Controllo e Analisi della Produzione

Ottimizzazione Miglioramento e Potenziamento Industriale

Andrea Valenti

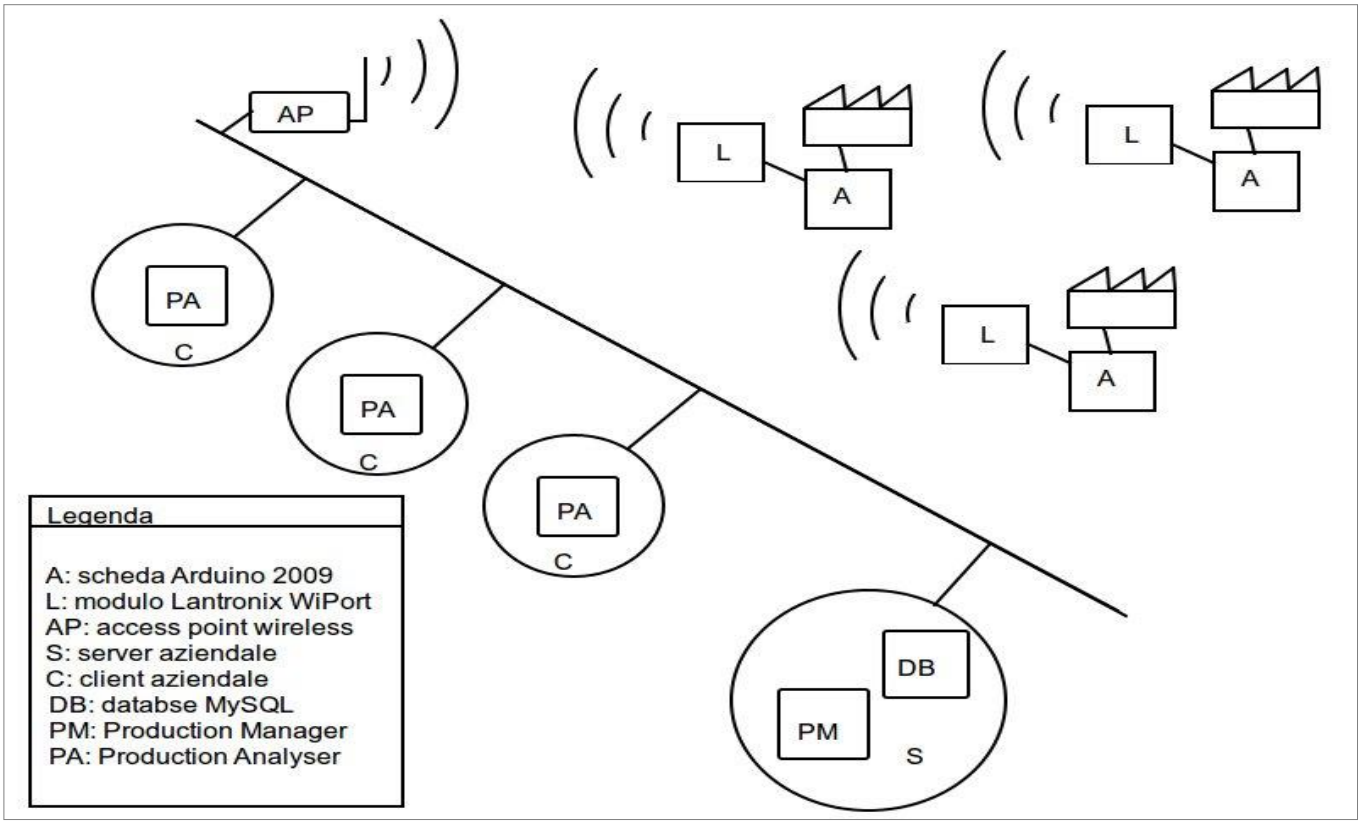
5A Informatica

ITIS Leonardo da Vinci

Anno scolastico 2011/2012

SOMMARIO

INTRODUZIONE



Lo scopo del progetto e' quello di raccogliere informazioni sull'andamento del ciclo produttivo all'interno di un'azienda, in modo da eliminare le inefficienze e individuare eventuali ottimizzazioni da effettuare.

I dati vengono rilevati da una macchina industriale generica utilizzando la scheda hardware Arduino Duemilanove. Viene creato un pacchetto seriale, il quale e' convertito in TCP/IP da un convertitore wireless Lantronix WiPort. Il pacchetto e' trasmesso sulla rete locale dell'azienda, e verra' ricevuto da un'applicativo scritto in Java in ascolto sul server centrale.

