curr > totalcon_display) {
15.     totalcon_display = totalcon_curr;
16.   }
17.
18.   //aggiungo l'istante iniziale
19.   dataSeries.add(datePrev.getTime(), totalcon_prev);
20.
21.   //aggiungo l'istante precedente al cambio di connessioni
22.   Date dateTemp = new Date(dateCurr.getTime()-1);
23.   dataSeries.add(dateTemp.getTime(), totalcon_prev);
24.   }
25.   totalcon_prev = totalcon_curr;
26.   datePrev = dateCurr;
27.   counter++;
28. }
```

3.3.2 Interfaccia utente di WincsTotalCon

L'introduzione del software in questione permette, sfruttando tre finestre diverse, la visualizzazione dei dati in maniera più efficiente e intuitiva.

Per utilizzare l'interfaccia, visibile in figura 3.11, è necessario anzitutto cliccare sul pulsante "open log file". Il file di log richiesto dall'applicazione deve essere creato ad

esempio come segue: cat Winmultiserver.log | grep totalcon > wms. È quindi possibile selezionare il file “wms”. Il completamente di questa azione fa automaticamente iniziare la procedura di analisi, popolando le tre finestre.

Figura 3.11 - Interfaccia utente di WincsTotalCon

Nella prima, “Riepilogo”, visibile è possibile leggere i risultati in formato testuale.

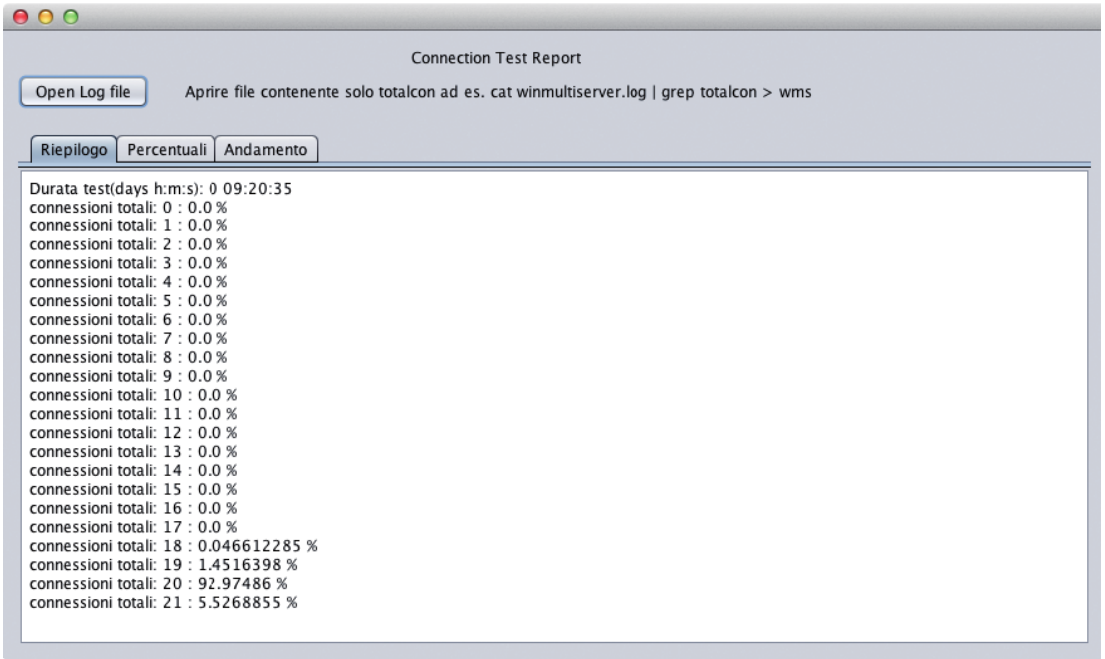


Figura 3.12 - Interfaccia “Riepilogo” di WincsTotalCon

Il numero totale delle connessioni trovate all'interno del file rappresenta una riga d'output. Al numero di connessioni segue la percentuale di connessioni in funzione del tempo totale del test. Questo risultato è stato ottenuto analizzando tutte le posizioni dell'*array* in cui è stato memorizzato il tempo per ogni numero di connessioni ed effettuando un'operazione di percentuale in riferimento al tempo totale del test.

```
1.  for (int j = 0; j <= totalcon_display; j++) {
2.    jTextArea_result.append("connessioni totali: " + j + " : " +
      (float)
3.    (total[ j] * 100.0f / acquisition_time) + " %\r\n");
4.    if (total[ j] != 0) {
5.      dataset.setValue((float) (total[ j] * 100.0f / acquisition_
        time), "", ""+j);
6.    }
7.  }
```

La seconda schermata, "Percentuali", fornisce, attraverso la creazione di un grafico a barre, per ogni numero di connessioni la percentuale in cui esse si sono verificate rispetto al tempo totale del test. È possibile vedere uno screenshot di questa schermata nella figura sottostante.

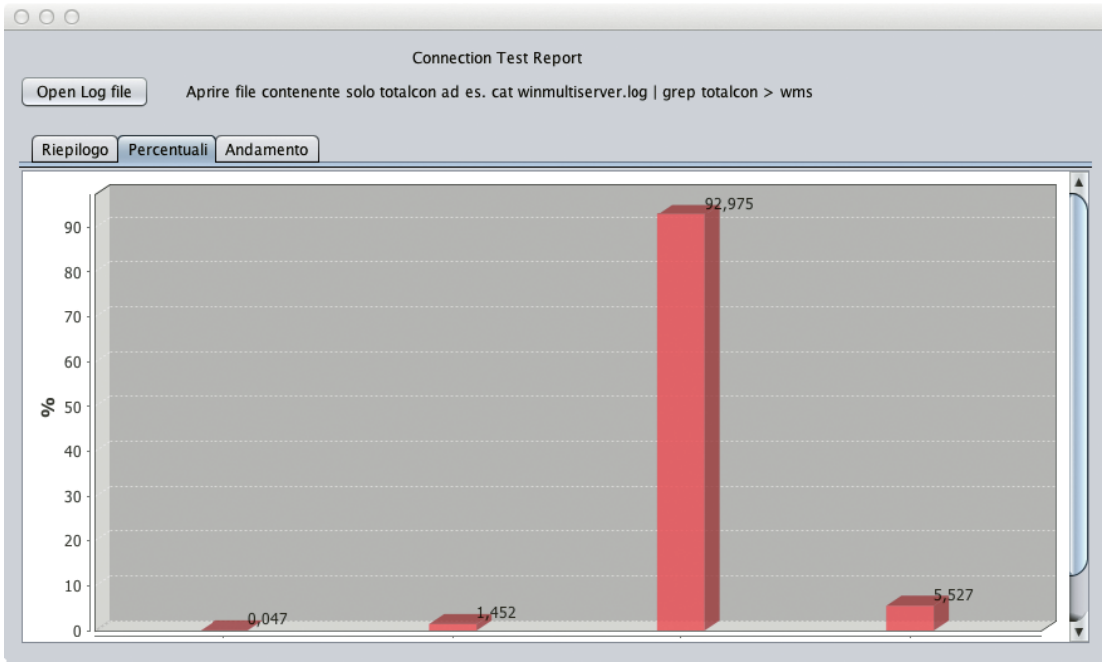


Figura 3.13 - Interfaccia "Percentuali" di WincsTotalCon

Come è possibile vedere sono omesse, per facilitare la leggibilità del grafico, il totale delle connessioni che si sono verificate zero volte durante il test.

Questo risultato è stato ottenuto sfruttando il framework open-source jFreeChart [3] nel seguente modo:

1) Viene creato un chart passando al costruttore la data series costruita con l'*array* delle connessioni totali

```
1.  JFreeChart chart = ChartFactory.createBarChart3D(
2.  "", // chart title
3.  "Connessioni totali", // domain axis label
4.  "%", // range axis label
5.  dataset, // data
6.  PlotOrientation.VERTICAL, // orientation
7.  false, // include legend
8.  true, // tooltips?
9.  false // URLs?
10. );
```

2) Viene aggiunto il chart ad un pannello che lo contiene

```
1.  ChartPanel chartPanel = new ChartPanel(chart, false);
2.  final CategoryPlot plot2 = chart.getCategoryPlot();
3.  plot2.setNoDataMessage("No data available");
4.  final CategoryItemRenderer renderer2 = plot2.getRenderer();
```

3) Viene scelto il layout per le barre che compongono il grafico

```
5.  final BarRenderer r = (BarRenderer) renderer2;
6.  r.setMaximumBarWidth(0.05);
7.  r.setBaseItemLabelGenerator(new StandardCategoryItemLabel
    Generator());
8.  r.setBaseItemLabelsVisible(true);
9.  r.setSeriesPositiveItemLabelPosition(0,
10. new ItemLabelPosition(ItemLabelAnchor.OUTSIDE1,
11. TextAnchor.BOTTOM_LEFT, TextAnchor.TOP_LEFT,
12. 2*Math.PI)
13. );
```

La terza schermata, "Andamento", visualizza l'andamento delle connessioni. Sulle ordinate è presente il numero di connessioni e sull'asse delle ascisse il tempo in cui quel dato numero di connessioni si sono verificate.

Il grafico in figura è ottenuto in maniera del tutto analoga all'interfaccia "Percentuali". Si rimanda quindi all'appendice C per il codice completo del software.

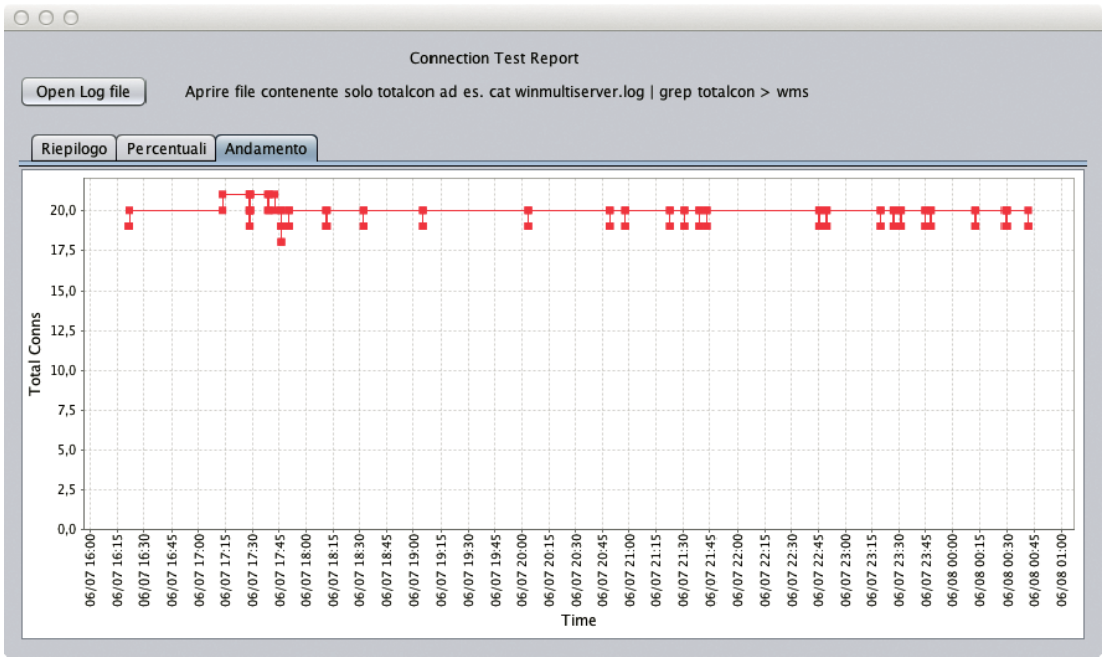


Figura 3.14 - Interfaccia "Andamento" di WincsTotalCon

3.4 Collaudo dei moduli - WinCollaudi

WinCollaudi ha lo scopo di agevolare l'operatore addetto al collaudo dei moduli, soprattutto attraverso due aspetti: automatizzare la creazione dei documenti necessari per la certificazione del buon o cattivo esito del collaudo e la lettura dei valori provenienti dal Winpack così da ridurre al minimo gli errori.

La componente di Communication Server (in seguito Winmultiserver) è sviluppata in C++. È un'applicazione multiserver, è in grado cioè di ricevere e gestire n connessioni, una per ognuno degli n Winpack che tenta la connessione al data collector. Una volta avvenuta la connessione del modulo bluetooth è necessario inoltre rispettare un protocollo di comunicazione interno tra il dispositivo e l'applicazione Winmultiserver. Si è reso necessario modificare e ampliare questa componente prima di procedere alla creazione del set di applicativi di 'alto livello' (in Java) e ciò si è tradotto nella creazione

di alcune classi capaci di ricevere i dati e di inviarli, attraverso dei *socket* ZeroMQ, all'applicazione Java in *bind* sugli stessi *socket*.

L'applicazione permette inoltre la visualizzazione in tempo reale dei dati provenienti dai Winpack e, ove possibile, viene permessa anche la visualizzazione delle singole traccead esempio per l'elettrocardiogramma.

Prima di approfondire nel dettaglio il software WinCollaudi, in particolare nella spiegazione dell'interfaccia presentata all'operatore, si vogliono approfondire i tre aspetti principali che caratterizzano il backend del software. In particolare:

- La scelta di implementare un *Message Queue*, in particolare scegliendo la liberia 0mq, per il trasferimento dei messaggi sui *socket*.
- La necessità di estendere Winmultiserver per dotarlo della libreria 0mq.
- La scelta di implementare un'applicazione modulare, scalabile e *multi-threading*.

3.4.1 *Message Queue*

I Berkeley Sockets (BSD) sono lo standard *de facto* per tutte le comunicazioni in rete: sviluppate nei primi anni '80, sono uno dei componenti critici più estesamente supportati in qualsiasi moderno sistema operativo. Per avere applicativi veloci ed efficienti essi necessitano di esplicita preparazione e regolazione, di una scelta del protocollo (TCP o UDP), di una gestione degli errori, ecc.

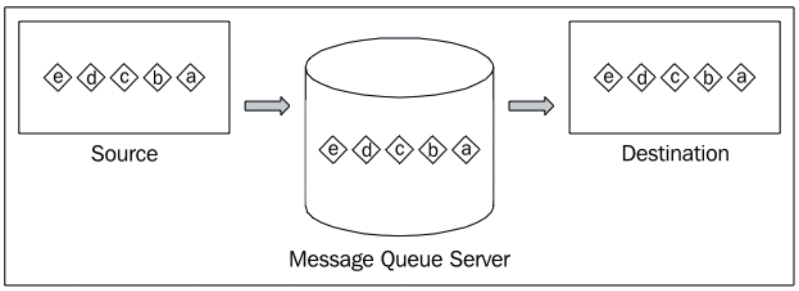


Figura 3.15 - Schema del Message Queue

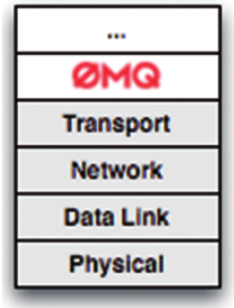
Per superare queste problematiche è possibile usare un componente software, il message queue, che provvede ad implementare un protocollo di comunicazione asincrono. Riesce cioè ad evitare cheil mittente e il ricevente interagiscano con la coda dei messaggi

allo stesso tempo. I messaggi infatti sono memorizzati mentre vengono ricevuti. Altro aspetto importante di questa tecnica è che non è necessario esplicitare la dimensione dei dati che devono essere trasmessi in un singolo messaggio. Ciò è molto importante perché nel caso in esame i pacchetti contengono messaggi di dimensione molto diversa, poiché diversi sono i moduli che generano questi dati che poi verranno impacchettati da Winmultiserver. Basti pensare infatti che ogni modulo ha una propria frequenza di campionamento che varia da 5Hz per il modulo temperatura ai 500Hz per il modulo Ecg4.

In letteratura esistono molti middleware open-source per lo scambio di messaggi. Per citarne alcuni possiamo elencare: ActiveMq, RabbitMq, 0mq. Analizzando la diversità di soluzioni possibili la scelta è stata di adottare 0mq. Come spiegato [4] tra le soluzioni possibili, esposte in [5], ZeroMQ [6] è considerata la migliore in termini di velocità di trasmissione e di ricezione dei pacchetti.

Inoltre ZeroMQ è già stata scelta dal CERN e ESRF per TANGO [7].

Un beneficio di questa scelta sono: avere una grande e attiva comunità di sviluppo e avere un supporto multi-language oltre ad ottenere buone performance [8].



La **libreria di rete ZeroMQ** (chiamata anche ØMQ o ZeroMQ), che implementa un livello di astrazione, evita la tediosa gestione di tutti i particolari di cui sopra.

Essa fornisce *socket* per trasmettere l'intero messaggio attraverso diversi transport, come inproc, IPC, TCP e multicast. Anzitutto, invece di essere orientata allo stream (TCP) o al *datagram* (UDP), ***ZeroMQ è message-oriented***. Questo significa che se un *socket* client invia un messaggio da 150 KB, il *socket* server lo riceverà per intero, senza bisogno di implementare espliciti *buffering* o *framing*. Un esempio di come può avvenire uno scambio di messaggi con ZeroMQ è il seguente:

```
1.  # create ZeroMQ request / reply socket pair
2.  ctx = ZeroMQ::Context.new
3.  req = ctx.socket ZeroMQ::REQ
4.  rep = ctx.socket ZeroMQ::REP
5.
6.  # connect sockets: notice that reply can connect first even with no server!
7.  rep.connect('tcp://127.0.0.1:5555')
8.  req.bind('tcp://127.0.0.1:5555')
9.  req.send ZeroMQ::Message.new('hello' * (1024*1024))
10.
11. msg = ZeroMQ::Message.new
```

```
12. rep.recv(msg)
13. msg.copy_out_string.size # => 5242880
```

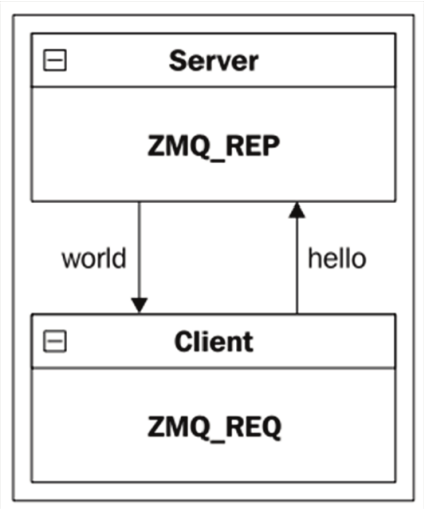


Figura 3.16 - Schema del Request - Reply

Un'altra caratteristica è quella dei **Transport Agnostic Socket**: esiste cioè una singola API unificata per inviare e ricevere messaggi. Questo significa che si può iniziare a colloquiare in IPC per comunicazioni locali veloci, e poi passare al TCP con un minimo sforzo, perché non bisogna preoccuparsi della logica di creazione, distruzione e riconnessione.

Altro beneficio di ZeroMQ è il **routing and network topology aware**: visto che non si deve gestire esplicitamente la connessione, nulla vieta che un singolo *socket* sia associato a due diverse porte, in ascolto delle richieste in arrivo. Similmente, possono essere inviati dati a due *socket* distinti con una singola chiamata API.

3.4.2 Estensione Winmultiserver

Winmultiserver è la componente installata nel client bluetooth, capace di dialogare con il modulo base e di inoltrare gli stessi dati ad altri client che a valle sfruttano questi dati.

Ad esempio, un client a valle chiedere questi dati per memorizzarli nel database, un altro li richiede per visualizzarli nell'applicazione web-client ad uso del personale sanitario che usa i dispositivi telemetrici.

Estendere Winmultiserver significa creare una classe che erediti i metodi già esistenti (la gestione dei *thread* e la gestione delle richieste interne) fornendo un'interfaccia ZeroMQ, come precedentemente descritto. Inoltre la classe deve inoltre i dati in arrivo, secondo il protocollo di comunicazione interno, ponendoli sul *socket* creato all'occorrenza.

Nel dettaglio la sezione della classe sviluppata per creare questa estensione, per quanto riguarda lo scambio di informazioni, è la seguente:

```
1.  // sequenza di request e replay
2.    try {
3.      si32_t zres;
4.
5.  // tempo iniziale
6.      TIMEUS tini;
7.
8.  // compongo il messaggio parziale con
9.  // lo slave id
10.   ui8_t ZeroMQid[ 2] = {   (ui8_t)((idslv>>0) & 0x00FF),
11.     (ui8_t)((idslv>>8) & 0x00FF) };
12.   ZeroMQ::message_t ZeroMQidmsg(ZeroMQid,
13.     sizeof(ZeroMQid),
14.     (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
15.     0 );
16. // spedisco il pacchetto
17.   zres = collaudi_send(idslv, ZeroMQidmsg, true);
18.   if ( zres == 0 )
19.     throw 0;
20.   else if ( zres < 0 )
21.     throw 1;
22. // stampo le informazioni di ogni token
23.   for ( ui32_t i=0; i<tokvec.size(); i++ )
24.   {
25.     SERVER::TOKEN tok = tokvec[ i] ;
26. // compongo il pacchetto
27.
28.     ui32_t ZeroMQid = {     tok.devid } ;
29.     ZeroMQ::message_t ZeroMQdeviceid(&ZeroMQid,
30.       sizeof(&ZeroMQid),
31.       (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
32.       0 );
33. // spedisco il pacchetto
34.       zres = collaudi_send(idslv, ZeroMQdeviceid, true);
35.       if ( zres == 0 )
36.         throw 0;
37.       else if ( zres < 0 )
38.         throw 1;
39.       ui64_t ZeroMQrtc = {    tok.rtcms } ;
40.       ZeroMQ::message_t ZeroMQrtcms(&ZeroMQrtc,
```