Il pacchetto viene elaborato e i dati vengono salvati in un database MySQL. In un secondo momento e' possibile visualizzarli per tenere sotto controllo lo stato della catena di produzione.

Per ridurre la mole di dati salvati all'interno del database, periodicamente sul server centrale sono mandati in

esecuzione servizi che eseguono una sintesi dei dati meno recenti.

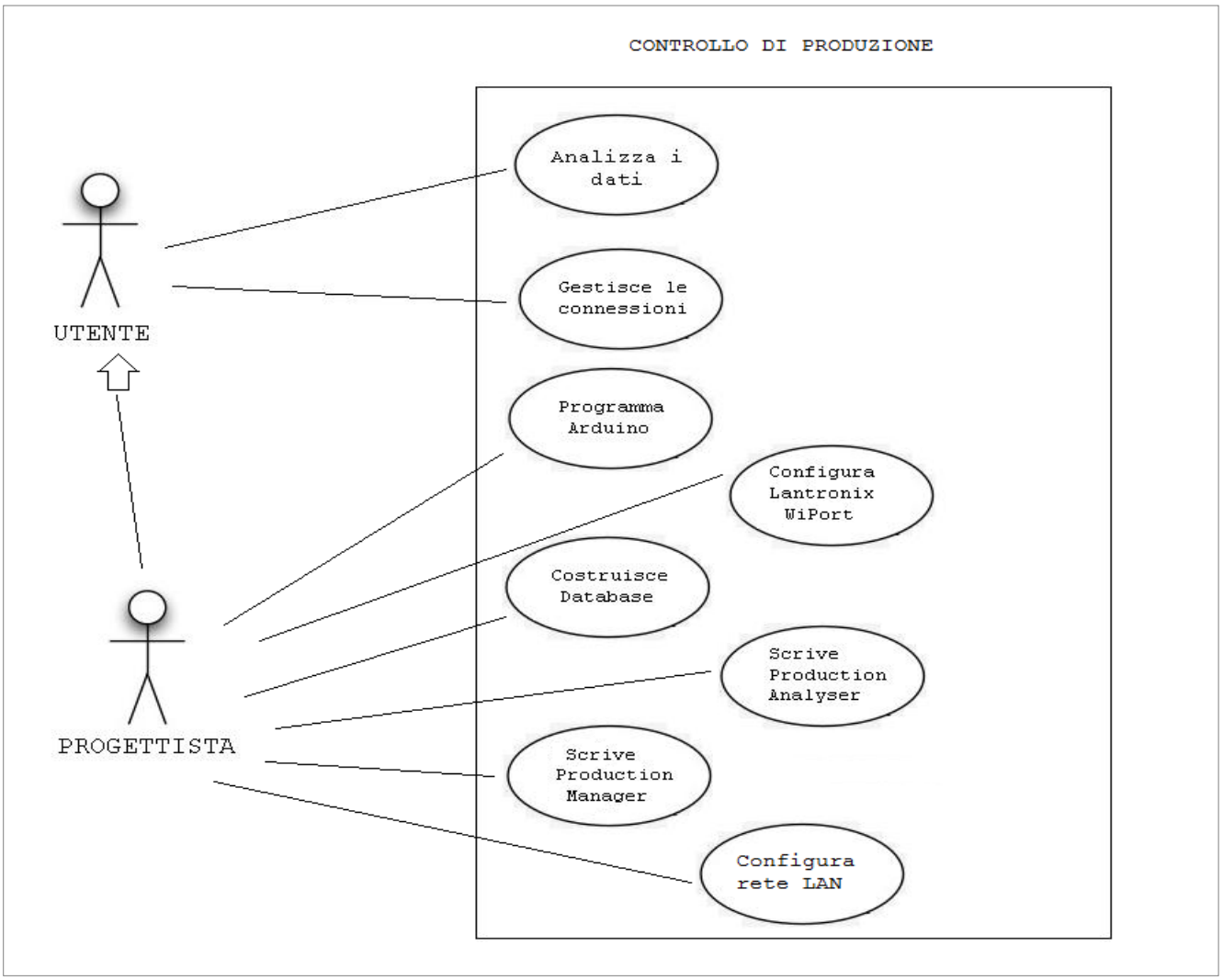
PRESENTAZIONE

L'argomento si inserisce nel 5 anno del corso di studi in particolare per la parte riguardante la gestione del database, l'interazione con esso mediante un linguaggio di programmazione e la progettazione di una, seppur semplice, rete locale.