```
41.           sizeof(ZeroMQrtc),
42.        (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
43.        0 );
44.
45.        zres = collaudi_send(idslv, ZeroMQrtcms, true);
46.        if ( zres == 0 )
47.          throw 0;
48.        else if ( zres < 0 )
49.          throw 1;
50.
51.        ZeroMQ::message_t msg( (void*)tok.payload,
52.           tok.length,
53.        (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
54.        0 );
55. // spedisco il pacchetto
56.        bool more = (i+1) < tokvec.size();
57.        zres = collaudi_send(idslv, msg, more);
58.        if ( zres == 0 )
59.          throw 0;
60.        else if ( zres < 0 )
61.          throw 1;
62.
63.   }
64. // ricevo la risposta
65.        ZeroMQ::message_t rep;
66.        zres = collaudi_recv(idslv, rep);
67.        if ( zres == 0 )
68.          throw 0;
69.        else if ( zres < 0 )
70.          throw 1;
71. // controllo la risposta
72.        ui8_t *repdata = (ui8_t*)rep.data();
73.        ui32_t replen = (ui32_t)rep.size();
74.        if ( replen != 1 )
75.          throw 2;
76. // è stato accettato il token?
77.        tokens_accepted = tokens_accepted && (repdata[ 0] ==0x01);
78.
79. // tempo finale
80.          TIMEUS tfin;
81.          TIMEUS tdiff(tfin-tini);
82. // stampa del tempo impiegato
83.      { MUTEXLOCK mlock(collaudi_time_mutex);
84.      cout << setfill(' ') << setw(7) << (tdiff.us_get()/1000) <<
      ".";
85.      cout << setfill('0') << setw(3) << (tdiff.us_get()%1000) <<
    endl; }
86.   }
```

3.4.3 *Multi-thread* - Modularità - Scalabilità

La programmazione orientata agli oggetti (OOP, Object Oriented Programming) [9] è un paradigma di programmazione che permette di definire oggetti software in grado di in-

teragire gli uni con gli altri attraverso lo scambio di messaggi. È particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare (contenimento, uso, specializzazione). Un ambito che più di altri riesce a sfruttare i vantaggi della programmazione ad oggetti è quello delle interfacce grafiche.

Tra gli altri vantaggi della programmazione orientata agli oggetti ricordiamo che:

- fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre;
- permette una più facile gestione e manutenzione di progetti di grandi dimensioni;
- l'organizzazione del codice sotto forma di classi favorisce la modularità e il **riuso di codice**.

La programmazione modulare è un paradigma di programmazione che consiste nella realizzazione di programmi suddivisi in moduli, ognuno dei quali svolge precise funzioni conl'obiettivo di semplificare lo sviluppo, il test e la manutenzione di programmi di grosse dimensioni, che vedono coinvolti più sviluppatori tramite il concetto della modularità.

I punti cardine della programmazione modulare sono:

- suddivisione del programma in singoli moduli;
- indipendenza dei moduli tra loro;
- interazione minima di ciascun modulo con il mondo esterno;
- facile testabilità dei moduli come entità isolate;
- dichiarazione esplicita e semplificazione delle interfacce mediante le quali i moduli comunicano tra loro.

I principali vantaggi della programmazione modulare sono:

- riutilizzabilità dei moduli;
- sviluppo separato (minima interazione tra sviluppatori).

Possiamo pensare un *thread* [10] come un flusso di esecuzione. Il classico processo è un flusso di esecuzione di un dato codice, ovvero un insieme di istruzioni macchina. In un software basato sul *multi-threading* un processo è costituito da più *thread*, ovvero più flussi di esecuzione, che possono virtualmente procedere indipendentemente e parallelamente. Con la programmazione standard non è possibile ricevere nuove richieste

fino a quando l'ultima richiesta ricevuta non è stata servita. Come è immaginabile molte delle richieste entranti andrebbero perse. Se però utilizziamo due *thread*, possiamo fare in modo che uno si dedichi a ricevere e accodare le richieste, e l'altro a servirle. Ovviamente ciò può realmente avvenire solo su macchine dotate di più processori, ma viene be simulato da ormai qualsiasi sistema operativo.

In Java un *thread* è rappresentato dalla classe Thread. Per creare un *thread* non dovremo fare altro che implementare una classe che estende la classe Thread. Ogni classe che estende la classe Thread deve contenere un metodo pubblico e void run().

```
1.  public class MyThread extends Thread {
2.    public void run(){
3.      System.out.println('thread');
4.    }
5.
6.    public MyThread(){
7.  // eventuali azioni per il costruttore
8.    }
9.  }
```

In WinCollaudi il *thread* principale continua a espletare il compito di *parser* per i dati provenienti dal Winmultiserver e mentre avviene la lettura di questi dati, altri *thread* si occupano della creazione di grafici o di mostrare e gestire le Gui per le varie fasi di colluado, diverse per ogni modulo.

Il vantaggio principale dei Thread è nelle prestazioni: operazioni come creazione, terminazione e cambio tra due *thread* di un processo richiedono meno tempo rispetto alla creazione, terminazione e cambio di processi. I *thread* migliorano anche l'efficienza della comunicazione fra i programmi in esecuzione. Nella maggior parte dei sistemi operativi, la comunicazione fra processi indipendenti richiede l'intervento del ***kernel*** per fornire un meccanismo di protezione e di comunicazione.

Nella figura sottostante è possibile vedere come gli aspetti di *multi-threading* e di modularità sono stati implementati nello sviluppo di WinCollaudi.

Le linee tratteggiate rappresentano tutte quelle connessioni esistenti ma non obbligatorie. È infatti possibile eseguire uno o più collaudi mentre si stanno visualizzando dei grafici così come è possibile concentrarsi solo sui dati visualizzati dal main frame senza svolgere ulteriori operazioni..

L'aspetto di scalabilità di carico è giustificato dall'implementazione *multi-threading*: più risorse equivalgono quindi a maggiori prestazioni.

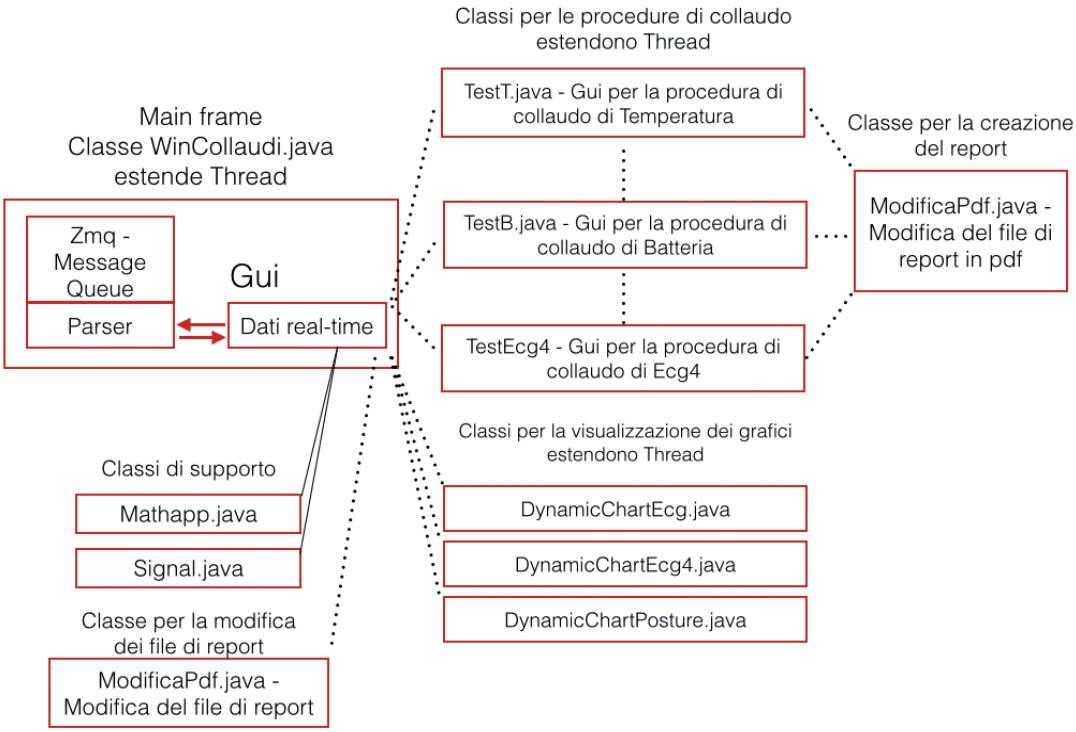


Figura 3.17 - Struttura di WinCollaudi

3.4.4 Interfaccia utente di WinCollaudi

L'interfaccia del software di collaudo è stata sviluppata al fine di ridurre al minimo i possibili errori durante varie fasi eseguite dall'operatore, grazie all'implementazione di semplici istruzioni date in sequenza e alla registrazione in automatico dei risultati. Vengono analizzate nel dettaglio le principali funzionalità a partire dal main frame riportato nella figura sottostante.

Le operazioni preliminari da effettuare sull'interfaccia riguardano la sezione "Collaudatore - Lotto - Seriale Wincard". È necessario inserire queste informazioni e successivamente cliccare sul pulsante "ok" per fa sì che l'interfaccia cominci a mostrare i dati eventualmente in arrivo dal Winmultiserver. Prima di iniziare la procedura di test su un particolare modulo è necessario inoltre inserire il nome di una sessione o aprirne una già esistente. La sessione contiene un riepilogo di tutti i moduli collaudati dal momento in cui è stata creata. Il tasto termina produce un report riassuntivo con l'esito di collaudo dei vari moduli, rimandando al particolare file per l'esame dettagliato di un collaudo su uno specifico modulo.

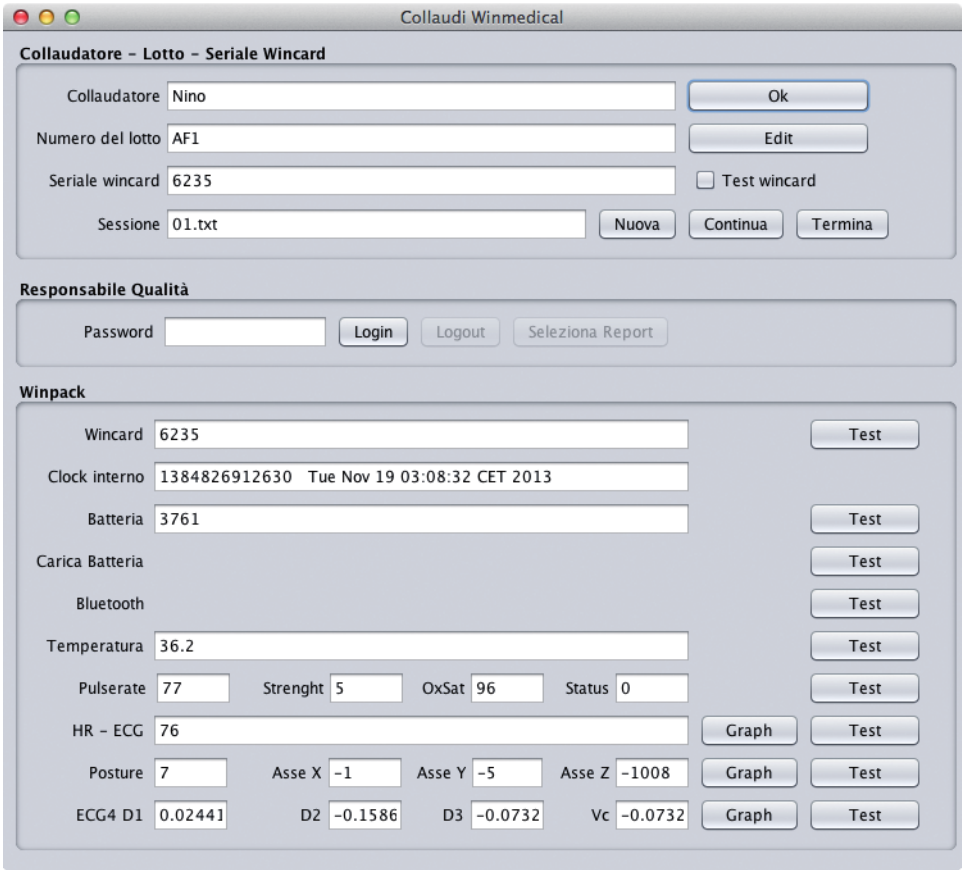


Figura 3.18 - Interfaccia principale di WinCollaudi

WinCollaudi è quindi pronto a dialogare con Winmultiserver e a ricevere i dati che verranno gestiti dal *parser*. È quest'ultimo modulo a popolare l'interfaccia principale mentre i dati vengono letti dal *socket*. Ad esempio la connessione ZeroMQ con Winmultiserver viene creata come segue (il codice completo è disponibile nell'appendice D di questo documento):

1) Creazione del *socket* e del contesto e operazione di *bind*.

```
1.  // parametri per ZMQ
2.  public static int REQUEST_TIMEOUT = 2500;  // msecs, (> 1000!)
3.  public static int REQUEST_RETRIES = 3;  // before we abandon
4.  public static String SERVER_ENDPOINT = "tcp://192.168.52.33:5555";
5.  int ack_i = 1;
6.  final byte[ ] ack = new byte[ 1] ;
7.  ack[ 0] = (byte) (ack_i >>> (0 * 8));
8.  final ZMQ.Context context = ZMQ.context(1);
9.  ZMQ.Socket client = context.socket(ZMQ.REP);
10. client.bind(SERVER_ENDPOINT);
```

2) Il *parser* inizia a distinguere i dati in arrivo dai vari moduli.

```
1.  received = client.recv(0);
2.
3.  // calcolo il dev_id
4.  if (received.length == 4){
5.    dev_id = ((received[3] << 24)&0x7f) + (received[2] << 16)
6.    (received[1] << 8) + received[0];
7.  }
```

3) I dati vengono e ad elaborati in accordo con il protocollo interno di comunicazione.

```
1.  // payload del modulo ECG
2.  // arrivano 1000 byte
3.  if (dev_id == 2)
4.  {
5.  int s1 = 0;
6.  int s2 = 0;
7.
8.  int l, h, m, ml, mh;
9.  heart = 0;
10. int index = 0;
11. for (int i = 0; i < 1000; i += 4) {
12.
13. l = ((int) received[i]) & 0xff;
14. h = ((int) received[i + 2]) & 0xff;
15. m = ((int) received[i + 1]) & 0xff;
16. ml = ((m >>> 4) << 8) & 0xf00;
17. mh = (m << 8) & 0xf00;
18.
19. s1 = l | mh;
20. s2 = h | ml;
21. ...
22. }
```

L'operatore può in ogni momento utilizzare i bottoni Graph per visualizzare, ove abbia senso farlo, i dati in arrivo da un determinato modulo. Cliccando ad esempio sul pulsante Graph del modulo Ecg si ottiene il risultato:

Figura 3.19 - Grafico dell'heart rate

Questo risultato è ottenuto attraverso l'utilizzo, come per il software WincsTotalCon di jFreeChart. Il grafico viene aggiornato in realtime e viene gestito da un'apposita classe in accordo con la figura 3.17. Nel caso d'esempio la classe in questione è DynamicaChartEcg.java che svolge le seguenti operazioni:

1) Crea un chart e una variabile timer che ogni secondo aggiorna il grafico

```
1.  private Timer timer = new Timer(1000, this);
2.
3.  public DynamicChartEcg(final String title) {
4.    super(title);
5.    this.seriesecg = new TimeSeries("ECG", Millisecond.class);
6.
7.  //ECG
8.       final TimeSeriesCollection datasetecg = new TimeSeries
        Collection(this.seriesecg);
9.    chartecg = createChart(datasetecg, "ECG");
10.
11. //Sets background color of chart
12.   chartecg.setBackgroundPaint(Color.LIGHT_GRAY);
13. //Created JPanel to show graph on screen
14.   contentecg = new JPanel(new BorderLayout());
15. //Created Chartpanel for chart area
16.   final ChartPanel chartPanelecg = new ChartPanel(chartecg);
17. //Added chartpanel to main panel
```

```
18.   contentecg.add(chartPanelecg);
19. //Sets the size of whole window (JPanel)
20.   chartPanelecg.setPreferredSize(new java.awt.Dimension(400, 400));
21. //Puts the whole content on a Frame
22.   setContentPane(contentecg);
23.
24.   timer.setInitialDelay(1000);
25. }
```

2) Ogni secondo la variabile timer chiama la funzione actionperformed aggiornando il grafico con le informazioni prelevate da una lista gestita dalla classe Signal.java.

```
1.  public void actionPerformed(final ActionEvent e) {
2.    Signal ecgsample = null;
3.
4.    if (!WinCollaudi.qecg.isEmpty())
5.    {
6.      ecgsample = WinCollaudi.qecg.poll();
7.
8.    if (!ecgsample.equals(null))
9.      {
10.     Long ecgtime = Long.decode(ecgsample.getTimestamp().toStri
    ng()).longValue();
11.
12.   for (int i=0;i<500;i++)
13. {
14.     Date date = new Date(ecgtime);
15.     Millisecond now = new Millisecond(date);
16.     this.seriesecg.addOrUpdate(now, ecgsample.getSampleindex(i));
17.     ecgtime+=2;
18.       }
19.     }
20.   }
21. }
```

3) Il metodo run infine visualizza il *frame* e attiva il timer

```
1.  public void run() {
2.
3.    timer.start();
4.
5.    Toolkit tk = Toolkit.getDefaultToolkit();
6.    Dimension screenSize = tk.getScreenSize();
7.    final int WIDTH = screenSize.width;
8.    final int HEIGHT = screenSize.height;
9.
10.   final JFrame ecg = new JFrame();
11.   ecg.add(contentecg);
12.   ecg.setSize(400,400);
13.   ecg.pack();
14.   ecg.setVisible(true);
15.   ecg.setSize(WIDTH/2, HEIGHT/2);
16.   ecg.setLocation(0,0);
17.
18. }
```

In maniera del tutto indipendente alla visualizzazione dei report l'operatore può svolgere operazioni di collaudo sui moduli di interesse. Continuando l'esempio con il modulo Ecg, cliccando sul tasto Test, viene visualizzata la seguente interfaccia.

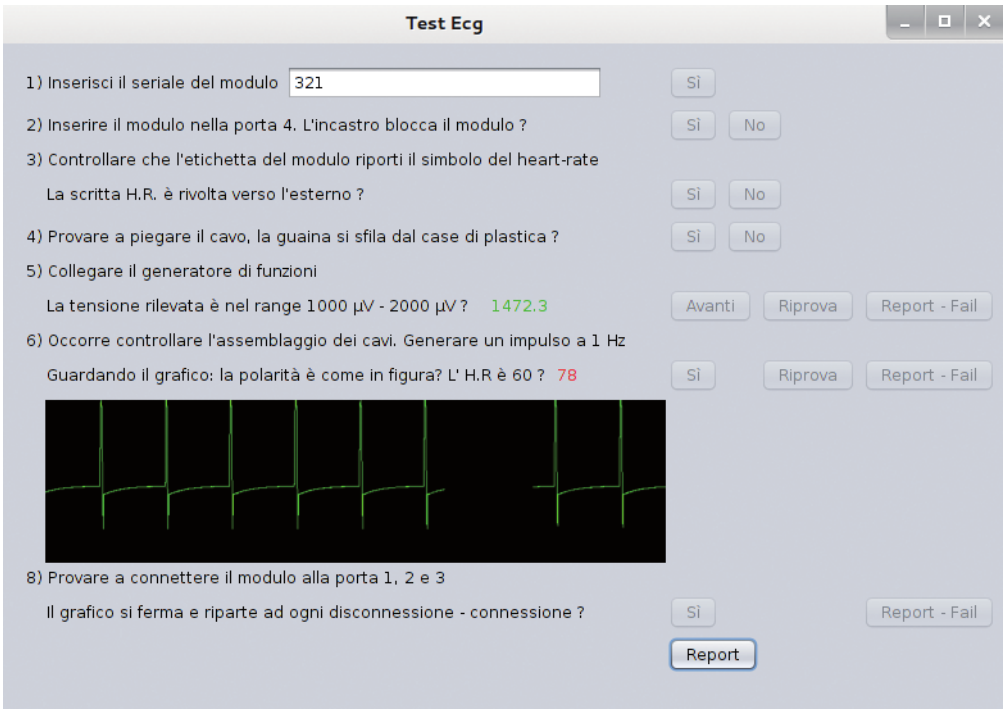


Figura 3.20 - Frame di collaudo di Ecg

È la classe TestEcg.java che si occupa della gestione del *frame* in questione. Ad ogni step della procedura di collaudo la classe popola un *array* che viene poi passato alla classe Dopdf che si occupa di creare il report.

L'*array* in questione si chiama flag e contiene, ad ogni posizione, un valore 0,1 o 2, indicando così se un determinato step ha avuto successo, non ha avuto successo o non è mai stato effettuato.

Ad esempio, cliccando su No allo step 4 la classe svolge le operazioni:

```
1.  flag[ 2] =0;
2.  jButton5.setEnabled(false);
3.  jButton6.setEnabled(false);
4.  jLabel6.setVisible(true);
5.  jLabel7.setVisible(true);
6.  jButton7.setVisible(true);
7.  jButton8.setVisible(true);
8.  jButton9.setVisible(true);
9.  jLabel1.setVisible(true);
```

Invece cliccando sul tasto Report la classe instanzia una nuova istanza di DoPdf passando al suo costruttore l'*array* con i flag che vengono analizzati e con cui viene creato il report. Svolge infatti le seguenti operazioni:

1) Invoca la creazione di una nuova classe DoPdf.

```
1.  try {
2.  DoPdf report = new DoPdf(modulo, lotto, collaudatore, seriale,
    sessione, flag);
3.  } catch (IOException ex) {
4.  Logger.getLogger(TestE.class.getName()).log(Level.SEVERE,
    null, ex);
5.  } catch (DocumentException ex) {
6.  Logger.getLogger(TestE.class.getName()).log(Level.SEVERE,
    null, ex);
7.  }
```

2) Chiude il Frame.