Il progetto, pur avendo un parte informatica molto consistente, affronta diverse materie del corso di studio: sistemi, elettronica, e, in misura minore, statistica, matematica e inglese.

SVILUPPO

Di seguito e' riportato il diagramma UML dei casi d'uso del



progetto:

Dallo schema sopra si evince che i tipi di utenze saranno sostanzialmente 2: l'utente standard, che si limitera' ad utilizzare l'applicazione avendo al massimo un controllo di base sulle connessioni con i server; e il progettista, che ha il compito di scrivere gli applicativi, configurare la rete e costruire il database.

Ovviamente, il progettista avra' anche diritto ad effettuare tutte le operazioni degli utenti standard.

1. Prima parte: Progettazione della rete locale.

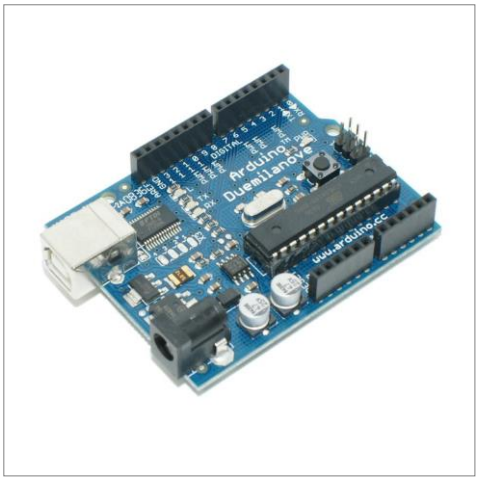
Supponendo di avere gia' una struttura fisica su cui appoggiarsi bastera' collegare sia il server centrale sia gli access-points WiFi alla LAN aziendale.

Supponendo di avere meno di 254 macchine da collegare, utilizzeremo una rete di classe C, quindi le nostre macchine e gli access-points dovranno avere IP: 192.168.x.x (se le macchine da collegare dovessero risultare maggiori di 254, bastera' utilizzare una rete di classe B, con IP: 172.16.x.x).

Se nell'azienda sono gia' presenti macchine collegate alla rete, e' buona norma racchiudere i diversi blocchi logici in diverse subnets. In questo modo e' possibile separare logicamente le macchine utilizzando pero' lo stesso mezzo fisico.

2.Seconda Parte: Rilevamento dati.

2.1 La scheda Arduino Duemilanove.



Per rilevare i dati provenienti dalla macchina industriale si e' deciso di usare una scheda Arduino Duemilanove.

Arduino è un framework open source che permette la prototipazione rapida e l'apprendimento veloce dei principi fondamentali dell'elettronica e della programmazione. Si basa su un

circuito stampato che integra un microcontrollore con PIN connessi alle porte I/O.

L'ambiente di sviluppo integrato di Arduino è un'applicazione multipiattaforma scritta in Java, ed è derivata dall'IDE creato per il linguaggio di programmazione Processing. E' inoltre fornito di una libreria software C/C++ chiamata "Wiring": la disponibilità della libreria rende molto più semplice implementare via software le comuni operazioni di input/output.

I programmi di Arduino sono scritti in C/C++, ma all'utilizzatore, è richiesto solo di definire due funzioni:

setup()

– Funzione invocata una sola volta all'inizio di un programma che può essere utilizzata per i settaggi iniziali.

loop()

– Funzione invocata ripetutamente, la cui esecuzione si interrompe solo con lo spegnimento della scheda.