```
8.  dispose();
```

La classe DoPdf.java infine attraverso uno switch distingue quale classe di test l'ha creata e analizzando l'*array* di flag produce il documento di report. In pratica svolge le seguenti operazioni:

1) Nel costruttore memorizza i dati in variabili interne

```
1.  DoPdf(String who, String lot, String coll, String SN, String
    sess, int[] f) {
2.    modulo = who;
3.    collaudatore = coll;
4.    lotto = lot;
5.    serial = SN;
6.    sessione = sess;
7.    flag = f;
8.  }
```

2) Crea il file con il nome corretto, che come già descritto, deve avere la forma TipoModulo_Numerodilotto_seriale_data_collaudatore.pdf

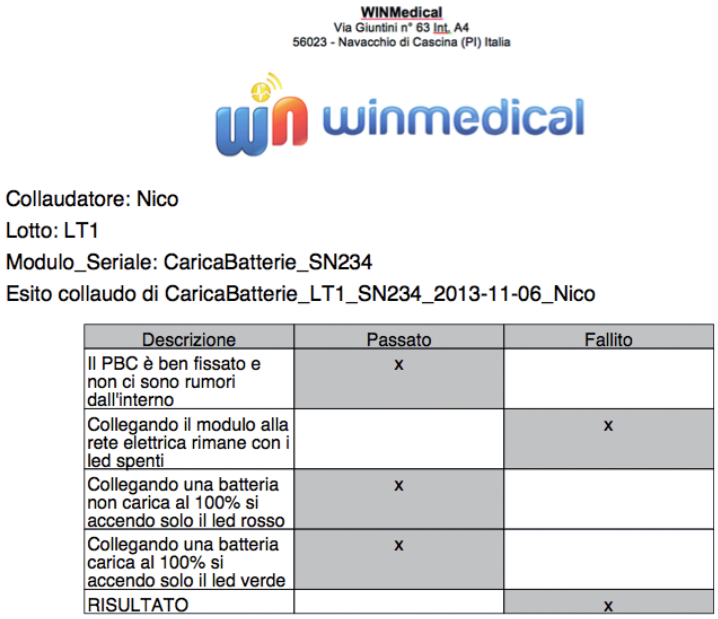
```
1.  Document document = new Document();
2.  try {
3.    PdfWriter.getInstance(document, new FileOutputStream(
4.    "./Results/"+who+"/"+who+"_"+lotto+"_"+SN+"_"+Sdate+"_"+
      collaudatore+".pdf"));
```

```
5.  } catch (DocumentException ex) {
6.    Logger.getLogger(DoPdf.class.getName()).log(Level.SEVERE,
      null, ex);
7.  }
8.  document.open();
```

3) Distingue, a seconda del modulo che l'ha creata, cosa scrivere nella tabella riassuntiva sul report di collaudo

```
1.  case "Ecg":
2.  tabella.addCell("Provando a piegare il cavo la guaina non
    si sfila dal case di plastica");
3.    c1.setPhrase(new Phrase("x"));
4.    c1.setHorizontalAlignment(Element.ALIGN_CENTER);
5.    if (flag[2]==1)
6.    {
7.      tabella.addCell(c1);
8.      tabella.addCell("");
9.    }
10.   if (flag[2]==0)
11.   {
12.     tabella.addCell("");
13.     tabella.addCell(c1);
14.   test_fail = true;
15.   }
16.   if (flag[2]==2)
17.   {
18.   tabella.addCell("");
19.   tabella.addCell("");
20.
21.   }
22.
23. tabella.addCell("Collegando il generatore di funzioni al modulo
    la risposta è corretta");
24.   c1.setPhrase(new Phrase("x"));
25.   c1.setHorizontalAlignment(Element.ALIGN_CENTER);
26.   if (flag[3]==1)
27.   {
28.     tabella.addCell(c1);
29.     tabella.addCell("");
30.   }
31.   if (flag[3]==0)
32.   {
33.     tabella.addCell("");
34.     tabella.addCell(c1);
35.     test_fail = true;
36.   }
37.   if (flag[3]==2)
38.   {
39.     tabella.addCell("");
40.     tabella.addCell("");
41.     ...
42.   }
```

Un esempio del file di report così creato è visibile in figura.



WINMedical
Via Giuntini n° 63 Int. A4
56023 - Navacchio di Cascina (PI) Italia

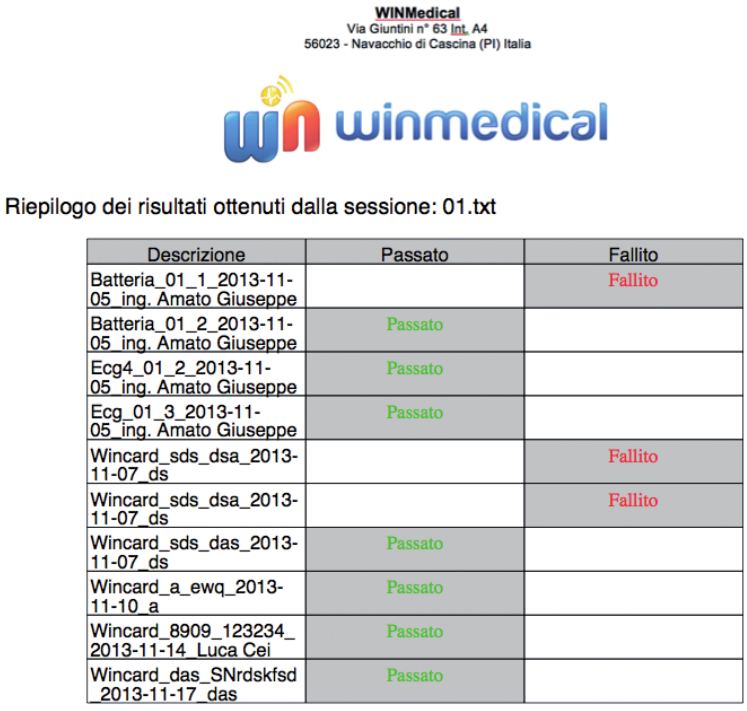
winmedical

Collaudatore: Nico
Lotto: LT1
Modulo_Seriale: CaricaBatterie_SN234
Esito collaudo di CaricaBatterie_LT1_SN234_2013-11-06_Nico

Descrizione	Passato	Fallito
Il PBC è ben fissato e non ci sono rumori dall'interno	x	
Collegando il modulo alla rete elettrica rimane con i led spenti		x
Collegando una batteria non carica al 100% si accendo solo il led rosso	x	
Collegando una batteria carica al 100% si accendo solo il led verde	x	
RISULTATO		x

Figura 3.21 - Report di collaudo di CaricaBatteria

Un report di sessione ha invece il seguente layout.



WINMedical
Via Giuntini n° 63 Int. A4
56023 - Navacchio di Cascina (PI) Italia

winmedical

Riepilogo dei risultati ottenuti dalla sessione: 01.txt

Descrizione	Passato	Fallito
Batteria_01_1_2013-11-05_ing. Amato Giuseppe		Fallito
Batteria_01_2_2013-11-05_ing. Amato Giuseppe	Passato	
Ecg4_01_2_2013-11-05_ing. Amato Giuseppe	Passato	
Ecg_01_3_2013-11-05_ing. Amato Giuseppe	Passato	
Wincard_sds_dsa_2013-11-07_ds		Fallito
Wincard_sds_dsa_2013-11-07_ds		Fallito
Wincard_sds_das_2013-11-07_ds	Passato	
Wincard_a_ewq_2013-11-10_a	Passato	
Wincard_8909_123234_2013-11-14_Luca Cei	Passato	
Wincard_das_SNrdskfsd_2013-11-17_das	Passato	

Figura 3.22 - Report di sessione

4. RISULTATI OTTENUTI

Gli strumenti introdotti hanno colmato un gap di strumenti informatici all'interno dell'azienda per gestire al meglio le procedure di collaudo e test aumentando sicuramente la velocità delle operazioni e garantendo allo stesso tempo l'assenza di errori da parte dell'operatore tecnico che si occupa delle varie fasi.

È possibile quantificare questi risultati:si stima che il risparmio in termini di tempo per eseguire il collaudo su un singolo modulo va dal 50% al 70% (la forbice dipende dal tipo di modulo che si vuole collaudare). Dove è preponderante la produzione di documenti rispetto alla fase 'tecnica' di collaudo il risparmio raggiunge il 70% del tempo totale;dove invece la fase di collaudo, statisticamente, fa impiegare il medesimo tempo per produrre documenti rispetto, ad esempio, alla configurazione degli oscilloscopi o dell'ispezione visiva, il risparmio rimane comunque notevole, cioè il 50% del tempo totale per il collaudo.

Ancor più buoni sono i risultati ottenuti in termini di errori commessi:dato che la produzione di documenti prevede l'inserimento di codici alfanumerici (che si riferiscono al lotto di produzione) e che i file contengono, a partire dal titolo, una serie di caratteri alfanumerici (come ad esempio *underscore*), gli errori da parte degli operatori sono molto frequenti e l'introduzione di questo sistema di automatizzazione riduce del 90% il numero totale degli errori commessi.

In termini di risultati ottenuti un altro aspetto non banale da considerare è il risparmio aziendale che può nascere dall'utilizzo di questo sistema automatizzato. Infatti sarà possibile far svolgere le operazioni di collaudo ad un personale meno tecnico o comunque meno formato, portando l'azienda a risparmiare in termini economici.

Infine è possibile dotare direttamente il fornitore di questo strumento, abbattendo così il numero totale di dispositivi da collaudare in azienda, che passerebbero dal 100% ad un campione per ogni lotto.

APPENDICE A: CODICE ESTENSIONE WINMULTISERVER

Poolgroup.cpp:

```
1.  #include "poolgroup.hpp"
2.  #include <iostream>
3.  #include <syslowlev.hpp>
4.
5.  using namespace syslowlev;
6.  using namespace wp1server;
7.  using namespace serverpool;
8.  using namespace poolgroup;
9.  using namespace std;
10.
11.
12. SERVERCOLLAUDI::SERVERCOLLAUDI() :
13.     SOCKET_TCP( TIMEOUT(1000) ),
14.     THREAD( THREAD::PRIORITY_NORMAL )
15. {
16.     acceptor = true;
17.     cancel = false;
18.     thread_go();
19. }
20.
21. SERVERCOLLAUDI::SERVERCOLLAUDI(SOCKET_TCP::pVIRTUAL& pv) :
22.     SOCKET_TCP(pv),
23.     THREAD( THREAD::PRIORITY_NORMAL )
24. {
25.     acceptor = false;
26.     cancel = false;
27.     thread_go();
28. }
29.
30. SERVERCOLLAUDI::~SERVERCOLLAUDI()
31. {
32.     { MUTEXLOCK ml( cancel_mutex );
33.     cancel = true; }
34.
35.     thread_waithalt();
36. }
37.
38. bool SERVERCOLLAUDI::link_cancel_io()
39. {
40.     MUTEXLOCK ml( cancel_mutex );
41.     return cancel;
42. }
43.
44. void SERVERCOLLAUDI::link_request_abort() {}
45.
46. void SERVERCOLLAUDI::thread_procedure()
47. {
48. if ( acceptor )
49.     {
50.       SOCKET_TCP::pVIRTUAL pv;
51.       SOCKET_TCP::tcp_addrport_t ap;
52.
```

```
53.          this->tcp_string_to_address("127.0.0.1:8000", ap);
54.
55.          tcp_bind(ap);
56.          tcp_listen(10);
57.
58.          for (;;)
59.          {
60.            if ( tcp_accept(ap, pv, TIMEOUTINF()) < 0 )
61.                  break;
62.
63.               pSERVERCOLLAUDI psrv = new SERVERCOLLAUDI(pv);
64.          }
65.      }
66.      else
67.      {
68.          for (;;)
69.          {
70.              char c;
71.          if ( link_recv((ui8_t*)&c, 1, TIMEOUTINF()) < 0 )
72.            break;
73.              cout << c;
74.          }
75.      }
76. }
77.
78.
79. /** exception poolgroup tcp */
80. static class POOLGROUP_BASE_EXCEPTION : public POOLGROUP_BASE::failure {
81. public:
82. const char* what() const throw() { return "class POOLGROUP_BASE error"; }
83. } poolgroup_base_exception;
84.
85. /** exception poolgroup FILE */
86. static class POOLGROUP_FILE_EXCEPTION : public POOLGROUP_BASE::failure {
87. public:
88. const char* what() const throw() { return "class POOLGROUP_FILE error"; }
89. } poolgroup_file_exception;
90.
91. /** exception poolgroup TOPLEVEL */
92. static class POOLGROUP_TOPLEVEL_EXCEPTION : public POOLGROUP_
    TOPLEVEL::failure {
93. public:
94. const char* what() const throw() { return "class POOLGROUP_TOPLEV
    error"; }
95. } poolgroup_toplevel_exception;
96.
97.
98. //*******************************************************
99. //
100.  // Gruppo di pool BASE
101.  // (deve essere ereditato)
102.  //
103.  //*******************************************************
104.  /** callback per notificare se il protocollo nascosto è abilitato */
105.  bool POOLGROUP_BASE::pool_is_hidden_enabled(ui16_t idmst,
      ui16_t idslv)
106.  {
107.      idmst = 0;
```

```
108.      idslv = 0;
109.      return ( group_firm_filename.size() > 0 );
110.  }
111.  /** callback che notifica la versione del firmware */
112.  bool POOLGROUP_BASE::pool_notify_firmware_version(ui16_t idmst,
      ui16_t idslv, ui32_t ver)
113.  {
114.  // informo della costruzione
115.     SYSLOGGER(SYSLOG::PRI_INFO, "class POOLGROUP_BASE" <<
116.        " idmst=" << idmst <<
117.        " idslv=" << idslv <<
118.        " version=" << ver );
119.  // tutto ok
120.      return true;
121.  }
122.  /** callback per aprire il nuovo firmware */
123.  std::istream* POOLGROUP_BASE::pool_firmware_open(ui16_t idmst,
      ui16_t idslv, ui32_t& size)
124.  {
125.    ifstream* file = new ifstream(group_firm_filename.c_str(),
      ifstream::binary | ifstream::in);
126.      if ( file != 0 )
127.      {
128.          if ( file->is_open() )
129.          {
130.              file->seekg(0, ifstream::end);
131.              size = (ui32_t)file->tellg();
132.              file->seekg(0, ifstream::beg);
133.          }
134.          else
135.          {
136.          delete file;
137.              file = 0;
138.          }
139.      }
140.      idmst = 0;
141.      idslv = 0;
142.      return file;
143.  }
144.  /** callback per chiudere il nuovo firmware */
145.  void POOLGROUP_BASE::pool_firmware_close(ui16_t idmst,
      ui16_t idslv, std::istream& file)
146.  {
147.      idmst = 0;
148.      idslv = 0;
149.      delete &file;
150.  }
151.  /** Avvio il thread interno.
152.  *  La classe che possiede il pool
153.  *  deve chiamare questa funzione dopo
154.  *  la inizializzazione. */
155.  void POOLGROUP_BASE::group_go()
156.  {
157.  // avvio tutti i pool
158.  for ( list<SERVERPOOL_BASE*>::iterator i=group_list.begin();
      i!=group_list.end(); i++ )
159.           (*i)->pool_go();
160.  }
```

```
161.  /** Distrugge tutti i pool e, quindi,
162.  *  tutti i server. È il complementare
163.  *  di group_go(), da chiamare alla
164.  *  chiusura della classe madre per garantire
165.  *  che tutti i thread siano spenti prima di
166.  *  distruggere delle risosrse condivise. */
167.  void POOLGROUP_BASE::group_destroy()
168.  {
169.  // segnalo lo stop a tutti i pool
170.  for ( list<SERVERPOOL_BASE*>::iterator i=group_list.begin();
      i!=group_list.end(); i++ )
171.           (*i)->pool_signal_stop();
172.  // distruzione di tutti i pool
173.      while ( !group_list.empty() )
174.      {
175.          delete group_list.front();
176.          group_list.pop_front();
177.      }
178.  }
179.  /** costruttore */
180.  POOLGROUP_BASE::POOLGROUP_BASE(const CFGTREE& cfg)
181.  {  try {
182.      list<pCFGTREE>  nl;
183.
184.  // controllo se esiste un firmware da programmare
185.      cfg.select_node("firmware", nl);
186.      if ( nl.size() > 1 )
187.      {
188.          throw 0;
189.      }
190.      else if ( nl.size() == 1 )
191.      {
192.   // prelevo il nome del file
193.      group_firm_filename = nl.front()->get_value();
194.  // provo ad aprirlo
195.      ifstream f(group_firm_filename.c_str(), ifstream::binary |
      ifstream::in);
196.  // se non è stato aperto annullo il nome del file
197.  //if ( !f.is_open() )
198.  //  group_firm_filename.clear();
199.      }
200.  // ripulisco la lista
201.      nl.clear();
202.
203.  // estraggo i descrittori dei server
204.      cfg.select_node("serverpool_tcp", nl);
205.  // per ogni descrittore creo un serverpool di tipo tcp
206.   for ( list<pCFGTREE>::const_iterator i=nl.begin(); i!=nl.end(); i++ )
207.      {
208.  // creo il server
209.      pSERVERPOOL_BASE psrv = new SERVERPOOL_TCP(**i, this);
210.          if ( psrv == 0 )
211.              throw 1;
212.          group_list.push_back(psrv);
213.      }
214.  // ripulisco la lista
215.      nl.clear();
216.
```

```
217.  #ifndef Winmultiserver_NOSSL
218.  // estraggo i descrittori dei server
219.      cfg.select_node("serverpool_ssl", nl);
220.  // per ogni descrittore creo un serverpool di tipo ssl
221.  for ( list<pCFGTREE>::const_iterator i=nl.begin(); i!=nl.end(); i++ )
222.      {
223.   // creo il server
224.      pSERVERPOOL_BASE psrv = new SERVERPOOL_SSL(**i, this);
225.      if ( psrv == 0 )
226.        throw 1;
227.      group_list.push_back(psrv);
228.      }
229.  // ripulisco la lista
230.      nl.clear();
231.  #endif
232.
233.  #ifndef Winmultiserver_NOBLUE
234.  // estraggo i descrittori dei server
235.      cfg.select_node("serverpool_rfcomm", nl);
236.  // per ogni descrittore creo un serverpool di tipo tcp
237.      for ( list<pCFGTREE>::const_iterator i=nl.begin();
        i!=nl.end(); i++ )
238.      {
239.  // creo il server
240.    pSERVERPOOL_BASE psrv = new SERVERPOOL_RFCOMM(**i, this);
241.          if ( psrv == 0 )
242.              throw 1;
243.          group_list.push_back(psrv);
244.      }
245.  // ripulisco la lista
246.      nl.clear();
247.  #endif
248.
249.  // deve esistere almeno un pool
250.      if ( group_list.size() == 0 )
251.          throw 1;
252.
253.  } catch (si32_t e) {
254.      switch (e)
255.      {
256.      case 1:
257.          while ( !group_list.empty() )
258.          {
259.              delete group_list.front();
260.              group_list.pop_front();
261.          }
262.      default:
263.          throw poolgroup_base_exception;
264.      }
265.  }
266.
267.  }
268.  /** distruttore */
269.  POOLGROUP_BASE::~POOLGROUP_BASE()
270.  {
271.  // distruggo tutti i pool
272.      group_destroy();
273.  }
```

```
274.
275.
276.
277.  //*************************************************************
278.  //
279.  //  Gruppo di pool FILE, scrive tutto su file
280.  //
281.  //*************************************************************
282.  /** callback dai server,
283.   *  viene chiamata per notificare un pacchetto s2c */
284.  bool POOLGROUP_FILE::pool_notify_s2cdata(
285.          ui16_t idmst,
286.          ui16_t idslv,
287.          const std::vector<SERVER::TOKEN>& tokvec,
288.          const ui8_t* packet_raw, ui32_t packet_rawlen
289.      )
290.  {
291.  // dummy, no warning
292.      packet_raw = 0;
293.      packet_rawlen = 0;
294.   // blocco il file
295.      file_mutex.lock();
296.  // se il file è aperto
297.  // allora ci scrivo sopra
298.      if ( file_out->is_open() )
299.      {
300.  // stampo le informazioni di ogni token
301.  for (std::vector<SERVER::TOKEN>::const_iterator i=tokvec.
      begin();
302.          i!=tokvec.end();
303.          i++ )
304.          {
305.        SERVER::TOKEN tok = *i;
306.          *file_out << dec << idmst << ", ";
307.          *file_out << dec << idslv << ", ";
308.          *file_out << dec << ((ui32_t)tok.type) << ", ";
309.          *file_out << dec << tok.rtcms << ", ";
310.    *file_out << hex << uppercase << setfill('0') << setw(8)
              << tok.devid << ", ";
311.    *file_out << hex << uppercase << setfill('0') << setw(8)
              << tok.serid << ", ";
312.    *file_out << dec << tok.length << ", ";
313.        for ( ui32_t i=0; i<tok.length; i++ )
314.              {
315.    *file_out << hex << uppercase << setfill('0') << setw(2)
            << ((ui32_t)tok.payload[ i] );
316.              }
317.              *file_out << endl;
318.          }
319.      }
320.  // sblocco il file
321.      file_mutex.unlock();
322.  // accetto sempre i token
323.      return true;
324.  }
325.  /** costruttore */
326.  POOLGROUP_FILE::POOLGROUP_FILE(const CFGTREE& cfg) :
327.      POOLGROUP_BASE(cfg)
```

```
328.  {
329.  // informo della costruzione
330.    SYSLOGGER(SYSLOG::PRI_INFO, "class POOLGROUP_FILE created");
331.
332.  // apro il file in uscita contenente i token
333.      file_out = new ofstream();
334.      if ( file_out == 0 )
335.          throw poolgroup_file_exception;
336.  // estraggo il nome del file in uscita
337.      file_out->open(cfg.get_value().c_str(),
338.      ofstream::binary | ofstream::trunc | ofstream::out);
339.  // informo sulla apertura del file
340.      SYSLOGGER(SYSLOG::PRI_INFO,
341.          "class POOLGROUP_GENERIC file: "    <<
342.          cfg.get_value()         <<
343.          " open status: "            <<
344.          file_out->is_open() );
345.  // avvio i pool
346.      group_go();
347.  }
348.  /** distruttore */
349.  POOLGROUP_FILE::~POOLGROUP_FILE()
350.  {
351.  // distruggo tutti i pool
352.      group_destroy();
353.
354.  // chiusura del file dei token
355.      file_out->close();
356.      delete file_out;
357.
358.  // informo della costruzione
359.  SYSLOGGER(SYSLOG::PRI_INFO, "class POOLGROUP_FILE destroyed");
360.  }
361.
362.  // Gruppo di pool COLLAUDI
363.
364.  /** check stop signal */
365.  bool POOLGROUP_COLLAUDI::collaudi_cancel_io()
366.  {
367.      MUTEXLOCK mlock(collaudi_stop_mutex);
368.      return collaudi_stop;
369.  }
370.  /** ferma le io ZeroMQ */
371.  void POOLGROUP_COLLAUDI::collaudi_cancel_request()
372.  {
373.      MUTEXLOCK mlock(collaudi_stop_mutex);
374.  // segnalo lo stop
375.      collaudi_stop = true;
376.  }
377.  /** crea e connette un socket */
378.  si32_t POOLGROUP_COLLAUDI::collaudi_connect(ui16_t idslv)
379.  {
380.    // se la io è cancellata, esco con errore
381.      if ( collaudi_cancel_io() )
382.          return -1;
383.
384.      // se il socket non è presente lo creo e lo connetto
385.      if ( collaudi_sock_vector[ idslv]  != 0 )
```

```
386.          return 0;
387.
388.      ZeroMQ::socket_t* s = 0;
389.      // creazione socket
390.      try {
391.          // apro il socket
392.      try { s = new ZeroMQ::socket_t(collaudi_sock_context,
          ZeroMQ_REQ); }
393.          catch ( ZeroMQ::error_t e ) { e.what();throw 0; }
394.          // imposto l'identificativo
395.        /*
396.          ui8_t ZeroMQid[3] = { 0x01,
397.             (ui8_t)((idslv>>0) & 0x00FF),
398.          (ui8_t)((idslv>>8) & 0x00FF) };
399.       try { s->setsockopt(ZeroMQ_IDENTITY, ZeroMQid, sizeof(ZeroMQid)); }
400.     catch ( ZeroMQ::error_t e ) { e.what();throw 1; }
401.     */
402.        // LINGER nullo, non mantiene i pacchetti in uscita
403.          int ZeroMQlinger = 0;
404.      try { s->setsockopt(ZeroMQ_LINGER, &ZeroMQlinger,
            sizeof(ZeroMQlinger)); }
405.          catch ( ZeroMQ::error_t e ) { e.what();throw 1; }
406.          // timeout ricezione e trasmissione nullo,
407.          // non bloccante
408.          int ZeroMQtmout = 0;
409.      try { s->setsockopt(ZeroMQ_RCVTIMEO, &ZeroMQtmout, sizeof(Zero
          MQtmout)); }
410.          catch ( ZeroMQ::error_t e ) { e.what();throw 1; }
411.      try { s->setsockopt(ZeroMQ_SNDTIMEO, &ZeroMQtmout, sizeof(Zero
          MQtmout)); }
412.          catch ( ZeroMQ::error_t e ) { e.what();throw 1; }
413.          // connetto il socket
414.          try { s->connect( collaudi_endpoint.c_str() ); }
415.          catch ( ZeroMQ::error_t e ) { e.what();throw 1; }
416.         // assegno il socket nel vettore
417.          collaudi_sock_vector[ idslv] = s;
418.
419.      // cattura eccezioni
420.     } catch ( int e ) {
421.          switch ( e ) {
422.          case 1:
423.              delete s;
424.          default:
425.      // azzero il socket nel vettore
426.        collaudi_sock_vector[ idslv] = 0;
427.      // informo della connessione su ZeroMQ
428.  SYSLOGGER(SYSLOG::PRI_ERR, "class SERVERGROUP_COLLAUDI" <<
429.        " idslv=" << idslv <<
430.        " connesione ZeroMQ FALLITA" );
431.           return -1;
432.          }
433.     }
434.
435.  // informo della connessione su ZeroMQ
436.    SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI" <<
437.                  " idslv=" << idslv <<
438.                  " connesione ZeroMQ avvenuta" );
439.
```

```
440.        // socket creato con successo
441.        return 0;
442.  }
443.  /** disconnetti il socket ZeroMQ */
444.  void POOLGROUP_COLLAUDI::collaudi_disconnect(ui16_t idslv)
445.  {
446.        // distruggo il socket e lo
447.       // sostituisco con un puntatore nullo
448.        delete collaudi_sock_vector[ idslv] ;
449.        collaudi_sock_vector[ idslv]  = 0;
450.  }
451.  /** lowlevel send */
452.  si32_t POOLGROUP_COLLAUDI::collaudi_send_ll(ZeroMQ::socket_
      t& s, ZeroMQ::message_t &msg, bool more)
453.  {
454.        bool res;
455.        si32_t msgsize = (si32_t)msg.size();
456.       // provo a trasmettere il ,messaggio
457.        try {  res = s.send(msg, ((more)?ZeroMQ_SNDMORE:0) ); }
458.        catch ( ZeroMQ::error_t e ) {  e.what();return -1; }
459.        // se ha trasmesso ritorno la dimensione
460.        // altrimenti ritorno 0
461.        return (si32_t)((res)?msgsize:0);
462.  }
463.  /** lowlevel recv */
464.  si32_t POOLGROUP_COLLAUDI::collaudi_recv_ll(ZeroMQ::socket_
      t& s, ZeroMQ::message_t &msg)
465.  {
466.        bool res;
467.      // provo a trasmettere il ,messaggio
468.        try {  res = s.recv(&msg, 0); }
469.        catch ( ZeroMQ::error_t e ) {  e.what();return -1; }
470.       // se ha trasmesso ritorno la dimensione
471.        // altrimenti ritorno 0
472.        return (si32_t)((res)?msg.size():0);
473.  }
474.  /** lowlevl select recv */
475.  si32_t POOLGROUP_COLLAUDI::collaudi_select_recv_ll(ZeroMQ::
      socket_t& s)
476.  {
477.        si32_t selres;
478.        ZeroMQ::pollitem_t sv;
479.       // inizializzo il selettore
480.        sv.fd = 0;
481.        sv.revents = 0;
482.        sv.socket = ((void*)s);
483.        sv.events = ZeroMQ_POLLIN;
484.        // provo la select
485.        try {  selres = ZeroMQ::poll(&sv, 1, (long)(collaudi_
           partial_tmout.us_get())); }
486.        catch ( ZeroMQ::error_t e ) {  e.what();return -1; }
487.        // ritorno il numero di socket selezionati
488.        return (si32_t)selres;
489.  }
490.  /** lowlevl select send */
491.  si32_t POOLGROUP_COLLAUDI::collaudi_select_send_ll
      ZeroMQ::socket_t& s)
492.  {
```

```
493.      si32_t selres;
494.      ZeroMQ::pollitem_t sv;
495.     // inizializzo il selettore
496.      sv.fd = 0;
497.      sv.revents = 0;
498.      sv.socket = ((void*)s);
499.      sv.events = ZeroMQ_POLLOUT;
500.     // provo la select
501.      try { selres = ZeroMQ::poll(&sv, 1, (long)(collaudi_
          partial_tmout.us_get())); }
502.      catch ( ZeroMQ::error_t e ) { e.what();return -1; }
503.      // ritorno il numero di socket selezionati
504.      return (si32_t)selres;
505.  }
506.  /** send not blocking */
507.  si32_t POOLGROUP_COLLAUDI::collaudi_send(ui16_t idslv,
      ZeroMQ::message_t &msg, bool more)
508.  {
509.      si32_t      byteswritten;
510.
511.      // prendo il socket dalla tabella
512.      ZeroMQ::socket_t* s = collaudi_sock_vector[idslv];
513.
514.      for (;;) {
515.     // controllo se la IO è stata abortita
516.          if ( collaudi_cancel_io() )
517.              return -1;
518.     // provo a trasmettere il buffer
519.          byteswritten = collaudi_send_ll(*s, msg, more);
520.     // errore??
521.          if ( byteswritten < 0 )
522.              return -1;
523.     // nessun byte trasmesso??
524.     // attendi la disponibilita' del socket
525.          else if ( byteswritten == 0 )
526.          {
527.       // se il timeout è maggiore di 0
528.       // controlla che non scatti
529.              if ( collaudi_total_tmout > 0 )
530.              {
531.                  TIMEUS  tini(0), tfin(0);
532.                  si32_t  selres;
533.        // istante di tempo iniziale
534.                  tini.us_get_system();
535.         // ciclo fino al timeout o alla
536.         // disponibilita' del socket
537.             do {
538.         // controllo se la IO è stata abortita
539.           if ( collaudi_cancel_io() )
540.              return -1;
541.           // provo ad attendere parzialmente
542.           selres = collaudi_select_send_ll(*s);
543.           if ( selres < 0 )
544.              return -1;
545.           else if ( selres > 0 )
546.              break;
547.           // acquisisco il tempo attuale
548.              tfin.us_get_system();
```

```
549.      } while ( collaudi_total_tmout.tmout_isinf() ||
       (tfin-tini)<collaudi_total_tmout );
550.          // se non si è liberato il socket entro il timeout esco
551.          if ( selres == 0 )
552.            break;
553.             }
554.          else
555.              {
556.   // timeout nullo, esci e basta
557.            break;
558.                }
559.             }
560.          else
561.            {
562.         // esco dal ciclo di trasmissione
563.            break;
564.             }
565.       }
566.
567.     // ritorno i bytes trasmessi
568.        return byteswritten;
569.  }
570.  /** recv not blocking */
571.  si32_t POOLGROUP_COLLAUDI::collaudi_recv(ui16_t idslv,
      ZeroMQ::message_t &msg)
572.  {
573.       si32_t      bytesread;
574.
575.     // prendo il socket dalla tabella
576.        ZeroMQ::socket_t* s = collaudi_sock_vector[ idslv] ;
577.
578.        for (;;) {
579.       // controllo se la IO è stata abortita
580.            if ( collaudi_cancel_io() )
581.                return -1;
582.       // provo a trasmettere il buffer
583.            bytesread = collaudi_recv_ll(*s, msg);
584.       // errore??
585.            if ( bytesread < 0 )
586.                return -1;
587.       // nessun byte trasmesso??
588.       // attendi la disponibilita' del socket
589.            else if ( bytesread == 0 )
590.            {
591.         // se il timeout è maggiore di 0
592.         // controlla che non scatti
593.                if ( collaudi_total_tmout > 0 )
594.                {
595.                     TIMEUS  tini(0), tfin(0);
596.                     si32_t  selres;
597.           // istante di tempo iniziale
598.                     tini.us_get_system();
599.           // ciclo fino al timeout o alla
600.           // disponibilita' del socket
601.                     do {
602.             // controllo se la IO è stata abortita
603.                 if ( collaudi_cancel_io() )
604.                    return -1;
```

```
605.            // provo ad attendere parzialmente
606.                        selres = collaudi_select_recv_ll(*s);
607.                        if ( selres < 0 )
608.                            return -1;
609.                        else if ( selres > 0 )
610.                            break;
611.            // acquisisco il tempo attuale
612.                        tfin.us_get_system();
613.    } while ( collaudi_total_tmout.tmout_isinf() ||
        (tfin-tini)<collaudi_total_tmout );
614.          // se non si è liberato il socket entro il timeout esco
615.                    if ( selres == 0 )
616.                        break;
617.                }
618.                else
619.                {
620.          // timeout nullo, esci e basta
621.                    break;
622.                }
623.            }
624.            else
625.            {
626.        // esco dal ciclo di trasmissione
627.                break;
628.            }
629.       }
630.
631.     // ritorno i bytes trasmessi
632.        return bytesread;
633.  }
634.  /** questa funzione non libera la memoria
635.   * del tipo ZeroMQ::message_t */
636.  void server_group_all_collaudi_msg_nofreedata(void
      *data, void *hint)
637.  {
638.     // dummy instruction for warning suppression
639.        data = 0;
640.        hint = 0;
641.  }
642.  /** callback dai server,
643.   * viene chiamata per notificare un pacchetto s2c */
644.  bool POOLGROUP_COLLAUDI::pool_notify_s2cdata(
645.                    ui16_t idmst,
646.                    ui16_t idslv,
647.              const std::vector<SERVER::TOKEN>& tokvec,
648.
               const ui8_t* packet_raw, ui32_t packet_rawlen
649.        )
650.  {
651.        MUTEXLOCK mlock(collaudi_sock_mutex[ idslv] );
652.
653.     // se il socket non è presente lo creo e lo connetto
654.        if ( collaudi_connect(idslv) < 0 )
655.            return false;
656.
657.     // informo del tentativo connessione su ZeroMQ
658.     SYSLOGGER(SYSLOG::PRI_DEBUG, "class SERVERGROUP_COLLAUDI" <<
659.        " packet " <<
```

```
660.      " idslv=" << idslv <<
661.      " numoftoken=" << tokvec.size() );
662.
663.    // tutti i token devono essere accettati
664.    // questa variabile viene posta a false
665.    // appena un token viene rifiutato
666.       bool    tokens_accepted = true;
667.
668.    // sequenza di request e replay
669.       try {
670.           si32_t zres;
671.
672.        // tempo iniziale
673.                TIMEUS tini;
674.
675.       // compongo il messaggio parziale con
676.       // lo slave id
677.           ui8_t ZeroMQid[ 2] = {   (ui8_t)((idslv>>0) & 0x00FF),
678.                                (ui8_t)((idslv>>8) & 0x00FF) };
679.           ZeroMQ::message_t ZeroMQidmsg(ZeroMQid,
680.                               sizeof(ZeroMQid),
681.  (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
682.                               0 );
683.       // spedisco il pacchetto
684.           zres = collaudi_send(idslv, ZeroMQidmsg, true);
685.           if ( zres == 0 )
686.               throw 0;
687.           else if ( zres < 0 )
688.               throw 1;
689.
690.
691.
692.       // stampo le informazioni di ogni token
693.           for ( ui32_t i=0; i<tokvec.size(); i++ )
694.           {
695.               SERVER::TOKEN tok = tokvec[ i];
696.         // compongo il pacchetto
697.
698.               ui32_t ZeroMQid = {     tok.devid };
699.               ZeroMQ::message_t ZeroMQdeviceid(&ZeroMQid,
700.                   sizeof(&ZeroMQid),
701.  (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
702.                            0 );
703.         // spedisco il pacchetto
704.
          zres = collaudi_send(idslv, ZeroMQdeviceid, true);
705.               if ( zres == 0 )
706.                   throw 0;
707.               else if ( zres < 0 )
708.                   throw 1;
709.
710.  /*
711.         ui32_t ZeroMQser[4] = {    tok.serid };
712.         ZeroMQ::message_t ZeroMQserialid(ZeroMQser,
713.                sizeof(ZeroMQser),
714.                (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
715.                0 );
716.         // spedisco il pacchetto
```

```
717.         zres = collaudi_send(idslv, ZeroMQserialid, true);
718.         if ( zres == 0 )
719.           throw 0;
720.         else if ( zres < 0 )
721.           throw 1;
722.  */
723.
724.                 ui64_t ZeroMQrtc = {    tok.rtcms };
725.
726.                 ZeroMQ::message_t ZeroMQrtcms(&ZeroMQrtc,
727.                                 sizeof(ZeroMQrtc),
728.(ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
729.                                 0 );
730.
731.         //ui8_t ZeroMQrtc[8] = {   tok.rtcms };
732.         //ZeroMQ::message_t ZeroMQrtcms(ZeroMQrtc,
733.         //       sizeof(ZeroMQrtc),
734.         //       (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
735.         //       0 );
736.         // spedisco il pacchetto
737.       zres = collaudi_send(idslv, ZeroMQrtcms, true);
738.                 if ( zres == 0 )
739.                     throw 0;
740.                 else if ( zres < 0 )
741.                     throw 1;
742.
743.                 ZeroMQ::message_t msg( (void*)tok.payload,
744.                                 tok.length,
745.  (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
746.                                 0 );
747.         // spedisco il pacchetto
748.                 bool more = (i+1) < tokvec.size();
749.                 zres = collaudi_send(idslv, msg, more);
750.                 if ( zres == 0 )
751.                     throw 0;
752.                 else if ( zres < 0 )
753.                     throw 1;
754.
755.           }
756.  /*
757.       // compongo il pacchetto
758.       ZeroMQ::message_t msg( (void*)packet_raw,
759.               packet_rawlen,
760.  (ZeroMQ_free_fn*)&server_group_all_collaudi_msg_nofreedata,
761.               0 );
762.       // spedisco il pacchetto
763.       zres = collaudi_send(idslv, msg, false);
764.       if ( zres == 0 )
765.         throw 0;
766.       else if ( zres < 0 )
767.         throw 1;
768.  */
769.       // ricevo la risposta
770.            ZeroMQ::message_t rep;
771.            zres = collaudi_recv(idslv, rep);
772.            if ( zres == 0 )
773.                throw 0;
774.            else if ( zres < 0 )
```

```
775.              throw 1;
776.      // controllo la risposta
777.           ui8_t *repdata = (ui8_t*)rep.data();
778.           ui32_t replen = (ui32_t)rep.size();
779.           if ( replen != 1 )
780.              throw 2;
781.      // è stato accettato il token?
782.    tokens_accepted = tokens_accepted && (repdata[ 0] ==0x01);
783.
784.        // tempo finale
785.              TIMEUS tfin;
786.              TIMEUS tdiff(tfin-tini);
787.        // stampa del tempo impiegato
788.              { MUTEXLOCK mlock(collaudi_time_mutex);
789.     cout << setfill(' ') << setw(7) << (tdiff.us_get()/1000) << ".";
790.   cout << setfill('0') << setw(3) << (tdiff.us_get()%1000) << endl; }
791.
792.     // catch degli errori logici
793.       } catch ( int e ) {
794.      // disconnetto il socket
795.           collaudi_disconnect(idslv);
796.      // informo sulla natura dell'errore
797.           switch ( e ) {
798.           case 0:
799.     SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI" <<
800.                            " idslv=" << idslv <<
801.                            " ZeroMQ TIMEOUT" );
802.              break;
803.           case 1:
804.     SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI" <<
805.                            " idslv=" << idslv <<
806.                            " ZeroMQ ERROR" );
807.              break;
808.           case 2:
809.           default:
810.   SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI" <<
811.                            " idslv=" << idslv <<
812.                            " ZeroMQ wrong replay length" );
813.              break;
814.           }
815.      // ritorno token scartato
816.           return false;
817.       }
818.
819.       return tokens_accepted;
820.   }
821.   /** costruttore */
822.   POOLGROUP_COLLAUDI::POOLGROUP_COLLAUDI(const CFGTREE& cfg) :
823.       POOLGROUP_BASE(cfg),
824.       collaudi_sock_context(1),
825.       collaudi_partial_tmout(TIMEMS(100)),
826.       collaudi_total_tmout(TIMEOUT(2000))
827.   {
828.     // informo sulla distruzione dei pool
829.   SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI created");
830.
831.     // prelevo l'indirizzo del endpoint
832.       collaudi_endpoint = cfg.get_value();
```

```
833.    // non stopppato
834.       collaudi_stop = false;
835.    // massimo numero di socket apribili
836.       collaudi_sock_maxnum = 1<<16;
837.    // annullo tutti i socket aperti
838.       for ( ui32_t i=0; i<collaudi_sock_maxnum; i++ )
839.           collaudi_sock_vector[ i] = 0;
840.    // avvio i pool
841.       group_go();
842.  }
843.  /** distruttore */
844.  POOLGROUP_COLLAUDI::~POOLGROUP_COLLAUDI()
845.  {
846.    // fermo le io ZeroMQ
847.       collaudi_cancel_request();
848.    // distruggo tutti i pool
849.       group_destroy();
850.
851.    // annullo tutti i socket aperti
852.       for ( ui32_t i=0; i<collaudi_sock_maxnum; i++ )
853.       {
854.       if ( collaudi_sock_vector[ i] != 0 )
855.               collaudi_disconnect( (ui16_t)i );
856.       }
857.
858.    // informo sulla distruzione dei pool
859.       SYSLOGGER(SYSLOG::PRI_INFO, "class SERVERGROUP_COLLAUDI
          destroyed");
860.  }
861.
862.
863.
864.  #endif
```

APPENDICE B: CODICE 3LDBEXTRACTOR

Frame.java:

```
1.  package pkg3logic_dbextractor;
2.
3.  import com.mongodb.*;
4.  import java.net.UnknownHostException;
5.  import java.util.Set;
6.  import com.toedter.calendar.*;
7.  import java.io.File;
8.  import java.io.FileWriter;
9.  import java.io.IOException;
10. import java.text.DateFormat;
11. import java.text.DecimalFormat;
12. import java.text.NumberFormat;
13. import java.text.ParseException;
14. import java.text.SimpleDateFormat;
15. import java.util.*;
16. import java.util.logging.Level;
17. import java.util.logging.Logger;
18. import org.bson.BasicBSONObject;
19. import org.bson.types.BSONTimestamp;
20.
21. /**
22.  *
23.  * @author nino
24.  */
25.
26. public class NewJFrame extends javax.swing.JFrame {
27.
28.   /**
29.    * Creates new form NewJFrame
30.    */
31.      public NewJFrame() {
32.           initComponents();
33.           setLocationRelativeTo(this);
34.
35.      }
36.
37.   /**
38.    * This method is called from within the constructor to initialize the form.
39.    * WARNING: Do NOT modify this code. The content of this method is always
40.    * regenerated by the Form Editor.
41.    */
42.   @SuppressWarnings("unchecked")
43.   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
44.      private void initComponents() {
45.
46.      spinner1 = new com.mindprod.spinner.Spinner();
47.      spinner2 = new com.mindprod.spinner.Spinner();
48.      spinner3 = new com.mindprod.spinner.Spinner();
49.      spinner4 = new com.mindprod.spinner.Spinner();
50.      jLabel1 = new javax.swing.JLabel();
51.      jTextField_dbaddress = new javax.swing.JTextField();
52.      jToggleButton_dbconnect = new javax.swing.JToggleButton();
```