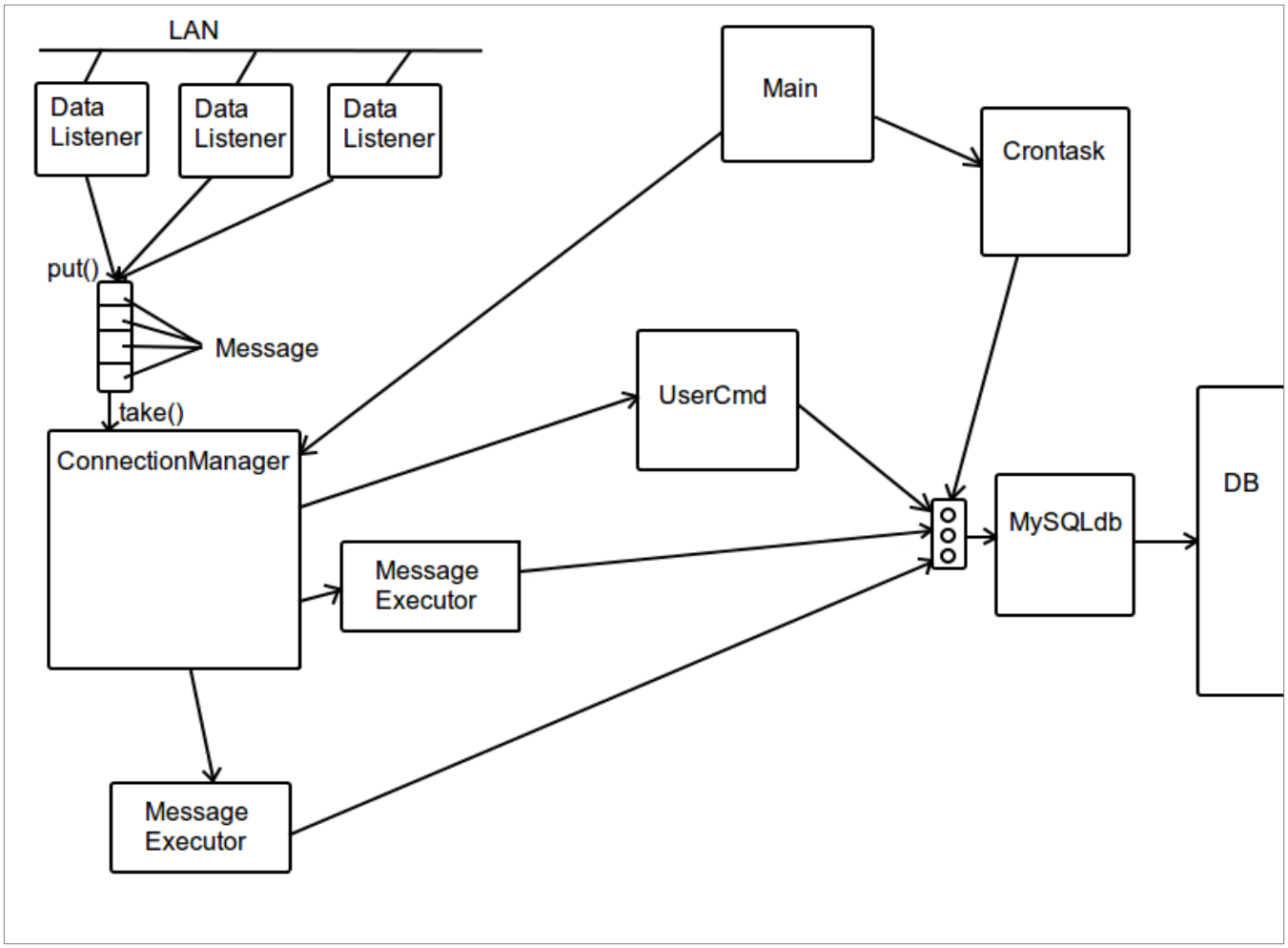
2.2 Convertitore seriale/TCP-IP Lantronix WiPort.

L'unica forma di output dell'Arduino 2009 e' sotto forma pacchetti seriali: per spedire i pacchetti su una rete LAN, occorre convertirli nel protocollo di rete TCP-IP.

Per svolgere questo compito e' stato scelto di utilizzare una scheda basata sul modulo di conversione seriale/WiFi Lantronix WiPort.

Oltre al suddetto modulo, la scheda implementa anche un semplice web server. Quando sono ricevuti dei dati tramite seriale, la scheda controllera' lo stato della connessione: se un client e' connesso, allora inviera' il pacchetto, altrimenti conservera' i dati nel buffer interno.

3.Terza Parte: L'applicazione principale.



3.1 Descrizione generale.

Il programma applicativo, scritto in Java, si occupa di risolvere tutte le problematiche correlate ad un progetto di questo tipo, dalla connessione con i server alla memorizzazione dei dati su un database. Per questo motivo il progetto e' stato diviso in 5 sotto-package, ognuno con un particolare compito da assolvere. Si e' optato per un'interfaccia testuale a linea di comando in quanto il programma potrebbe essere eseguito su server non dotati di shell grafica.