```
53.   jComboBox_db_collections = new javax.swing.JComboBox();
54.   jLabel2 = new javax.swing.JLabel();
55.   jButton_getCount = new javax.swing.JButton();
56.   jLabel_count = new javax.swing.JLabel();
57.   jSpinner_from = new javax.swing.JSpinner();
58.   jSpinner_to = new javax.swing.JSpinner();
59.   jLabel3 = new javax.swing.JLabel();
60.   jButton1 = new javax.swing.JButton();
61.   jLabel4 = new javax.swing.JLabel();
62.   jLabel_error = new javax.swing.JLabel();
63.   jComboBox_db_collectionspatient = new javax.swing.JComboBox();
64.   jLabel5 = new javax.swing.JLabel();
65.
66.   setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_
      ON_CLOSE);
67.
68.   jLabel1.setText("DB address");
69.
70.   jTextField_dbaddress.addActionListener(new java.awt.
      event.ActionListener() {
71.   public void actionPerformed(java.awt.event.Action
      Event evt) {
72.           jTextField_dbaddressActionPerformed(evt);
73.             }
74.         });
75.
76.         jToggleButton_dbconnect.setText("Connect");
77.   jToggleButton_dbconnect.addActionListener(new java.
      awt.event.ActionListener() {
78.   public void actionPerformed(java.awt.event.ActionEvent evt) {
79.     jToggleButton_dbconnectActionPerformed(evt);
80.             }
81.         });
82.
83.         jComboBox_db_collections.setMaximumRowCount(20);
84.         jComboBox_db_collections.setToolTipText("");
85.   jComboBox_db_collections.addActionListener(new java.
      awt.event.ActionListener() {
86.   public void actionPerformed(java.awt.event.ActionEvent evt) {
87.   jComboBox_db_collectionsActionPerformed(evt);
88.             }
89.         });
90.
91.         jLabel2.setText("Patient Name");
92.
93.         jButton_getCount.setText("Get Count");
94.         jButton_getCount.addActionListener(new java.awt.
           event.ActionListener() {
95.         public void actionPerformed(java.awt.event.Action
           Event evt) {
96.                 jButton_getCountActionPerformed(evt);
97.             }
98.         });
99.
100.  jSpinner_from.setModel(new javax.swing.SpinnerDateModel(new
      java.util.Date(), null, null, java.util.Calendar.SECOND));
101.
       jSpinner_from.setEditor(new javax.swing.JSpinner.DateEditor
```

```
      (jSpinner_from, "dd/MM/yyyy HH:mm:ss"));
102.     // Code adding the component to the parent container - not shown here
103.
104.  jSpinner_to.setModel(new javax.swing.SpinnerDateModel(new
      java.util.Date(), null, null, java.util.Calendar.SECOND));
105.  jSpinner_to.setEditor(new javax.swing.JSpinner.DateEditor
      (jSpinner_to, "dd/MM/yyyy HH:mm:ss"));
106.
107.  jLabel3.setText("From:");
108.
109.  jButton1.setText("Extract");
110.  jButton1.addActionListener(new java.awt.event.Action
      Listener() {
111.  public void actionPerformed(java.awt.event.ActionEvent evt) {
112.        jButton1ActionPerformed(evt);
113.        }
114.     });
115.
116.     jLabel4.setText("To:");
117.
118.  jComboBox_db_collectionspatient.setMaximumRowCount(20);
119.  jComboBox_db_collectionspatient.setToolTipText("");
120.  jComboBox_db_collectionspatient.addActionListener
      (new java.awt.event.ActionListener() {
121.  public void actionPerformed(java.awt.event.Action
      Event evt) {
122.  jComboBox_db_collectionspatientActionPerformed(evt);
123.              }
124.         });
125.
126.    jLabel5.setText("v1.1     10-10-13");
127.
128.    javax.swing.GroupLayout layout = new javax.swing.
        GroupLayout(getContentPane());
129.          getContentPane().setLayout(layout);
130.          layout.setHorizontalGroup(
131.  layout.createParallelGroup(javax.swing.GroupLayout.
      Alignment.LEADING)
132.      .addGroup(layout.createSequentialGroup()
133.  .addGroup(layout.createParallelGroup(javax.swing.
      GroupLayout.Alignment.LEADING)
134.  .addGroup(layout.createSequentialGroup()
135.  .addContainerGap()
136.  .addGroup(layout.createParallelGroup(javax.swing.
      GroupLayout.Alignment.LEADING)
137.  .addComponent(jLabel_error, javax.swing.GroupLayout.
DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

138.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      layout.createSequentialGroup()
139.  .addComponent(jComboBox_db_collections, 0, javax.swing.
      GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
140.  .addPreferredGap(javax.swing.Layout
      Style.ComponentPlacement.RELATED)
141.  .addComponent(jButton_getCount)
142.  .addGap(1, 1, 1))
143.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
      layout.createSequentialGroup()
```

```
  144.  .addComponent(jToggleButton_dbconnect, javax.swing.Group
Layout.PREFERRED_SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE)
  145.  .addPreferredGap(javax.swing.LayoutStyle.Component
        Placement.UNRELATED)
  146.  .addGroup(layout.createParallelGroup(javax.swing.
        GroupLayout.Alignment.LEADING)
  147.  .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_
      SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
  148.  .addGroup(layout.createSequentialGroup()
  149.  .addComponent(jLabel2)
  150.  .addGap(0, 0, Short.MAX_VALUE)))
  151.  .addPreferredGap(javax.swing.LayoutStyle.Component
        Placement.RELATED)
  152.  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING, false)
  153.  .addComponent(jTextField_dbaddress)
  154.  .addComponent(jComboBox_db_collectionspatient, 0, 346,
        Short.MAX_VALUE)))
  155.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        layout.createSequentialGroup()
  156.  .addGap(0, 137, Short.MAX_VALUE)
  157.  .addGroup(layout.createParallelGroup(javax.swing.Group
        Layout.Alignment.LEADING)
  158.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        layout.createSequentialGroup()
  159.  .addComponent(jLabel3)
  160.  .addGap(18, 18, 18)
  161.  .addComponent(jSpinner_from, javax.swing.GroupLayout.PREFERRED_
        SIZE, 207, javax.swing.GroupLayout.PREFERRED_SIZE))
  162.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        layout.createSequentialGroup()
  163.  .addComponent(jLabel4)
  164.  .addGap(18, 18, 18)
  165.  .addComponent(jSpinner_to, javax.swing.GroupLayout.PREFERRED_
        SIZE, 207, javax.swing.GroupLayout.PREFERRED_SIZE)))
  166.  .addGap(65, 65, 65)
  167.  .addComponent(jLabel_count, javax.swing.GroupLayout.PREFERRED_
        SIZE, 81, javax.swing.GroupLayout.PREFERRED_SIZE))))
  168.  .addGroup(layout.createSequentialGroup()
  169.  .addGap(227, 227, 227)
  170.  .addComponent(jButton1)
  171.  .addGap(0, 0, Short.MAX_VALUE))
  172.  .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        layout.createSequentialGroup()
  173.  .addGap(0, 0, Short.MAX_VALUE)
  174.  .addComponent(jLabel5)))
  175.  .addContainerGap())
  176.          );
  177.  layout.setVerticalGroup(
  178.  layout.createParallelGroup(javax.swing.GroupLayout.
        Alignment.LEADING)
  179.  .addGroup(layout.createSequentialGroup()
  180.  .addContainerGap()
  181.  .addGroup(layout.createParallelGroup(javax.swing.Group
        Layout.Alignment.BASELINE)
  182.  .addComponent(jTextField_dbaddress, javax.swing.GroupLayout.
        PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
183.      .addComponent(jLabel1)
184.        .addComponent(jToggleButton_dbconnect))
185.    .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.RELATED)
186.    .addGroup(layout.createParallelGroup(javax.swing.
      GroupLayout.Alignment.BASELINE)
187.  .addComponent(jComboBox_db_collectionspatient, javax
      .swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.
      DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
188.  .addComponent(jLabel2))
189.  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement
      .UNRELATED)
190.  .addGroup(layout.createParallelGroup(javax.swing.Group
      Layout.Alignment.BASELINE)
191.  .addComponent(jComboBox_db_collections, javax.swing.Group
      Layout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE)
192.  .addComponent(jButton_getCount))
193.  .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.UNRELATED)
194.  .addGroup(layout.createParallelGroup(javax.swing
      .GroupLayout.Alignment.LEADING)
195.  .addComponent(jLabel_count, javax.swing.GroupLayout
    .PREFERRED_SIZE, 21, javax.swing.GroupLayout.PREFERRED_SIZE)
196.  .addGroup(layout.createSequentialGroup()
197.  .addGroup(layout.createParallelGroup(javax.swing
      .GroupLayout.Alignment.BASELINE)
198.  .addComponent(jLabel3)
199.  .addComponent(jSpinner_from, javax.swing.GroupLayout
      .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
      javax.swing.GroupLayout.PREFERRED_SIZE))
200.  .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.RELATED)
201.  .addGroup(layout.createParallelGroup(javax.swing.Group
      Layout.Alignment.BASELINE)
202.  .addComponent(jSpinner_to, javax.swing.GroupLayout
      .PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_
      SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
203.  .addComponent(jLabel4))))
204.  .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.UNRELATED)
205.  .addComponent(jButton1)
206.  .addGap(13, 13, 13)
207.  .addComponent(jLabel_error, javax.swing.GroupLayout
    .PREFERRED_SIZE, 35, javax.swing.GroupLayout.PREFERRED_SIZE)
208.  .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.RELATED, 63, Short.MAX_VALUE)
209.                  .addComponent(jLabel5))
210.          );
211.
212.          pack();
213.      } // </editor-fold>//GEN-END:initComponents
214.
215.   private void jTextField_dbaddressActionPerformed(java.awt
    .event.ActionEvent evt) {//GEN-FIRST:event_jTextField_dbaddressActionPerformed
216.            } //GEN-LAST:event_jTextField_dbaddressActionPerformed
217.
218.  private void jToggleButton_dbconnectActionPerformed(java.awt
```

```
  .event.ActionEvent evt) {//GEN-FIRST:event_jToggleButton_dbconnectActionPerformed
219.
220.      if ("".equals(jTextField_dbaddress.getText()))
221.      {
222.          System.out.println("Empty field");
223.          return;
224.      }
225.      try
226.      {
227.          String user = "winmed";
228.          String passwd = "winmed";
229.          MongoOptions mo = new  MongoOptions();
230.          mo.setConnectTimeout(50000);
231.  Mongo m = new Mongo( jTextField_dbaddress.getText());
232.        db = m.getDB( "winmed_db" );
233.  boolean auth = db.authenticate(user,passwd.toCharArray());
234.
235.          DBCollection coll;
236.          DBCursor cur;
237.          coll = db.getCollection("winal.patients");
238.  BasicDBObject fields = new BasicDBObject("name",true)
     .append("surname",true);
239.          BasicDBObject query = new BasicDBObject();
240.          cur = coll.find(query, fields);
241.          String patient;
242.          while (cur.hasNext())
243.          {
244.              patient = cur.next().get("name").toString();
245.  patient = patient+" "+ cur.curr().get("surname").toString();
246.    jComboBox_db_collectionspatient.addItem(patient);
247.          }
248.      }
249.      catch (UnknownHostException | MongoException e)
250.      {
251.          System.out.println("Problema connessione con DB");
252.      }
253.      catch (java.io.IOException e )
254.      {
255.          System.exit(0);
256.      }
257.      } //GEN-LAST:event_jToggleButton_dbconnectActionPerformed
258.
259.  private void jComboBox_db_collectionsActionPerformed(java.awt
.event.ActionEvent evt) { //GEN-FIRST:event_jComboBox_db_collectionsActionPerformed
260.
261.      } //GEN-LAST:event_jComboBox_db_collectionsActionPerformed
262.
263.  private void jButton_getCountActionPerformed(java.awt
  .event.ActionEvent evt) { //GEN-FIRST:event_jButton_getCountActionPerformed
264.  DBCollection coll = db.getCollection(jComboBox_db
     _collections.getSelectedItem().toString());
265.  jLabel_count.setText(String.valueOf(coll.getCount()));
266.      } //GEN-LAST:event_jButton_getCountActionPerformed
267.
268.  private void jButton1ActionPerformed(java.awt.event.Action
    Event evt) { //GEN-FIRST:event_jButton1ActionPerformed
269.          jLabel_error.removeAll();
270.          long dateunix_temp = 0;
```

```
271.       double v_temp = 0.0;
272.
273.  Object date_from = jSpinner_from.getNextValue().toString();
274.  String date_to = jSpinner_to.getNextValue().toString();
275.          DBCollection coll;
276.          DBCursor cur;
277.          BasicDBObject query = new BasicDBObject();
278.          boolean errorextractall = false;
279.          BasicDBObject query_pat = new BasicDBObject();
280.          BasicDBObject fields = new BasicDBObject();
281.
282.          String wpserial = null;
283.  String patient = jComboBox_db_collectionspatient.getSelected
    Item().toString();
284.  String app = jComboBox_db_collections.getSelectedItem()
     .toString();
285.      String nameclear = patient.replace(" ","");
286.      File dir = new File("./Test/"+patient);
287.        dir.mkdirs();
288.           // query che seleziona i record solo in base agli intervalli introdotti nella form
289.  query.put("t", BasicDBObjectBuilder.start("$gte", jSpinner
  _from.getValue()).add("$lte", jSpinner_to.getValue()).get());
290.
291.     // table contiene la stringa del singolo modulo di cui si vogliono estrarre i dati
292.  String table = jComboBox_db_collections.getSelectedItem()
     .toString();
293.
294.  Date datefromfile = (Date) jSpinner_from.getValue();
295.  String dateform =(DateFormat.getDateTimeInstance(DateFormat
     .SHORT, DateFormat.LONG).format(datefromfile));
296.
297.      Date datetofile = (Date) jSpinner_to.getValue();
298.          String dateformto =(DateFormat.getDateTimeInstance
    (DateFormat.SHORT, DateFormat.LONG).format(datetofile));
299.
300.      // se si è scelto di estratte tutto allora jCheckBox è selezionato e il modulo da cui estrarre i dati
301.      // verrà scelto solo successivamente
302.
303.    if (app.endsWith("temperatures")||app.endsWith("AllItem"))
304.          {
305.  String app2 = app.replaceAll("AllItem", "temperatures");
306.      // viene creata la collezione, diversa per ogni scelta di table
307.      coll = db.getCollection(app2);
308.
309.       // fields varierà per ogni scelta di table a seconda dell'organizzazione data al db
310.       // in questo caso i valori che ci interessano sono v (valore di temperatura) e t (timestamp)
311.  fields = new BasicDBObject("v",true).append("t",true)
     .append("_id",false);
312.
313.       // cursore alla collezione ("t",1) ordinamento crescente
314.  cur = coll.find(query,fields).sort(new BasicDBObject("t", 1));
315.
316.         NumberFormat nf = NumberFormat.getInstance();
317.         nf.setMaximumFractionDigits(3);
318.       nf.setMinimumFractionDigits(3);
319.
320.         if(cur.hasNext())
321.             {
```

```
322.          double v_err = 0, t_err = 0;
323.           double tot_smpl = (double) cur.count();
324.           double v;
325.          boolean first = true;
326.    File file = new File(dir, ""+dateform.substring(0,2)
      +"-"+dateform.substring(3,5)+"-"+dateform.substring(6,8)+
327.  "-"+dateform.substring(9,11)+":"+dateform.substring(12,14)
        +":"+dateform.substring(15,17)+
328.  "_"+dateformto.substring(0,2)+"- "+dateformto
      .substring(3,5)+"-"+dateformto.substring(6,8)+
329.  "-"+dateformto.substring(9,11)+":"+dateformto.substring
     (12,14)+":"+dateformto.substring(15,17)+
330.  "_"+nameclear+"_temperatures.csv");
331.        try {
332.  FileWriter newJsp = new FileWriter(file);
333.      } catch (IOException ex) {
334.  Logger.getLogger(NewJFrame.class.getName()).log(Level
      .SEVERE, null, ex);
335.        }
336.
337.       try
338.                   {
339.      try (FileWriter writer = new FileWriter(file))
340.                       {
341.    writer.append("v | gap_v | gap_t | dateunix \n");
342.        while(cur.hasNext())
343.                            {
344.  //System.out.println(cur.next().get("v").toString());
345.  // per ogni valore del cursore convertiamo il valore di temperatura in formato stringa in double
346.
v = Double.parseDouble((cur.next().get("v").toString()));
347.  //double v = (double) (cur.next().get("v"));
348.
349.  // leggiamo il tempo dal db per ogni record e, considerando il fuso orario aggiungiamo 2 ore
350.      Date date = (Date) (cur.curr().get("t"));
351.  // le due ore hanno un ordine di grandezza maggiore perchè consideriamo un unix time con millisecondi
352.  long dateunix = date.getTime() + 7200000;
353.      if (first)
354.      {
355.        dateunix_temp = dateunix - 5000;
356.             v_temp = v + 0.1;
357.        first = false;
358.      }
359.
360.     // per ogni modulo dovremo controllare il giusto gap tra un record e il successivo
361.      // che dipenderà dal tempo di campionamento del modulo in questione
362.     long gap_t = dateunix - dateunix_temp;
363.
364.     if ((gap_t > 5100)||(gap_t < 4900))
365.                                 {
366.     // contiamo gli errori per ottenere una percentuale di errore
367.           t_err += 1;
368.        errorextractall = true;
369.            }
370.        dateunix_temp = dateunix;
371.          double gap_v = Math.abs((v - v_temp)*1000);
372.
373.  if ((((v == 1) && (v_temp == 44.9)) || ((v == 0)
```

```
  && (v_temp == 1)) || ((v == 25.1) && (v_temp == 0))) == true)
374.       {
375.  gap_v = 100;
376.       }
377.
378.      v_temp = v;
379.
          // il controllo degli errori viene effettuato anche sul valore di temperatura e il precedente,
380.      // il valore assoluto tra due valori non deve superare 101 (nel caso di temperatura)
381.
382.  if ((gap_v > 101)||(gap_v < 99))
383.      {
384.        v_err += 1;
385.      errorextractall = true;
386.    }
387.      // scriviamo in una label nascosta nella form i risultati ottenuti in termini di errore
388.      // gli stessi valori verranno in seguito scritto in fondo ad ogni file di report creato
389.
390.  writer.append(v+" , "+nf.format(gap_v) +" , "+gap_t+" ,
       "+dateunix+" \n");
391.
392.       }
393.       writer.flush();
394.       double date_err = t_err/tot_smpl;
395.         double value_err = v_err/tot_smpl;
396.       if (app.endsWith("AllItem")== false)
397.      jLabel_error.setText("No errors found");
398.          else
399.                    {
400.      if (errorextractall == true)
401.    jLabel_error.setText("Found errors");
402.                         }
403.  writer.append("REPORT for TEMPERATURE\r\n"
404.  + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
errors #: "+t_err+"\r\nTimestamp Errors perc: "+date_err+"% \r\n"
405.  + "Values extracted: "+tot_smpl+"\r\nValue Errors #: "+v_err+"
      \r\nValue Errors perc: "+value_err+"%");
406.
407.          }
408.     }
409.       catch(IOException e)
410.      {
411.         e.printStackTrace();
412.            }
413.      }
414.          }
415.
416.       if (app.endsWith("ecgs")||app.endsWith("AllItem"))
417.         {
418.  String app2 = app.replaceAll("AllItem", "ecgs");
419.
420.  coll = db.getCollection(app2);
421.  // come per la temperatura interessano valore puntuale e time
422.  fields = new BasicDBObject("v",true).append("t",true).append
      ("_id",false);
423.  cur = coll.find(query, fields).sort(new BasicDBObject("t", 1));
424.  if (cur.hasNext())
425.             {
```

```
426.  File file = new File(dir, ""+dateform.substring(0,2)+"-"
    +dateform.substring(3,5)+"-"+dateform.substring(6,8)+
427.  "-"+dateform.substring(9,11)+":"+dateform.substring(12,14)
    +":"+dateform.substring(15,17)+
428.  "_"+dateformto.substring(0,2)+"-"+dateformto.substring(3,5)
     +"-"+dateformto.substring(6,8)+
429.    "-"+dateformto.substring(9,11)+":"+dateformto.substring(12,14)
     +":"+dateformto.substring(15,17)+
430.          "_"+nameclear+"_ecgs.csv");
431.       try {
432.                   FileWriter newJsp = new FileWriter(file);
433.         } catch (IOException ex) {
434.    Logger.getLogger(NewJFrame.class.getName()).log(Level
        .SEVERE, null, ex);
435.         }
436.
437.       try
438.     {
439.     try (FileWriter writer = new FileWriter(file))
440.           {
441.           double heartsamplecheck;
442.           double heartsample;
443.           double heartsampleprev = 1;
444.           boolean first = true;
445.           Date date;
446.           long dateu;
447.           long dateprev = 0;
448.           boolean firsttime = false;
449.           double v_err = 0, t_err = 0,s_err = 0;
450.           double tot_smpl = (double) cur.count();
451.           Iterator a;
452.           ArrayList x = null;
453.
454.           writer.append("heartsample | dateunix \n");
455.
456.           while(cur.hasNext())
457.                          {
458.  // ogni record contiene un array di valori (500 al secondo, in accordo con il protocollo)
459.         x = (ArrayList)cur.next().get("v");
460.         date = (Date) (cur.curr().get("t"));;
461.         long dateunix = date.getTime() + 7200000;
462.         dateprev = dateunix - 1000;
463.         if (x.size()!=500)
464.                 {
465.                 s_err += 1;
466.         errorextractall = true;
467.           }
468.
469.         a = x.iterator();
470.       boolean datestamp = true;
471.
472.    // per ogni valore del cursore dobbiamo fare un ciclo per ogni valore dell'array
473.        while(a.hasNext())
474.                 {
475.     // convertiamo ognuno dei 500 valori in double
476.    heartsample = Double.parseDouble(a.next().toString());
477.
478.    if (datestamp)
```

```
479.            {
480.          // la data verrà scritta nella prima riga dei 500 campioni
481.       writer.append(heartsample+" , "+dateunix+"\n");
482.         datestamp = false;
483.
484.         if (dateunix - dateprev != 1000)
485.                   {
486.         t_err += 1;
487.         errorextractall = true;
488.                   }
489.                   }
490.
491.         else
492.       writer.append(heartsample+"\n");
493.
494.         if (first)
495.                   {
496.       heartsampleprev = heartsample;
497.       first = false;
498.             continue;
499.          }
500.
501.         heartsamplecheck = heartsample - heartsampleprev;
502.         heartsampleprev = heartsample;
503.
504.        // il valore tra due valori successivi non può scostarsi per più di 0.001 (rampa)
505.       if (Math.abs(heartsamplecheck) > 0.001)
506.                   {
507.                   v_err += 1;
508.             errorextractall = true;
509.                   }
510.
511.                   }
512.                   dateprev=dateunix;
513.                   }
514.                   writer.flush();
515.             double date_err = t_err/tot_smpl;
516.             double value_err = v_err/tot_smpl;
517.             double sample_err = s_err/(cur.size()*x.size());
518.
519.           if (app.endsWith("AllItem")==false)
520.  jLabel_error.setText("No errors found");
521.         else
522.             if (errorextractall == true)
523.  jLabel_error.setText("Found errors");
524.  writer.append("REPORT for ECGs \r\n"
525. + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
      errors #: "+t_err+"\r\nTimestamp Errors perc:
      "+date_err+"% \r\n"
526.  + "Hearth Rate values extracted: "+tot_smpl+"\r\nHearth
      Rate vaules Errors#: "+v_err+"\r\nHearth Rate values
    Errors perc: "+value_err+"% \r\n"
527.  + "ECG Sample values extracted: "+(cur.size()*x.size())
      +"\r\nECG numSample Errors #: "+s_err+"\r\n ECG numSample
     Errors perc: "+sample_err+"%");
528.                   }
529.    }
530.
```

```
531.             catch(IOException e)
532.                   {
533.                       e.printStackTrace();
534.                   }
535.           }
536.          }
537.
538.  if (app.endsWith("ecg5s")||app.endsWith("AllItem"))
539.          {
540.  String app2 = app.replaceAll("AllItem", "ecg5s");
541.
542.      coll = db.getCollection(app2);
543.     // pl,l1,l2 saranno tre array, ognuno da 250 valori, in accordo con il protocollo
544.  fields = new BasicDBObject("pl",true).append("l1",true)
    .append("l2",true).append("t",true).append("_id",false);
545.  cur = coll.find(query, fields).sort(new BasicDBObject("t", 1));
546.
547.    NumberFormat nf = NumberFormat.getInstance();
548.    nf.setMaximumFractionDigits(3);
549.      nf.setMinimumFractionDigits(3);
550.
551.    if(cur.hasNext())
552.    {
553.    File file = new File(dir, ""+dateform.substring(0,2)+"-
    "+dateform.substring(3,5)+"-"+dateform.substring(6,8)+
554.  "-"+dateform.substring(9,11)+":"+dateform.substring(12,14
      +":"+dateform.substring(15,17)+
555.  "_"+dateformto.substring(0,2)+"-"+dateformto.substring(3,5
        +"-"+dateformto.substring(6,8)+
556.  "-"+dateformto.substring(9,11)+":"+dateformto
      .substring(12,14)+":"+dateformto.substring(15,17)+
557.    "_"+nameclear+"_ecg5s.csv");
558.    try {
559.  FileWriter newJsp = new FileWriter(file);
560.    } catch (IOException ex) {
561.    Logger.getLogger(NewJFrame.class.getName()).log(Level.
      SEVERE, null, ex);
562.           }
563.
564.           try
565.           {
566.  try (FileWriter writer = new FileWriter(file))
567.           {
568.         double pl_sample;
569.         double l1_sample;
570.         double l2_sample;
571.
572.         double pl_samplecheck;
573.         double l1_samplecheck;
574.         double l2_samplecheck;
575.
576.         double pl_sampleprev = 1;
577.         double l1_sampleprev = 1;
578.         double l2_sampleprev = 1;
579.
580.         boolean first = true;
581.         Date date;
582.         long dateu;
```

```
583.            long dateprev = 0;
584.            boolean firsttime = false;
585.
586.        double v_err = 0, t_err = 0, s_err = 0, pl_v_err = 0,
     l1_v_err = 0, l2_v_err = 0;
587.      double tot_smpl = (double) cur.count();
588.
589.         ArrayList pl = null;
590.         Iterator i_pl;
591.
592.                  writer.append(
593.        "pl_sample | l1_sample | l2_sample | dateunix \n");
594.
595.            while(cur.hasNext())
596.                        {
597.         pl = (ArrayList)cur.next().get("pl");
598.         ArrayList l1 = (ArrayList)cur.curr().get("l1");
599.         ArrayList l2 = (ArrayList)cur.curr().get("l2");
600.
601.         date = (Date) (cur.curr().get("t"));
602.         long dateunix = date.getTime() + 7200000;
603.                              dateprev = dateunix - 500;
604.    if ((pl.size()!=250)||(l1.size()!=250)||(l2.size()!=250))
605.                  {
606.        s_err += 1;
607.          errorextractall = true;
608.            }
609.
610.            i_pl = pl.iterator();
611.            Iterator i_l1 = l1.iterator();
612.            Iterator i_l2 = l2.iterator();
613.
614.            boolean datestamp = true;
615.
616.    while(i_pl.hasNext() && i_l1.hasNext() && i_l2.hasNext())
617.                               {
618.  pl_sample = Double.parseDouble(i_pl.next().toString());
619.  l1_sample = Double.parseDouble(i_l1.next().toString());
620.  l2_sample = Double.parseDouble(i_l2.next().toString());
621.
622.  if (datestamp)
623.                  {
624.  writer.append(nf.format(pl_sample)+" , "+nf.format(l1
     _sample)+" , "+nf.format(l2_sample)+" , "+dateunix+"\n");
625. datestamp = false;
626.  if (dateunix - dateprev != 500)
627.            {
628.       t_err += 1;
629.  errorextractall = true;
630.     }
631.     }
632.
633.    else
634.    writer.append(nf.format(pl_sample)+" , "+nf.format
       (l1_sample)+" , "+nf.format(l2_sample)+"\n");
635.
636.        if (first)
637.        {
```

```
638.    pl_sampleprev = pl_sample;
639.    l1_sampleprev = l1_sample;
640.    l2_sampleprev = l2_sample;
641.    first = false;
642.    continue;
643.         }
644.
645.    pl_samplecheck = pl_sample - pl_sampleprev;
646.    pl_sampleprev = pl_sample;
647.
648.    l1_samplecheck = l1_sample - l1_sampleprev;
649.    l1_sampleprev = l1_sample;
650.
651.    l2_samplecheck = l2_sample - l2_sampleprev;
652.    l2_sampleprev = l2_sample;
653.
654.  // il valore assoluto tra due valori successivi non può superare 0.01
655.  //  if ((Math.abs(pl_samplecheck) > 0.01)&&(Math.abs(l2_samplecheck) > 0.01)&&(Math.abs(l1
        _samplecheck) > 0.01))
656.    //            {
657.    //               v_err += 1;
658.    //               errorextractall = true;
659.    //            }
660.   // il valore assoluto tra due valori successivi non può superare 0.01
661.    if (Math.abs(pl_samplecheck) > 0.01)
662.             {
663.          pl_v_err += 1;
664.      errorextractall = true;
665.        }
666.    if (Math.abs(l2_samplecheck) > 0.01)
667.      {
668.        l2_v_err += 1;
669.        errorextractall = true;
670.                                    }
671.    if (Math.abs(l1_samplecheck) > 0.01)
672.                                     {
673.        l1_v_err += 1;
674.    errorextractall = true;
675.                         }
676.                         }
677.          dateprev=dateunix;
678.                  }
679.          writer.flush();
680.          double date_err = t_err/tot_smpl;
681.  //double value_err = v_err/tot_smpl;
682.          double pl_value_err = pl_v_err/tot_smpl;
683.          double l1_value_err = l1_v_err/tot_smpl;
684.          double l2_value_err = l2_v_err/tot_smpl;
685.          double sample_err = s_err/(cur.size()*pl.size());
686.
687.        if (app.endsWith("AllItem")==false)
688.          jLabel_error.setText("No errors found");
689.        else
690.        if (errorextractall == true)
691.          jLabel_error.setText("Found errors");
692.        writer.append("REPORT for ECG5 \r\n"
693.   + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp errors
    #: "+t_err+"\r\nTimestamp Errors perc: "+date_err+"% \r\n"
```

```
694.    + "Packets extracted (L1,L2,PL): "+tot_smpl+"\r\n"
695.    + "PL Errors #: "+pl_v_err+"\r\nPL values Errors
      perc: "+pl_value_err+"% \r\n"
696.    + "L1 Errors #: "+l1_v_err+"\r\nL1 values Errors
        perc: "+l1_value_err+"% \r\n"
697.  + "L2 Errors #: "+l2_v_err+"\r\nL2 values Errors
        perc: "+l2_value_err+"% \r\n"
698.      + "ECG Sample values extracted (L1|L2|PL): "+(cur.size()
      *pl.size())+"\r\nECG numSample Errors #: "+s_err+"\r\n
    ECG numSample Errors perc: "+sample_err+"%");
699.                       }
700.                   }
701.
702.          catch(IOException e)
703.                   {
704.              e.printStackTrace();
705.                   }
706.           }
707.          }
708.
709.      if (app.endsWith("positions")||app.endsWith("AllItem"))
710.          {
711.      String app2 = app.replaceAll("AllItem", "positions");
712.
713.      coll = db.getCollection(app2);
714.  fields = new BasicDBObject("av",true).append
      ("p",true).append("t",true).append("_id",false);
715.  cur = coll.find(query, fields).sort(new BasicDBObject
      ("t", 1));
716.      if (cur.hasNext())
717.              {
718.    File file = new File(dir, ""+dateform.substring(0,2)+"-"
    +dateform.substring(3,5)+"-"+dateform.substring(6,8)+
719.  "-"+dateform.substring(9,11)+":"+dateform.substring(12,14)+":"
      +dateform.substring(15,17)+
720.    "_"+dateformto.substring(0,2)+"-"+dateformto.substring(3,5)
      +"-"+dateformto.substring(6,8)+
721.    "-"+dateformto.substring(9,11)+":"+dateformto.substring
      (12,14)+":"+dateformto.substring(15,17)+
722.  "_"+nameclear+"_positions.csv");
723.        try {
724.  FileWriter newJsp = new FileWriter(file);
725.  } catch (IOException ex) {
726.  Logger.getLogger(NewJFrame.class.getName())
      .log(Level.SEVERE, null, ex);
727.         }
728.
729.       try
730.                       {
731.       try (FileWriter writer = new FileWriter(file))
732.           {
733.           int p_sample;
734.           double av_sample;
735.
736.           int p_samplecheck;
737.           double av_samplecheck;
738.
739.           int p_sampleprev = 1;
```

```
740.            double av_sampleprev = 1;
741.
742.            boolean first = true;
743.            Date date;
744.            long dateu;
745.            long dateprev = 0;
746.            boolean firsttime = false;
747.
748.            double v_err = 0, t_err = 0,s_err = 0;
749.            double tot_smpl = (double) cur.count();
750.
751.            ArrayList p = null;
752.            Iterator i_p;
753.
754.            writer.append(
755.          "p_sample | x | y | z | dateunix \n");
756.
757.            while(cur.hasNext())
758.                              {
759.            p = (ArrayList)cur.next().get("p");
760.          ArrayList av = (ArrayList)cur.curr().get("av");
761.
762.          date = (Date) (cur.curr().get("t"));
763.          long dateunix = date.getTime() + 7200000;
764.            dateprev = dateunix - 1000;
765.          if ((p.size()!=1)||(av.size()!=20))
766.                   {
767.            s_err += 1;
768.                   errorextractall = true;
769.                        }
770.
771.          i_p = p.iterator();
772.        Iterator i_av = av.iterator();
773.
774.          ArrayList xyz;
775.          Iterator i_xyz;
776.
777.          boolean datestamp = true;
778.
779.          while(i_p.hasNext() && i_av.hasNext())
780.                                  {
781.        p_sample = Integer.parseInt(i_p.next().toString());
782.        xyz = (ArrayList)i_av.next();
783.        i_xyz = xyz.iterator();
784.        if (datestamp)
786.                         {
787.  writer.append(p_sample+" , "+Double.parseDouble(i_xyz
      .next().toString())+" , "+Double.parseDouble(i_xyz.next()
      .toString())+" , "+Double.parseDouble(i_xyz.next()
      .toString())+" , "+dateunix+"\n");
788.        datestamp = false;
789.        if (dateunix - dateprev != 1000)
790.        {
791.             t_err += 1;
792.          errorextractall = true;
793.            }
794.      }
795.
```

```
796.            else
797.  writer.append(p_sample+" , "+Double.parseDouble(i_xyz.next
    ().toString())+" , "+Double.parseDouble(i_xyz.next().toString
    ())+" , "+Double.parseDouble(i_xyz.next().toString())+"\n");
798.
799.          if (first)
800.                  {
801.          p_sampleprev = p_sample;
802.          first = false;
803.          continue;
804.             }
805.
806.        p_samplecheck = p_sample - p_sampleprev;
807.        p_sampleprev = p_sample;
808.
809.        if (((p_sample == 1 && p_sampleprev == 3)) ||
810.        ((p_sample == 3 && p_sampleprev == 1)))
811.                  {
812.        if (Math.abs(p_samplecheck) > 2)
813.                  {
814.        v_err += 1;
815.        errorextractall = true;
816.                        }
817.                  }
818.
819.        else if (Math.abs(p_samplecheck) > 1)
820.            {
821.          v_err += 1;
822.                  errorextractall = true;
823.                  }
824.             }
825.
826.                  dateprev=dateunix;
827.                  }
828.           writer.flush();
829.          double date_err = t_err/tot_smpl;
830.          double value_err = v_err/tot_smpl;
831.          double sample_err = s_err/(cur.size()*p.size());
832.
833.
834.
835.
836.          if (app.endsWith("AllItem")==false)
837.        jLabel_error.setText("No errors found");
838.          else
839.          if (errorextractall == true)
840.        jLabel_error.setText("Found errors");
841.        writer.append("REPORT for POSITION\r\n"
842.  + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
     errors #: "+t_err+"\r\nTimestamp Errors perc:
     "+date_err+"% \r\n"
843.  + "(X,Y,Z) Values extracted: "+tot_smpl+"\r\n(X,Y,Z)
     Value Errors #: "+v_err+"\r\n(X,Y,Z) Value Errors perc:
     "+value_err+"%\r\n"
844.  + "Position values extracted : "+(cur.size()*p.size())
     +"\r\nPosition values Errors #: "+s_err+"\r\nPosition
     values Errors perc: "+sample_err+"%");
845.
```

```
846.                            }
847.                        }
848.             catch(IOException e)
849.                        {
850.                            e.printStackTrace();
851.                        }
852.           }
853.          }
854.
855.    if (app.endsWith("oximeters")||app.endsWith("AllItem"))
856.          {
857.  String app2 = app.replaceAll("AllItem", "oximeters");
858.
859.              coll = db.getCollection(app2);
860.              fields = new BasicDBObject(
861.                   "v",true)
862.                       .append("oxs",true)
863.                       .append("pr",true)
864.                       .append("t",true)
865.                       .append("_id",false);
866.  cur = coll.find(query, fields).sort(new BasicDBObject("t", 1));
867.
868.       if (cur.hasNext()) {
869.      File file = new File(dir, ""+dateform.substring(0,2)+"-
"+dateform.substring(3,5)+"-"+dateform.substring(6,8)+
870.  "-"+dateform.substring(9,11)+":"+dateform.substring
      (12,14)+":"+dateform.substring(15,17)+
871.  "_"+dateformto.substring(0,2)+"-"+dateformto.substring(3,5)+"-
"+dateformto.substring(6,8)+
872.  "-"+dateformto.substring(9,11)+":"+dateformto.substring
      (12,14)+":"+dateformto.substring(15,17)+
873.  "_"+nameclear+"_oximeters.csv");
874.       try {
875.      FileWriter newJsp = new FileWriter(file);
876.      } catch (IOException ex) {
877.  Logger.getLogger(NewJFrame.class.getName()).log
      (Level.SEVERE, null, ex);
878.                  }
879.
880.       try
881.       {
882.       try (FileWriter writer = new FileWriter(file))
883.                       {
884.            double v_samplecheck;
885.            double v_sample;
886.            double v_sampleprev = 1;
887.
888.            int oxs_samplecheck = 0;
889.            int oxs_sample = 0;
890.            int oxs_sampleprev = 1;
891.
892.            int pr_samplecheck = 0;
893.            int pr_sample = 0;
894.            int pr_sampleprev = 1;
895.
896.            boolean first = true;
897.            Date date;
898.            long dateu;
```

```
899.            long dateprev = 0;
900.            boolean firsttime = false;
901.
902.            double v_err = 0,
903.                  t_err = 0,
904.                  s_err = 0,
905.                  vosx_err = 0,
906.                  vpr_err = 0;
907.            double tot_smpl = cur.count();
908.            ArrayList v = null;
909.            Iterator i_v;
910.
911.            writer.append(
912.          "v_sample | oxs_sample | pr_sample | dateunix \n");
913.
914.           while(cur.hasNext())
915.                           {
916.          v = (ArrayList)cur.next().get("v");
917.          date = (Date) (cur.curr().get("t"));
918.  oxs_sample = Integer.parseInt(cur.curr().get("oxs").toString());
919.  pr_sample = Integer.parseInt(cur.curr().get("pr").toString());
920.
921.           long dateunix = date.getTime() + 7200000;
922.           dateprev = dateunix - 1000;
923.           if (v.size()!=60)
924.                          {
925.                  s_err += 1;
926.              errorextractall = true;
927.                  }
928.
929.              i_v = v.iterator();
930.              boolean datestamp = true;
931.
932.              while(i_v.hasNext())
933.                                {
934.    v_sample = Double.parseDouble(i_v.next().toString());
935.
936.       if (datestamp)
937.                  {
938.  writer.append(v_sample+" , "+oxs_sample+" , "+pr_sample+" ,
       "+dateunix+"\n");
939.       datestamp = false;
940.
941.       if (dateunix - dateprev != 1000)
942.                  {
943.                  t_err += 1;
944.       errorextractall = true;
945.         }
946.       }
947.
948.       else
949.  writer.append(v_sample+" , "+oxs_sample+" , "+pr_sample+"\n");
950.
951.       if (first)
952.         {
953.              v_sampleprev = v_sample;
954.              oxs_sampleprev = oxs_sample;
955.              pr_sampleprev = pr_sample;
```

```
956.            first = false;
957.            continue;
958.                  }
959.
960.            v_samplecheck = v_sample - v_sampleprev;
961.            v_sampleprev = v_sample;
962.
963.            if (Math.abs(v_samplecheck) > 1)
964.                        {
965.                        v_err += 1;
966.            errorextractall = true;
967.                        }
968.                        }
969.          oxs_samplecheck = oxs_sample - oxs_sampleprev;
970.          pr_samplecheck = pr_sample - pr_sampleprev;
971.
972.    if ((((oxs_sample == 127) && (oxs_sampleprev == 99))
  || ((oxs_sample == 99) && (oxs_sampleprev == 127))) == false)
973.                  {
974.    if (Math.abs(oxs_samplecheck) > 1)
975.            {
976.             vosx_err += 1;
977.    errorextractall = true;
978.
979.                  }
980.                  }
981.
982.    if (Math.abs(pr_samplecheck) > 1)      983.
                        {
984.    vpr_err += 1;
985.    errorextractall = true;
986.
987.      }
988.
989.      oxs_sampleprev = oxs_sample;
990.      pr_sampleprev = pr_sample;
991.      dateprev=dateunix;
992.                  }
993.        writer.flush();
994.        double date_err = t_err/tot_smpl;
995.        double value_err = v_err/tot_smpl;
996.        double pulserate_err = vpr_err/tot_smpl;
997.        double ox_err = vosx_err/tot_smpl;
998.        double sample_err = s_err/(cur.size()*v.size());
999.        if (app.endsWith("AllItem")==false)
1000.                  jLabel_error.setText("No errors found");
1001.           else
1002.       if (errorextractall == true)
1003.       jLabel_error.setText("Found errors");
1004.
1005.       writer.append("REPORT for OXIMETER\r\n"
1006. + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
errors #: "+t_err+"\r\nTimestamp Errors perc: "+date_err+"% \r\n"
1007. + "V Values extracted: "+tot_smpl+"\r\nV Values Errors
  #: "+v_err+"\r\nV Values Errors perc: "+value_err+"%\r\n"
1008. + "OSX values extracted : "+tot_smpl+"\r\nOSX values
Errors #: "+vosx_err+"\r\nOSX values Errors perc: "+ox_err+"%\r\n"
1009. + "PR values extracted: "+tot_smpl+"\r\nPR Values
```