3.2 Package "database".

Il database e' di fondamentale importanza per questo tipo di applicazione. Sara' il programma ad occuparsi quasi

interamente alla gestione del database. Prima di avviare l'applicazione, e' necessarrio creare solo 2 tabelle: “avaiable_connections” e “daily_log”. La prima contiene i sockets dei server a cui bisogna connettersi, e verra' gestita dall'utente; la seconda e' invece gestita dal programma e registra l'andamento della produzione.

Col passare del tempo, verranno create le tabelle contenenti i dati relativi agli anni ed ai mesi passati.

Il package e’ composto da un’interfaccia, “DBConnector”, e 3 classi, “MySQLdb” , ”QueryResult” e “Record”. Il suo compito principale e’ quello di eseguire le query sul database e risolvere i problemi di accesso concorrente legati al multithread.

- DBConnector:

  Definisce i metodi necessari per interagire con un database.

- MySQLdb:

  Implementa DBConnector; permette la connessione, la disconnessione e l’esecuzione di query su un database MySQL. E’ dotata internamente di un campo Semaphore per evitare l’esecuzione simultanea di 2 query sullo stesso database.

- QueryResult:

  Rappresenta il risultato di una SELECT, gestito internamente sotto forma di una lista di Record.

- Record:

  Rappresenta un singolo record di un database.

3.3 Package “socket”.

Ha il compito di gestire la comunicazione tra il programma e la rete LAN aziendale. Il package contiene 2 classi, “DataListener” e “DummyServer”.

- DataListener:

Si occupa di effettuare una connessione TCP-IP con i server e, se quest'ultima va a buon fine, si mette in ascolto di dati. Una volta ricevuto un pacchetto, viene creato un “Message” (classe che vedremo piu’ avanti)

contenente i dati del pacchetto. Il messaggio e' inserito in una struttura di tipo FIFO, in attesa di essere elaborato da altre classi. La classe eredita da "Thread", permettendo cosi' un'esecuzione asincrona della classe.

- DummyServer:

Lo scopo di questa classe e' di simulare il server TCP-IP sulle schede Lantronix WiPort. Utile per provare modifiche o estensioni del programma senza dover prima cablare la rete.

3.4 Package "core".

E' il nucleo del programma, si occupa di prendere i messaggi in coda, di elaborarli e di eseguire sul database le queries opportune. E' composto da 6 classi:

- ConnectionManager:

E' la classe piu' importante del programma: e' l'unica classe chiamata direttamente dal metodo main(), si occupa di controllare le connessioni disponibili e di istanziare un thread "DataListener" per ogni server. Dopo essersi occupato delle connessioni, comincia a prelevare i messaggi in coda e crea un thread "MessageExecutor" per processare il messaggio. Inoltre, si occupa di istanziare un thread per la gestione dei comandi inviati dall'utente.

- MessageExecutor:

  Ha il compito di aggiornare il database in base ai dati del messaggio in corso di elaborazione. Sostanzialmente aggiorna la tabella "daily_log" del database, dopo avere convertito i dati in tipi riconosciusti dal database. In quanto classe derivata da "Thread", puo' essere mandata in esecuzione in modo asincrono rispetto all'applicazione.

- Machine:

  Astrazione di una generica macchina industriale.

- Message:

  Rappresenta il messaggio in attesa di essere processato.

- State:

In realta' non e' una classe, bensi' un'interfaccia. Contiene tutti i 3 possibili stati in cui puo' trovarsi una macchina industriale: WAITING, WORKING, ERROR.

- UserCmd:

  Gestore dell'interfaccia utente, esegue i comandi che l'utente invia al programma. Utile nel caso sia necessario modificare la tabella delle connessioni o per visualizzare informazioni di debug. Per maggiori informazioni riguardo i comandi dell'interfaccia utente si rimanda all'appendice A.

3.5 Package "crontask".

Per evitare l'eccessivo accumulo di informazioni obsolete all'interno del database, si e' scelto di mandare in esecuzione periodicamente dei servizi che si occupassero di riassumere le informazioni delle tabelle piu' datate.

Per fare cio' si e' deciso di sfruttare il demone "crontab", presente su tutti i sistemi operativi Unix o Unix-like (vedi appendice B).

Il Package e' composto da un'unica classe.

- Crontask:

  Si occupa di eseguire le operazioni pianificate. Tutti i metodi sono dichiarati statici per diminuire l'occupazione di memoria. La classe si divide in metodi privati, che svolgono azioni generiche e metodi pubblici, che effettivamente si occupano di ridurre la mole di dati. Il metodo daily() si occupa di interpretare i dati giornalieri calcolando i pezzi prodotti, il tempo totale di accensione e il tempo di lavoro per ogni macchina. In un secondo momento, le informazioni vengono salvate in una nuova tabella e vengono cancellate tutte le righe di "daily_log".

  Anche monthly() e annuary() si comportano in modo analogo, ma vengono eseguiti rispettivamente con cadenza mensile e annuale.

3.6 Package "utils".

E’ una raccolta di classi statiche che implementano alcune utilita’ di vario genere.

- ErrorHandler:

  Permette una gestione degli errori molto simile a quella del C.

- DateHandler:

  Implementa alcuni metodi per la gestione delle date e per la loro interazione con il database.

- SocketHandler:

  Si occupa principalmente di gestire gli streams di I/O sui sockets.

3.7 La classe Main.

E' l'entry point dell'applicazione.

Il metodo main() controlla se sono stati passati dei parametri al programma: se non ci sono parametri, verra' avviata l'esecuzione standard; se invece e' stata passata un'opzione valida, verra' mandato in esecuzione il relativo metodo di “Crontask”: daily(), monthly() o annuary().

4.Quarta Parte: Visualizzazione e analisi dei dati.

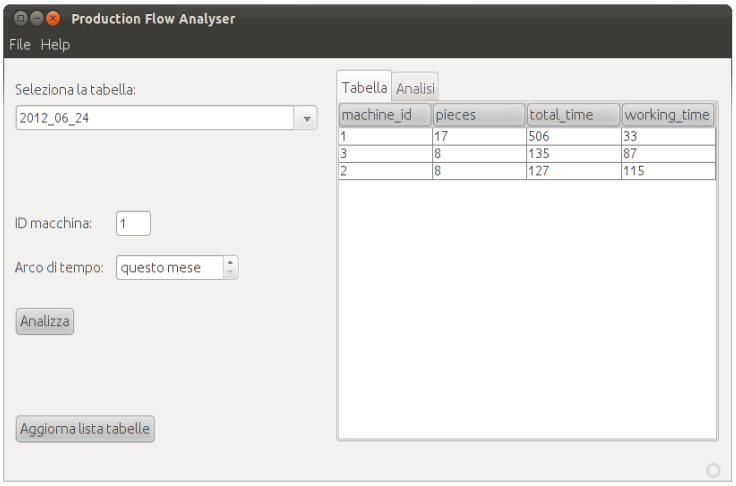
4.1 Informazioni generali.

La visualizzazione dei dati viene eseguita su un'applicazione differente dalla precedente, anch'essa scritta in Java.

Si e' preferito utilizzare il Java piuttosto che un liguaggio web-oriented (come PHP) per il semplice motivo che utilizzare applicazioni non strettamente connesse ad Internet garantisce una sicurezza maggiore rispetto ad un sito web, che potrebbe essere soggetto ad attacchi di vario tipo (DoS, SQLInjection, Buffer Overflow, Shellcode): con questi metodi un potenziale malintenzionato potrebbe riuscire ad accedere ai dati sensibili contenuti nel database.

Usare applicativi locali (ed avendo l'accortezza di distribuirli solo alle persone autorizzate) riduce enormemente rischi di questo genere.

4.2 Il layout.



Il layout dell'applicazione e' abbastanza semplice: a sinistra e' presente una listbox contenente i nomi delle tabelle del database che e' possibile consultare, qualche bottone per l'aggiornamento dei componenti e un form di input in cui si definiscono i parametri con cui effettuare l'analisi dei dati.

Sul lato destro, invece, e' presente un pannello multischeda: nella prima scheda verra' visualizzata la tabella selezionata nella listbox, nella seconda saranno visualizzati i risultati dell'analisi dei dati (vedi appendice C).

CONCLUSIONE

In conclusione il risultato globale del progetto puo' considerarsi soddisfacente: il progetto raggiunge tutti gli obbiettivi prefissati.