```
      Errors #: "+vpr_err+"\r\nPR Values Errors perc:
    "+pulserate_err+"%\r\n"
1010. + "Samples values extracted: "+(cur.size()*v.size())
       +"\r\nPR Values Errors #: "+s_err+"\r\nPR Values Errors
      perc: "+sample_err+"%\r\n");
1011.           }
1012.     }
1013.
1014.     catch(IOException e)
1015.           {
1016.         e.printStackTrace();
1017.           }
1018.         }
1019.         }
1020.
1021.   if (app.endsWith("tonometers")||app.endsWith("AllItem"))
1022.         {
1023.     String app2 = app.replaceAll("AllItem", "tonometers");
1024.
1025.     coll = db.getCollection(app2);
1026.     fields = new BasicDBObject("ton",true)
1027.           .append("t",true)
1028.           .append("sp",true)
1029.           .append("dp",true)
1030.           .append("mp",true)
1031.           .append("pr",true)
1032.           .append("_id",false);
1033. cur = coll.find(query, fields).sort(new BasicDBObject("t", 1));
1034.           if (cur.hasNext()) {
1035.
1036.   File file = new File(dir, ""+dateform.substring(0,2)+
1037.           "-"+dateform.substring(3,5)+
1038.           "-"+dateform.substring(6,8)+
1039.           "-"+dateform.substring(9,11)+
1040.           ":"+dateform.substring(12,14)+
1041.           ":"+dateform.substring(15,17)+
1042.           "_"+dateformto.substring(0,2)+
1043.           "-"+dateformto.substring(3,5)+
1044.           "-"+dateformto.substring(6,8)+
1045.           "-"+dateformto.substring(9,11)+
1046.           ":"+dateformto.substring(12,14)+
1047.           ":"+dateformto.substring(15,17)+
1048.           "_"+nameclear+"_tonometers.csv");
1049.           try {
1050.           FileWriter newJsp = new FileWriter(file);
1051.           } catch (IOException ex) {
1052.       Logger.getLogger(NewJFrame.class.getName())
        .log(Level.SEVERE, null, ex);
1053.                 }
1054.
1055.           try
1056.                {
1057.           try (FileWriter writer = new FileWriter(file))
1058.                 {
1059.                 int ton_samplecheck;
1060.                 int ton_sample;
1061.                 int ton_sampleprev = 1;
1062.
```

```
1063.                 int sp_samplecheck = 0;
1064.                 int sp_sample = 0;
1065.                 int sp_sampleprev = 1;
1066.
1067.                 int dp_samplecheck = 0;
1068.                 int dp_sample = 0;
1069.                 int dp_sampleprev = 1;
1070.
1071.                 int mp_samplecheck = 0;
1072.                 int mp_sample = 0;
1073.                 int mp_sampleprev = 1;
1074.
1075.                 int pr_samplecheck = 0;
1076.                 int pr_sample = 0;
1077.                 int pr_sampleprev = 1;
1078.
1079.
1080.                 boolean first = true;
1081.                 Date date;
1082.                 long dateu;
1083.                 long dateprev = 0;
1084.                 boolean firsttime = false;
1085.
1086.                 double ton_err = 0,
1087.                       t_err = 0,
1088.                       s_err = 0,
1089.                       vsp_err = 0,
1090.                       vdp_err = 0,
1091.                       vmp_err = 0,
1092.                       vpr_err = 0;
1093.
1094.                 double tot_smpl = cur.count();
1095.                 ArrayList ton = null;
1096.                 Iterator i_ton;
1097.
1098.                 writer.append(
1099.         "ton_sample | sp_sample | dp_sample | mp_sample
          | pr_sample | dateunix \n");
1100.
1101.         while(cur.hasNext())
1102.                 {
1103.         ton = (ArrayList)cur.next().get("ton");
1104.         date = (Date) (cur.curr().get("t"));
1105.         sp_sample = Integer.parseInt(cur.curr().get
          ("sp").toString());
1106.         dp_sample = Integer.parseInt(cur.curr().get("dp")
          .toString());
1107.         mp_sample = Integer.parseInt(cur.curr().get("mp")
        .toString());
1108. pr_sample = Integer.parseInt(cur.curr().get("pr").toString());
1109.
1110.       long dateunix = date.getTime() + 7200000;
1111.       dateprev = dateunix - 1000;
1112.       if (ton.size()!=100)
1113.                       {
1114.                 s_err += 1;
1115.                 errorextractall = true;
1116.                       }
```

```
1117.
1118.           i_ton = ton.iterator();
1119.           boolean datestamp = true;
1120.
1121.
1122.
1123.       while(i_ton.hasNext())
1124.                       {
1125.     ton_sample = Integer.parseInt(i_ton.next().toString());
1126.
1127.       if (datestamp)
1128.                 {
1129.                 writer.append(
1130.                 ton_sample+" , "+
1131.                 sp_sample+" , "+
1132.                 dp_sample+" , "+
1133.                 mp_sample+" , "+
1134.                 pr_sample+" , "+
1135.                 dateunix+"\n");
1136.                 datestamp = false;
1137.
1138.       if (dateunix - dateprev != 1000)
1139.                 {
1140.                 t_err += 1;
1141.                       errorextractall = true;
1142.                             }
1143.           }
1144.
1145.                 else
1146.                       writer.append(
1147.                       ton_sample+" , "+
1148.                       sp_sample+" , "+
1149.                       dp_sample+" , "+
1150.                       mp_sample+" , "+
1151.                       pr_sample+" , "+
1152.                                             "\n");
1153.
1154.                 if (first)
1155.                 {
1156.                 ton_sampleprev = ton_sample;
1157.                 sp_sampleprev = sp_sample;
1158.                 dp_sampleprev = dp_sample;
1159.                 mp_sampleprev = mp_sample;
1160.                 pr_sampleprev = pr_sample;
1161.                 first = false;
1162.                 continue;
1163.                 }
1164.
1165.           ton_samplecheck = ton_sample - ton_sampleprev;
1166.           ton_sampleprev = ton_sample;
1167.
1168.           if (Math.abs(ton_samplecheck) > 1)
1169.                       {
1170.                 ton_err += 1;
1171.                 errorextractall = true;
1172.                       }
1173.                       }
1174.           sp_samplecheck = sp_sample - sp_sampleprev;
```

```
1175.           dp_samplecheck = dp_sample - dp_sampleprev;
1176.           mp_samplecheck = mp_sample - mp_sampleprev;
1177.           pr_samplecheck = pr_sample - pr_sampleprev;
1178.
1179.           if (Math.abs(sp_samplecheck) > 1)
1180.                       {
1181.                       vsp_err += 1;
1182.                 errorextractall = true;
1183.
1184.                 }
1185.           if (Math.abs(dp_samplecheck) > 1)
1186.                       {
1187.                       vdp_err += 1;
1188.                 errorextractall = true;
1189.
1190.                 }
1191.           if (Math.abs(mp_samplecheck) > 1)
1192.                       {
1193.                       vmp_err += 1;
1194.                 errorextractall = true;
1195.
1196.                 }
1197.           if (Math.abs(pr_samplecheck) > 1)
1198.                       {
1199.                       vpr_err += 1;
1200.                 errorextractall = true;
1201.
1202.                 }
1203.
1204.                             sp_sampleprev = sp_sample;
1205.                             dp_sampleprev = dp_sample;
1206.                             mp_sampleprev = mp_sample;
1207.                             pr_sampleprev = pr_sample;
1208.                             dateprev=dateunix;
1209.                       }
1210.                 writer.flush();
1211.           double date_err = t_err/tot_smpl;
1212.           double value_err = ton_err/tot_smpl;
1213.           double pulserate_err = vpr_err/tot_smpl;
1214.           double sp_err = vsp_err/tot_smpl;
1215.           double dp_err = vdp_err/tot_smpl;
1216.           double mp_err = vmp_err/tot_smpl;
1217.           double sample_err = s_err/(cur.size()*ton.size());
1218.              if (app.endsWith("AllItem")==false)
1219.           jLabel_error.setText("No errors found");
1220.           else
1221.              if (errorextractall == true)
1222.           jLabel_error.setText("Found errors");
1223.
1224.           writer.append("REPORT for TONOMETER\r\n"
1225.    + "Timestamp vaules extracted: "+tot_smpl+"\r\nTimestamp
errors #: "+t_err+"\r\nTimestamp Errors perc: "+date_err+"% \r\n"
1226.    + "TON Values extracted: "+tot_smpl+"\r\nTON Values Errors
  #: "+ton_err+"\r\nTON Values Errors perc: "+value_err+"%\r\n"
1227. + "SP values extracted : "+tot_smpl+"\r\nSP values Errors
  #: "+vsp_err+"\r\nSP values Errors perc: "+sp_err+"%\r\n"
1228. + "MP values extracted : "+tot_smpl+"\r\nMP values Errors
  #: "+vmp_err+"\r\nMP values Errors perc: "+mp_err+"%\r\n"
```

```
1229. + "DP values extracted : "+tot_smpl+"\r\nDP values Errors
  #: "+vdp_err+"\r\nDP values Errors perc: "+dp_err+"%\r\n"
1230. + "PR values extracted: "+tot_smpl+"\r\nPR Values Errors
  #: "+vpr_err+"\r\nPR Values Errors perc: "+pulserate_err+"%\r\n"
1231. + "Samples values extracted: "+(cur.size()*ton.size()
    +"\r\nPR Values Errors #: "+s_err+"\r\nPR Values Errors
    perc: "+sample_err+"%\r\n");
1232.           }
1233.         }
1234.
1235.         catch(IOException e)
1236.            {
1237.                  e.printStackTrace();
1238.            }
1239.            }
1240.         }
1241.
1242.     } //GEN-LAST:event_jButton1ActionPerformed
1243.
1244.     private void jComboBox_db_collectionspatientAction
        Performed(java.awt.event.ActionEvent evt) { //GEN-
        FIRST:event_jComboBox_db_collectionspatientActionPerformed
1245.
1246.     try
1247.     {
1248.         jComboBox_db_collections.removeAllItems();
1249.         DBCollection coll;
1250.         DBCursor cur;
1251.         BasicDBObject query_pat = new BasicDBObject();
1252.
1253.         coll = db.getCollection("winal.patients");
1254. BasicDBObject fields = new BasicDBObject("uid",true).append
     ("name",true).append("surname",true);
1255.         cur = coll.find(query_pat, fields);
1256.         String uid = null;
1257.         String app2 = null;
1258.         while (cur.hasNext())
1259.         {
1260.           cur.next();
1261.     app2 = cur.curr().get("name").toString()+" "+cur.curr(
        .get("surname").toString();
1262.   if (app2.equals(jComboBox_db_collectionspatient.get
       SelectedItem().toString()))
1263.           {
1264.             uid = cur.curr().get("uid").toString();
1265.           }
1266.         }
1267.         String app = "samples."+uid;
1268.
1269.         Set<String> colls = db.getCollectionNames();
1270.         for (String s : colls) {
1271.             if (s.startsWith(app))
1272.                 jComboBox_db_collections.addItem(s);
1273.         }
1274.
1275.         jComboBox_db_collections.removeItemAt(0);
1276.         app = app+".AllItem";
1277.         jComboBox_db_collections.addItem(app);
```

```
1278.     }
1279.     catch (Exception ex) {
1280.
1281.      }
1282.     } //GEN-LAST:event_jComboBox_db_collectionspatientActionPerformed
1283.
1284.   /**
1285.    * @param args the command line arguments
1286.    */
1287.     public static void main(String args[]) {
1288.    /*
1289.    * Set the Nimbus look and feel
1290.    */
1291.    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
1292.      /*
1293. * If Nimbus (introduced in Java SE 6) is not available, stay with the
1294. * default look and feel. For details see
1295. * http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
1296.     */
1297.    try {
1298. for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing
    .UIManager.getInstalledLookAndFeels()) {
1299.     if ("Nimbus".equals(info.getName())) {
1300. javax.swing.UIManager.setLookAndFeel(info.getClassName());
1301.                     break;
1302.                    }
1303.                }
1304.            } catch (ClassNotFoundException ex) {
1305. java.util.logging.Logger.getLogger(NewJFrame.class
     .getName()).log(java.util.logging.Level.SEVERE, null, ex);
1306.            } catch (InstantiationException ex) {
1307. java.util.logging.Logger.getLogger(NewJFrame.class
     .getName()).log(java.util.logging.Level.SEVERE, null, ex);
1308.            } catch (IllegalAccessException ex) {
1309.   java.util.logging.Logger.getLogger(NewJFrame.class
     .getName()).log(java.util.logging.Level.SEVERE, null, ex);
1310. } catch (javax.swing.UnsupportedLookAndFeelException ex) {
1311. java.util.logging.Logger.getLogger(NewJFrame.class
     .getName()).log(java.util.logging.Level.SEVERE, null, ex);
1312.            }
1313.     //</editor-fold>
1314.
1315.     /*
1316.      * Create and display the form
1317.      */
1318.           java.awt.EventQueue.invokeLater(new Runnable() {
1319.
1320.               public void run() {
1321.                   new NewJFrame().setVisible(true);
1322.               }
1323.           });
1324.     }
1325.
1326.
1327.
1328.    // Variables declaration - do not modify//GEN-BEGIN:variables
1329.     private javax.swing.JButton jButton1;
1330.     private javax.swing.JButton jButton_getCount;
```

```
1331.     private javax.swing.JComboBox jComboBox_db_collections;
1332.     private javax.swing.JComboBox jComboBox_db_
          collectionspatient;
1333.     private javax.swing.JLabel jLabel1;
1334.     private javax.swing.JLabel jLabel2;
1335.     private javax.swing.JLabel jLabel3;
1336.     private javax.swing.JLabel jLabel4;
1337.     private javax.swing.JLabel jLabel5;
1338.     private javax.swing.JLabel jLabel_count;
1339.     private javax.swing.JLabel jLabel_error;
1340.     private javax.swing.JSpinner jSpinner_from;
1341.     private javax.swing.JSpinner jSpinner_to;
1342.     private javax.swing.JTextField jTextField_dbaddress;
1343.     public javax.swing.JToggleButton jToggleButton_dbconnect;
1344.     private com.mindprod.spinner.Spinner spinner1;
1345.     private com.mindprod.spinner.Spinner spinner2;
1346.     private com.mindprod.spinner.Spinner spinner3;
1347.     private com.mindprod.spinner.Spinner spinner4;
1348.   // End of variables declaration//GEN-END:variables
1349.     public static DB db;
```

APPENDICE C: CODICE WINCSTOTALCON

Main.java:

```
1.  package wincs_totalcon;
2.
3.
4.  import com.itextpdf.text.Document;
5.  import com.itextpdf.text.Font;
6.  import com.itextpdf.text.Image;
7.  import com.itextpdf.text.pdf.PdfWriter;
8.  import com.itextpdf.text.pdf.codec.PngImage;
9.  import java.awt.BorderLayout;
10. import java.awt.Color;
11. import java.awt.Dimension;
12. import java.awt.Insets;
13. import java.awt.TextField;
14. import java.awt.image.BufferedImage;
15. import java.io.BufferedReader;
16. import java.io.BufferedWriter;
17. import java.io.ByteArrayOutputStream;
18. import java.io.DataInputStream;
19. import java.io.File;
20. import java.io.FileInputStream;
21. import java.io.FileNotFoundException;
22. import java.io.FileOutputStream;
23. import java.io.FileWriter;
24. import java.io.IOException;
25. import java.io.InputStreamReader;
26. import java.io.OutputStream;
27. import java.io.PrintStream;
28. import java.io.PrintWriter;
29. import java.text.DateFormat;
30. import java.text.DecimalFormat;
31. import java.text.NumberFormat;
32. import java.text.ParseException;
33. import java.text.SimpleDateFormat;
34. import java.util.ArrayList;
35. import java.util.Date;
36. import java.util.HashMap;
37. import java.util.Iterator;
38. import java.util.List;
39. import java.util.Locale;
40. import java.util.Set;
41. import java.util.TimeZone;
42. import java.util.Vector;
43. import java.util.logging.Level;
44. import java.util.logging.Logger;
45. import java.util.Map;
46. import java.util.concurrent.TimeUnit;
47. import org.jfree.chart.ChartFactory;
48. import org.jfree.chart.ChartPanel;
49. import org.jfree.chart.JFreeChart;
50. import org.jfree.chart.axis.DateAxis;
51. import org.jfree.chart.axis.DateTickUnit;
52. import org.jfree.chart.axis.NumberAxis;
```

```
53. import org.jfree.chart.labels.StandardXYToolTipGenerator;
54. import org.jfree.chart.plot.CategoryPlot;
55. import org.jfree.chart.plot.PlotOrientation;
56. import org.jfree.chart.plot.XYPlot;
57. import org.jfree.chart.renderer.xy.StandardXYItemRenderer;
58. import org.jfree.data.category.DefaultCategoryDataset;
59. import org.jfree.data.time.TimeSeries;
60. import org.jfree.data.xy.XYSeries;
61. import org.jfree.data.xy.XYSeriesCollection;
62. import javax.swing.ImageIcon;
63. import org.jfree.chart.ChartColor;
64. import org.jfree.chart.axis.CategoryAxis;
65. import org.jfree.chart.axis.CategoryLabelPositions;
66. import org.jfree.chart.axis.NumberTickUnit;
67. import org.jfree.chart.axis.ValueAxis;
68. import org.jfree.chart.labels.ItemLabelAnchor;
69. import org.jfree.chart.labels.ItemLabelPosition;
70. import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
71. import org.jfree.chart.renderer.category.BarRenderer;
72. import org.jfree.chart.renderer.category.CategoryItemRenderer;
73. import org.jfree.chart.title.TextTitle;
74. import org.jfree.ui.TextAnchor;
75.
76. /**
77.  *
78.  * @author nino
79.  */
80. public class TotalConn extends javax.swing.JFrame {
81.
82.   /**
83.    * Creates new form TotalConn
84.    */
85.   public TotalConn() {
86.         initComponents();
87.     }
88.
89.   /**
90.    * This method is called from within the constructor to initialize the form.
91.    * WARNING: Do NOT modify this code. The content of this method is always
92.    * regenerated by the Form Editor.
93.    */
94.      @SuppressWarnings("unchecked")
95.   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
96.     private void initComponents() {
97.
98.         jButton1 = new javax.swing.JButton();
99.         jTabbedPane1 = new javax.swing.JTabbedPane();
100.          jScrollPane1 = new javax.swing.JScrollPane();
101.          jTextArea_result = new javax.swing.JTextArea();
102.          jScrollPane2 = new javax.swing.JScrollPane();
103.          jScrollPane3 = new javax.swing.JScrollPane();
104.          jLabel1 = new javax.swing.JLabel();
105.          jLabel2 = new javax.swing.JLabel();
106.
107.          setDefaultCloseOperation(javax.swing.WindowConstants
       .EXIT_ON_CLOSE);
108.          setCursor(new java.awt.Cursor(java.awt.Cursor.
       DEFAULT_CURSOR));
```

```
109.
110.          jButton1.setText("Open Log file");
111.          jButton1.addActionListener(new java.awt.event
        .ActionListener() {
112.  public void actionPerformed(java.awt.event.ActionEvent evt) {
113.                  jButton1ActionPerformed(evt);
114.              }
115.          });
116.
117.          jTabbedPane1.setMaximumSize(null);
118.
119.          jTextArea_result.setEditable(false);
120.          jTextArea_result.setColumns(20);
121.          jTextArea_result.setRows(5);
122.          jScrollPane1.setViewportView(jTextArea_result);
123.
124.          jTabbedPane1.addTab("Riepilogo", jScrollPane1);
125.          jTabbedPane1.addTab("Percentuali", jScrollPane2);
126.          jTabbedPane1.addTab("Andamento", jScrollPane3);
127.
128.          jLabel1.setText("Connection Test Report");
129.
130.  jLabel2.setText("Aprire file contenente solo totalcon ad es.
       cat Winmultiserver.log | grep totalcon > wms");
131.
132.  javax.swing.GroupLayout layout = new javax.swing.GroupLayou
      (getContentPane());
133.          getContentPane().setLayout(layout);
134.          layout.setHorizontalGroup(
135.  layout.createParallelGroup(javax.swing.GroupLayout
      .Alignment.LEADING)
136.  .addGroup(layout.createSequentialGroup()
137.  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout
      .Alignment.LEADING)
138.  .addGroup(layout.createSequentialGroup()
139.  .addGap(12, 12, 12)
140.  .addGroup(layout.createParallelGroup(javax.swing
      .GroupLayout.Alignment.LEADING)
141.  .addComponent(jTabbedPane1, javax.swing.GroupLayout
      .DEFAULT_SIZE, 857, Short.MAX_VALUE)
142.  .addGroup(layout.createSequentialGroup()
143.  .addComponent(jButton1)
144.  .addGap(29, 29, 29)
145.  .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
     586, javax.swing.GroupLayout.PREFERRED_SIZE))))
146.  .addGroup(layout.createSequentialGroup()
147.  .addGap(326, 326, 326)
148.                          .addComponent(jLabel1)))
149.                  .addGap(12, 12, 12))
150.          );
151.          layout.setVerticalGroup(
152.  layout.createParallelGroup(javax.swing.GroupLayout.Alignment
      .LEADING)
153.      .addGroup(layout.createSequentialGroup()
154.                  .addGap(14, 14, 14)
155.                  .addComponent(jLabel1)
156.  .addPreferredGap(javax.swing.LayoutStyle.Component
      Placement.RELATED)
```

```
157.  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout
      .Alignment.BASELINE)
158.                      .addComponent(jButton1)
159.                      .addComponent(jLabel2))
160.                  .addGap(18, 18, 18)
161.  .addComponent(jTabbedPane1, javax.swing.GroupLayout
    .DEFAULT_SIZE, 342, Short.MAX_VALUE)
162.                  .addGap(12, 12, 12))
163.          );
164.
165.          pack();
166.      } // </editor-fold>//GEN-END:initComponents
167.
168.      private void jButton1ActionPerformed(java.awt.event
        .ActionEvent evt) { //GEN-FIRST:event_jButton1ActionPerformed
169.
170.  javax.swing.JFileChooser FileChooser = new javax.swing
      .JFileChooser();
171.          File file = null;
172.
173.          Date datePrev, dateCurr;
174.          datePrev = new java.util.Date();
175.          int returnVal = FileChooser.showOpenDialog(this);
176.          if (returnVal == FileChooser.APPROVE_OPTION) {
177.              file = FileChooser.getSelectedFile();
178.          } else {
179.        System.out.println("File access cancelled by user.");
180.          }
181.
182.          FileInputStream fstream = null;
183.          try {
184.              fstream = new FileInputStream(file);
185.          } catch (FileNotFoundException ex) {
186.              Logger.getLogger(TotalConn.class.getName()).log
            (Level.SEVERE, null, ex);
187.          }
188.          DataInputStream in = new DataInputStream(fstream);
189.  BufferedReader br = new BufferedReader(new InputStream
      Reader(in));
190.          String strLine;
191.
192.          long time = 0;
193.          long acquisition_time = 0;
194.          int totalcon_curr = 0;
195.          int totalcon_prev = 0;
196.          int totalcon_display = 0;
197.
198.          ArrayList<Long> Ltime_diff = new ArrayList<>();
199.        ArrayList<Integer> Lconnections = new ArrayList<>();
200.
201.          Lconnections.add(0,0);
202.          Ltime_diff.add(0,Long.parseLong("0"));
203.
204.          long[] total = new long[1000];
205.          XYSeries dataSeries = new XYSeries("");
206.          try {
207.              int counter = 0;
208.       //Read File Line By Line
```

```
209.              while ((strLine = br.readLine()) != null) {
210.                  String d = strLine.substring(0/*, 15*/);
211.  SimpleDateFormat df = new SimpleDateFormat("MMM dd HH:mm:ss",
       Locale.ENGLISH);
212.                  dateCurr = df.parse(d);
213.                  String[ ] tok = strLine.split("totalcon=");
214.                  String[ ] tokens = tok[ 1] .split(" ");
215.       totalcon_curr = Integer.parseInt(tokens[ 0] );
216.
217.                  if (counter > 0) {
218.
219.       time = dateCurr.getTime() - datePrev.getTime();
220.       Lconnections.add(counter,totalcon_prev);
221.       Ltime_diff.add(counter,time);
222.
223.       acquisition_time += time;
224.
225.       total[ totalcon_prev]  += time;
226.
227.       if (totalcon_curr > totalcon_display) {
228.            totalcon_display = totalcon_curr;
229.       }
230.  //aggiungo l'istante iniziale
231.      dataSeries.add(datePrev.getTime(), totalcon_prev);
232.  //aggiungo l'istante precedente al cambio di connessioni
233.      Date dateTemp = new Date(dateCurr.getTime()-1);
234.      dataSeries.add(dateTemp.getTime(), totalcon_prev);
235.
236.                  }
237.                  totalcon_prev = totalcon_curr;
238.                  datePrev = dateCurr;
239.                  counter++;
240.              }
241.    SimpleDateFormat dtot = new SimpleDateFormat("HH:mm:ss");
242.              Date totTime = new Date(acquisition_time);
243.              jTextArea_result.setText("");
244.  jTextArea_result.setText("Durata test(days h:m:s):
      " +TimeUnit.MILLISECONDS.toDays(acquisition_time)+"
      "+ dtot.format(totTime)+"\r\n");
245. DefaultCategoryDataset dataset = new DefaultCategoryDataset();
246.
247.       // create TAB2 Chart
248.      JFreeChart chart = ChartFactory.createBarChart3D(
249.             "", // chart title
250.              "Connessioni totali", // domain axis label
251.              "%", // range axis label
252.              dataset, // data
253.              PlotOrientation.VERTICAL, // orientation
254.              false, // include legend
255.              true, // tooltips?
256.              false // URLs?
257.              );
258.
259.              for (int j = 0; j <= totalcon_display; j++) {
260.
261.  jTextArea_result.append("connessioni totali: " + j
+" : " + (float) (total[ j] * 100.0f / acquisition_time) + " %\r\n");
262.            if (total[ j]  != 0)
```

```
263.                 {
264.  dataset.setValue((float) (total[ j] * 100.0f / acquisition_time)
  , "", ""+j);
265.
266.                 }
267.
268.             }
269.
270.  ChartPanel chartPanel = new ChartPanel(chart, false);
271.  final CategoryPlot plot2 = chart.getCategoryPlot();
272.    plot2.setNoDataMessage("No data available");
273.  final CategoryItemRenderer renderer2 = plot2.getRenderer();
274.  final BarRenderer r = (BarRenderer) renderer2;
275.    r.setMaximumBarWidth(0.05);
276.  r.setBaseItemLabelGenerator(new StandardCategoryItem
      LabelGenerator());
277.    r.setBaseItemLabelsVisible(true);
278.      r.setSeriesPositiveItemLabelPosition(0,
279.      new ItemLabelPosition(ItemLabelAnchor.OUTSIDE1,
        TextAnchor.BOTTOM_LEFT, TextAnchor.TOP_LEFT, 2*Math.PI)
280.
281.             );
282.
283.  jScrollPane2.getViewport().add(chartPanel, BorderLayout.CENTER);
284.
285.             //Create TAB3 Chart
286.             DateAxis dateAxis = new DateAxis("Time");
287.             DateTickUnit unit = null;
288.         unit = new DateTickUnit(DateTickUnit.MINUTE, 15);
289.  DateFormat chartFormatter = new SimpleDateFormat("MM/dd
      HH:mm");
290.  dateAxis.setDateFormatOverride(chartFormatter);
291.             dateAxis.setTickUnit(unit);
292.  NumberAxis valueAxis = new NumberAxis("Total Conns");
293.
294.
295.  XYSeriesCollection xyDataset = new XYSeriesCollection
      (dataSeries);
296. StandardXYToolTipGenerator ttg = new StandardXYToolTip
      Generator(
297.  "{ 0} : { 2} ", chartFormatter, NumberFormat.getInstance());
298.  StandardXYItemRenderer renderer = new StandardXYItemRenderer(
299.  StandardXYItemRenderer.SHAPES_AND_LINES, ttg, null);
300.             renderer.setShapesFilled(true);
301.XYPlot plot = new XYPlot(xyDataset, dateAxis, valueAxis, renderer);
302.  JFreeChart chart1 = new JFreeChart("", JFreeChart.DEFAULT
      _TITLE_FONT, plot, false);
303.  chart1.setBackgroundPaint(java.awt.Color.WHITE);
304.
305.  ChartPanel chartPanel2 = new ChartPanel(chart1, false);
306.  chartPanel2.setPreferredSize(new Dimension(200, 70));
307.  jScrollPane3.getViewport().add(chartPanel2, BorderLayout
      .CENTER);
308.
309.             plot = chart1.getXYPlot();
310.             ValueAxis xaxis = plot.getDomainAxis();
311.             xaxis.setVerticalTickLabels(true);
312.
```

```
313.         } catch (ParseException ex) {
314.  Logger.getLogger(TotalConn.class.getName()).log(Level
      .SEVERE, null, ex);
315.         } catch (IOException ex) {
316.  Logger.getLogger(TotalConn.class.getName()).log(Level
      .SEVERE, null, ex);
317.         }
318.
319.     } //GEN-LAST:event_jButton1ActionPerformed
320.
321.     private class log_struct {
322.
323.         int totalcon;
324.         long time;
325.
326.         public log_struct() {
327.             this.time = 0;
328.             this.totalcon = 0;
329.         }
330.
331.         int getTotalCon() {
332.             return totalcon;
333.         }
334.
335.         long getTime() {
336.             return time;
337.         }
338.
339.         void setTotalCon(int totalcon) {
340.             this.totalcon = totalcon;
341.         }
342.
343.         void setTime(long time) {
344.             this.time = time;
345.         }
346.     }
347.
348.   /**
349.    * @param args the command line arguments
350.    */
351.       public static void main(String args[]) {
352.      /* Set the Nimbus look and feel */
353.      //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
354.      /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
355.       * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
356.       */
357.       try {
358.    for (javax.swing.UIManager.LookAndFeelInfo info : javax
      .swing.UIManager.getInstalledLookAndFeels()) {
359.       if ("Nimbus".equals(info.getName())) {
360.  javax.swing.UIManager.setLookAndFeel(info.getClassName());
361.                       break;
362.                   }
363.               }
364.         } catch (ClassNotFoundException ex) {
365.  java.util.logging.Logger.getLogger(TotalConn.class.get
      Name()).log(java.util.logging.Level.SEVERE, null, ex);
366.         } catch (InstantiationException ex) {
```

```
367.  java.util.logging.Logger.getLogger(TotalConn.class.get
      Name()).log(java.util.logging.Level.SEVERE, null, ex);
368.          } catch (IllegalAccessException ex) {
369.  java.util.logging.Logger.getLogger(TotalConn.class.get
      Name()).log(java.util.logging.Level.SEVERE, null, ex);
370.
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {
371.  java.util.logging.Logger.getLogger(TotalConn.class.get
        Name()).log(java.util.logging.Level.SEVERE, null, ex);
372.          }
373.      //</editor-fold>
374.
375.      /* Create and display the form */
376.          java.awt.EventQueue.invokeLater(new Runnable() {
377.              public void run() {
378.                  new TotalConn().setVisible(true);
379.              }
380.          });
381.      }
382.    //javax.swing.JFileChooser FileChooser;
383.    // Variables declaration - do not modify//GEN-BEGIN:variables
384.       private javax.swing.JButton jButton1;
385.       private javax.swing.JLabel jLabel1;
386.       private javax.swing.JLabel jLabel2;
387.       private javax.swing.JScrollPane jScrollPane1;
388.       private javax.swing.JScrollPane jScrollPane2;
389.       private javax.swing.JScrollPane jScrollPane3;
390.       private javax.swing.JTabbedPane jTabbedPane1;
391.       private javax.swing.JTextArea jTextArea_result;
392.    // End of variables declaration//GEN-END:variables
```

APPENDICE D: CODICE WINCOLLAUDI

Sezione *parser* del main:

```
1.  public class Connect extends Thread {
2.
3.          public Connect() {
4.              start();
5.          }
6.
7.          private long current = System.currentTimeMillis();
8.
9.          @Override
10.         public void run() {
11.
12.             int ack_i = 1;
13.             final byte[] ack = new byte[1];
14.             ack[0] = (byte) (ack_i >>> (0 * 8));
15.
16.           final ZeroMQ.Context context = ZeroMQ.context(1);
17.
18.         ZeroMQ.Socket client = context.socket(ZeroMQ.REP);
19.             client.bind(SERVER_ENDPOINT);
20.
21.             byte[] received = null;
22.             int battery = 0;
23.             boolean first = true;
24.
25.             long value_back = 0;
26.             while (true) {
27.                 try {
28.                     received = null;
29.                     received = client.recv(0);
30.                     long value = 0;
31.
32.                     int dev_id = 0;
33.
34.         // controllo del seriale del microcontrollore
35.         // first byte is the least significant
36.             if (received.length == 2) {
37.             value = (received[1] << 8) + received[0];
38.
39.             if (client.hasReceiveMore() == false) {
40.                                 client.send(ack, 1);
41.                             }
42.
43.              do{
44.
45.                   received = null;
46.                   received = client.recv(0);
47.
48.             if (first){
49.             new Thread(new backdoor(received)).start();
50.                   first = false;
51.                   value_back = value;
52.                             }
```

```
53.             if (value_back != value)
54.             new Thread(new backdoor(received)).start();
55.
56. // Rifiuta tutti i Winpack con seriale del microcontrollore
57. // diverso da quello impostato da GUI
58.   if ((!Long.toString(value).equals(jTextField6.getText()))
59.   &&(jCheckBox1.isSelected()==false))
60.       continue;
61.
62.   if (jCheckBox1.isSelected()==true){
63.   jTextField6.setText(Long.toString(value));
64.
65.                       }
66.   jTextField15.setText(Long.toString(value));
67. // calcolo il dev_id
68.   if (received.length == 4){
69. dev_id = ((received[3] << 24)&0x7f) + (received[2] << 16)
    + (received[1] << 8) + received[0];
70.   received = null;
71.   received = client.recv(0);
72.   if (received.length == 8)
73.   {
74.     rtcms = BigInteger.valueOf(0);
75.   BigInteger byte7 = BigInteger.valueOf(received[7]&0xff);
76.   BigInteger byte6 = BigInteger.valueOf((received[6]&0xff));
77.   BigInteger byte5 = BigInteger.valueOf((received[5]&0xff));
78.   BigInteger byte4 = BigInteger.valueOf((received[4]&0xff));
79.   BigInteger byte3 = BigInteger.valueOf(received[3]&0xff);
80.   BigInteger byte2 = BigInteger.valueOf((received[2]&0xff));
81.   BigInteger byte1 = BigInteger.valueOf((received[1]&0xff));
82.   BigInteger byte0 = BigInteger.valueOf((received[0]&0xff));
83.
84.   rtcms = rtcms.add(byte0).add(byte1.shiftLeft(8)).
85.   add(byte2.shiftLeft(16)).add(byte3.shiftLeft(24)).
86.   add(byte4.shiftLeft(32)).add(byte5.shiftLeft(40)).
87.   add(byte6.shiftLeft(48)).add(byte7.shiftLeft(56));
88.
89. Date date = new Date(Long.decode(rtcms.toString()).longValue());
90. jTextField7.setText(rtcms.toString() +"   " + date);
91.                   }
92.             }
93.           received = null;
94.           received = client.recv(0);
95.
96. // payload del modulo base
97. // arrivano 12 byte
98.         if (dev_id == 1)
99.             {
100.  mv = (received[11] << 24) + (received[10] << 16) + (received[9]
   << 8) + received[8];
101.      String app = Integer.toString(mv);
102.      jTextField8.setText(app);
103.            battery++;
104.    if (mv < 3400 || mv > 4300) {
105.            battery = 0;
106.                 }
107.    if (battery == 10) {
108.            battery = 0;
```

```
109.                  }
110.                  }
111.
112.        // payload del modulo ECG
113.        // arrivano 1000 byte
114.                 if (dev_id == 2)
115.                              {
116.                   int s1 = 0;
117.                   int s2 = 0;
118.
119.                   int l, h, m, ml, mh;
120.                   heart = 0;
121.                   int index = 0;
122.                   for (int i = 0; i < 1000; i += 4) {
123.
124.       l = ((int) received[ i]) & 0xff;
125.       h = ((int) received[ i + 2]) & 0xff;
126.       m = ((int) received[ i + 1]) & 0xff;
127.       ml = ((m >>> 4) << 8) & 0xf00;
128.       mh = (m << 8) & 0xf00;
129.
130.              s1 = l | mh;
131.              s2 = h | ml;
132.
133.     ecgsamples1=s1*0.791123;
134.     ecgsamples2=s2*0.791123;
135.
136.     ecgsample[ index] =ecgsamples1;
137.     ecgsample[ ++index] =ecgsamples2;
138.     ++index;
139.     }
140.
141.     heart = ((int) received[ 3]) & 0xff;
142.   jTextField10.setText(Integer.toString(heart));
143.
144. //System.out.println(support.ecgMaxMin(ecgsample));
145.
146. Signal app = new Signal(rtcms,ecgsample);
147.          if (graphecgon)
148.                 qecg.add(app);
149.               }
150.
151. // payload del modulo temperatura
152. // arrivano 4 byte
153.         if (dev_id == 3)
154.            {
155. temperature = (received[ 3] << 24) + (received[ 2] << 16) +
     (received[ 1] << 8) + received[ 0];
156. temperature = temperature / 10;
157. String app = Double.toString(temperature);
158.                 jTextField9.setText(app);
159.           }
160.
161.    // payload del modulo posizione
162.    // arrivano 160 byte
163.         if (dev_id == 4){
164.           int index = 0;
165.          for (int i = 0; i < 160; i += 8) {
```

```
166.  // valore compreso tra 1036 e 1044
167.  short app = received[ i + 1];
168.  app = (short) (app << 8);
169.  short posx = (short) (app | received[ i]);
170.      px = posx;
171.    jTextField22.setText(Integer.toString(posx));
172.
173.     // valore compreso tra 14 e 30
174.  app = received[ i + 3];
175.  app = (short) (app << 8);
176.  short posy = (short) (app | received[ i + 2]);
177.    py = posy;
178.  jTextField21.setText(Integer.toString(posy));
179.
180.  // valore compreso tra 1042 e 1056
181.  app = received[ i + 5];
182.  app = (short) (app << 8);
183.  short posz = (short) (app | received[ i + 4]);
184.  pz = posz;
185.  jTextField23.setText(Integer.toString(posz));
186.
187.  int posture = ((int)received[ i + 6]  & 0x07);
188.  jTextField13.setText(Integer.toString(posture));
189.
190.          axisx[ index]  = posx;
191.          axisy[ index]  = posy;
192.          axisz[ index]  = posz;
193.          ++index;
194.                  }
195.  Signal x = new Signal(rtcms,axisx);
196.    qaxisx.add(x);
197.
198.  Signal y = new Signal(rtcms,axisy);
199.    qaxisy.add(y);
200.
201.  Signal z = new Signal(rtcms,axisz);
202.    qaxisz.add(z);
203.               }
204.
205.  // payload del modulo Ecg4
206.  // arrivano 1670 byte
207.            if (dev_id == 5){
208.
209.  //   loff[0].n = ((pei->loff & 0x0001)!=0)?true:false;
210.  //   loff[1].n = ((pei->loff & 0x0002)!=0)?true:false;
211.  //   loff[2].n = ((pei->loff & 0x0004)!=0)?true:false;
212.  //   loff[0].p = ((pei->loff & 0x0100)!=0)?true:false;
213.  //   loff[1].p = ((pei->loff & 0x0200)!=0)?true:false;
214.  //   loff[2].p = ((pei->loff & 0x0400)!=0)?true:false;
215.
216.  // 1n
217.  loff[ 0]  = (received[ 0]  << 8  | (received[ 1])&0x0001);
218.  // 2n
219.  loff[ 1]  = (received[ 0]  << 8  | (received[ 1])&0x0002);
220.  // 3n
221.  loff[ 2]  = (received[ 0]  << 8  | (received[ 1])&0x0004);
222.  // 1p
223.  loff[ 3]  = (received[ 0]  << 8  | (received[ 1])&0x0100);
```

```
224.  // 2p
225.  loff[ 4] = (received[ 0] << 8 | (received[ 1])&0x0200);
226.  // 3p
227.  loff[ 5] = (received[ 0] << 8 | (received[ 1])&0x0400);
228.
229.  heartd1 = ((int) received[ 2]) & 0xff;
230.  heartd2 = ((int) received[ 3]) & 0xff;
231.  heartd3 = ((int) received[ 4]) & 0xff;
232.  heartvc = ((int) received[ 5]) & 0xff;
233.
234.  int index = 0;
235.  for (int i = 170; i < 1670; i += 6) {
236.  //LEAD 1
237.
238.  short d1 = received[ i + 1];
239.  d1 = (short) (d1 << 8);
240.  short d1mv = (short) (d1 | (((short)received[ i])&0xff));
241.  double d1v = d1mv * 0.006103516;
242.  jTextField14.setText(Double.toString(d1v));
243.    d1vect[ index] = d1v;
244.
245.  //LEAD 2
246.  short d2 = received[ i + 3];
247.  d2 = (short) (d2 << 8);
248.  short d2mv = (short) (d2 | (((short)received[ i + 2])&0xff));
249.  double d2v = d2mv * 0.006103516;
250.  jTextField24.setText(Double.toString(d2v));
251.    d2vect[ index] =d2v;
252.
253.  //LEAD 3 calcolata come d2-d1
254.      double d3v = d2v - d1v;
255.  jTextField25.setText(Double.toString(d3v));
256.    d3vect[ index] =d3v;
257.
258.  //LEAD Vc
259.  short vc = received[ i + 5];
260.  vc = (short) (vc << 8);
261.  short vcmv = (short) (vc | (((short)received[ i + 4])&0xff));
262.  double vcv = vcmv * 0.006103516;
263.  jTextField26.setText(Double.toString(vcv));
264.  vcvect[ index] = vcv;
265.
266.  ++index;
267.
268.         }
269.  Signal d1 = new Signal(rtcms,d1vect);
270.    qd1.add(d1);
271.
272.  Signal d2 = new Signal(rtcms,d2vect);
273.    qd2.add(d2);
274.
275.  Signal d3 = new Signal(rtcms,d3vect);
276.    qd3.add(d3);
277.
278.  Signal vc = new Signal(rtcms,vcvect);
279.    qvc.add(vc);
280.        }
281.
```

```
282.  // payload del modulo pulsossimetro
283.  // arrivano 64 byte
284.      if (dev_id == 8)
285.             {
286.
287.  // status
288.      status = received[63];
289.
290.  jTextField18.setText(Integer.toString(status));
291.
292.  // PulseRate: 30÷254 OxSat: 0÷99 Strenght: 0÷8
293.  pulserate = received[61]&0xFF;
294.  jTextField11.setText(Integer.toString(pulserate));
295.
296.  oxsat = received[60];
297.  jTextField19.setText(Integer.toString(oxsat));
298.
299.    strenght = received[62];
300.  jTextField20.setText(Integer.toString(strenght));
301.                                }
302.
303.  }while(client.hasReceiveMore() == true);
304.          client.send(ack, 1);
305.                      }
306.                  } catch (Exception e) {
307.                  }
308.              }
309.          }
310.      }
```

BIBLIOGRAFIA

[1] KRAMER DOUGLAS, *The Java Platform*, White Paper, Sun Microsystems, 1996.

[2] Syslog(3) manual page, *UNIX Programmer's Manual*. Berkeley, CA: USENIX Association , 1984.

[3] GILBERT MORGNER, *JFreeChart, a free Java class library for generating charts*, Publisher Full Text, 2007.

[4] LE GOC et al, *Prototype of a simple zeromq-based Rpc in replacement of Corba in Nomad*.

[5] DWORAK A. et al, *Middleware trends and market leaders* 2011. No. CERN-ATS-2011-196, 2011.

[6] iMatix ZeroMQ: *http://www.zeromq.org/*

[7] ESRF TANGO, *http://www.tango-controls.org/*

[8] BROWN WILSON, *The Architecture of Open Source Applications*, *http://www.aosabook.org/en/ZeroMQ.html*

[9] Dyke Kunz, *Object-oriented programming*, IBM Systems Journal 28.3, 1989.

[10] Hyde Paul, *Java thread programming*, Vol. 1, Sams, 1999.