Dal punto di vista informatico, il progetto affronta alcuni aspetti molto interessanti: dalla la gestione del multi-thread all'interfacciamento di un'applicazione Java con un database MySQL alla gestione dei sockets fino agli aspetti piu' algoritmici legati alla manipolazione e analisi delle informazioni.

Il linguaggio Java si e' dimostrato in grado di gestire tutte le problematiche incontrate, grazie ad una libreria standard ben fornita e alla sua semplicità intrinseca. Molte classi sono implementate direttamente in modo thread-safe, rendendo la gestione della concorrenza del tutto trasparente al programmatore.

L'IDE NetBeans e' al giorno d'oggi uno dei piu' completi in circolazione, capace di gestire innumerevoli linguaggi oltre a Java (PHP, C/C++ solo per citarne alcuni). La gestione dei progetti e' semplice e intuitiva e il generatore di GUI aiuta notevolmente lo sviluppo di applicazioni grafiche. E' dotato di molte altre caratteristiche interessanti come il debugger in tempo reale, il generatore Javadoc incorporato e svariate funzioni di refractoring.

Il progetto si presta bene ad eventuali future espansioni e/o modifiche. In futuro potrebbero essere sviluppate interfacce con diversi tipi di macchine industriali, oppure potrebbe essere affinata la parte riguardante le carte di controllo prendendo in considerazione un maggior numero di parametri. In ogni caso, la struttura Object Oriented dell'applicazione presenta l'elastica' necessaria per ogni tipo di modifiche, che possono essere effettuate senza stravolgere il codice di partenza.

APPENDICI

APPENDICE A: I comandi dell'interfaccia utente.

L'utente ha a disposizione alcuni comandi per interagire con l'applicazione. Questi comandi sono generalmente orientati alla gestione delle connessioni con i server wi-fi di rilevamento dati:

- **add [ip] [porta]**: aggiunge alla tabella delle connessioni il socket [ip]:[porta].
- **delete [ID]**: elimina dalla tabella delle connessioni il socket con id = [ID]
- **list**: visualizza la tabella delle connessioni.
- **help**: visualizza la lista dei comandi.

In futuro la lista dei comandi potrebbe espandersi per comprendere il controllo su altri aspetti dell'applicazione.

APPENDICE B: Guida a crontab.

Nei sistemi operativi Unix e Unix-like, il comando "crontab" consente la pianificazione di comandi, ovvero consente di registrarli presso il sistema per essere poi mandati in esecuzione periodicamente.

"crontab" usa un demone, chiamato "crond", che è costantemente in esecuzione in background e, una volta al minuto, legge i contenuti del registro dei comandi pianificati ed esegue quelli per cui si è esaurito il periodo di attesa.

Ogni linea di un file "crontab" segue un formato particolare, composta da una serie di campi separati da spazi o tabulazioni:

```
.---------------- [m]inute: minuto (0 - 59)
|  .------------- [h]our: ora (0 - 23)
|  |  .---------- [d]ay [o]f [m]onth: giorno del mese (1 - 31)
|  |  |  .------- [mon]th: mese (1 - 12) OPPURE jan,feb,mar,apr...
|  |  |  |  .---- [d]ay [o]f [w]eek: giorno della settimana (0 - 6) (domenica=0 o 7)
|  |  |  |  |                        OPPURE sun,mon,tue,wed,thu,fri,sat
|  |  |  |  |
*  *  *  *  *  comando da eseguire
```

Per esempio, il comando:

```
02 03 * * 1,5 ls /etc/python > /opt/ctr.txt
```

ogni lunedì e venerdì, alle 3:02 del mattino visualizzera' i file presenti nella cartella /etc/python redirezionando l' output nel file ctr.txt nella cartella /opt.

Esaminiamo ora il file crontab necessario al funzionamento corretto della nostra applicazione:

```
# Production Manager crontab file

# Esegue 1 minuto dopo la mezzanotte di ogni giorno.
1 0 * * * java -jar "/run/ProductionManager.jar" -d > "/etc/cronfiles/cron_output"

# Esegue alle ore 00:03 del primo giorno di ogni mese.
3 0 1 * * java -jar "/run/ProductionManager.jar" -m > "/etc/cronfiles/cron_output"

#Esegue alle 00:05 di ogni 31 dicembre
5 0 31 12 * java -jar "/run/ProductionManager.jar" -y > "/etc/cronfiles/cron_output"
```

Tutti e tre i comandi mandano in esecuzione ProductionManager.jar passandogli ognuno un parametro differente:

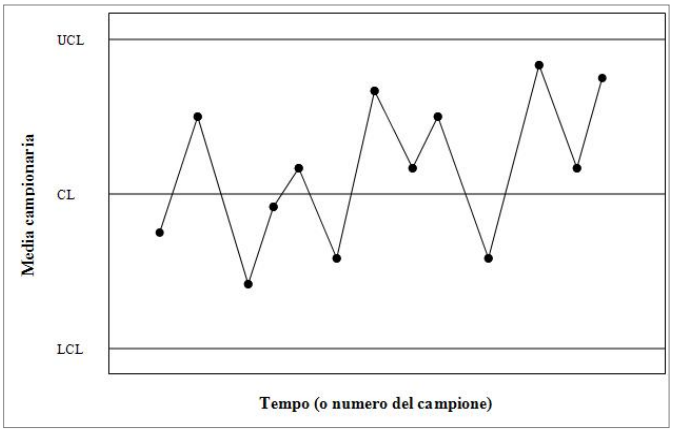
-d in caso di esecuzione giornaliera.

-m in caso di esecuzione mensile.

-y in caso di esecuzione annuale.

In tutti e tre i casi l'output del file viene rediretto nel file "cron_output" mediante l'operatore di redirezione ">"; questo risulta utile in caso di messaggi di errori non gestiti durante l'esecuzione: il testo dell'errore, essendo salvato nel suddetto file, sara' disponibile per la consultazione e per il futuro debugging.

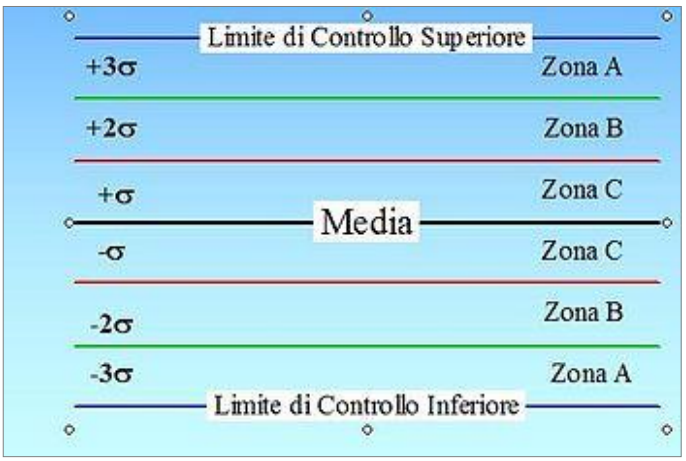
APPENDICE C: Le carte di controllo.



Le Carte di controllo sono uno strumento per mantenere sotto controllo i vari parametri di un processo.

All'interno di un processo di produzione sono presenti due tipi di variabilità:

- La **variabilità naturale** o accidentale che indica l'effetto cumulato di un gran numero di piccole cause inevitabili ed incontrollabili.
- La **variabilità sistematica** che indica distorsioni nel processo che possono essere dovute a macchine non regolari, materie prime difettose, errori degli operatori, ecc...



L'obiettivo è quello di individuare la presenza nel processo di variabilità sistematica (poiché la presenza della variabilità naturale è impossibile da eliminare e non influenza particolarmente la produzione). Se all'interno di un processo di produzione è presente solo una variabilità naturale il processo si dice **in controllo**, mentre in presenza di variabilità sistematica il processo è detto **fuori controllo**.

Oltre alla linea centrale nella carta sono presenti due limiti, uno inferiore (Lower Control Limit) ed uno superiore (Upper Control Limit) che sono determinati tramite varie considerazioni statistiche; se all'interno del processo è presente una variabilità sistematica, e quindi qualcosa che determina errori nella produzione, uno o più punti si troveranno all'esterno dell'area delimitata dai limiti di controllo.

Ci sono molte tipologie di carte di controllo, tutte con la stessa struttura base (linea mediana, linea superiore e linea inferiore di controllo).

Tra queste ricordiamo:

- Le carte di controllo per **attributi** che vengono costruite per il controllo di caratteristiche discrete o non misurabili (come ad esempio le non conformità, il successo/fallimento di qualcosa, l'accettazione o il rifiuto di un proposta, l'essere corretto o non corretto di un progetto) ecc. e si basano solo sulla rilevazione dello stato delle unità (ad esempio difettose-non difettose) o sulla presenza/assenza di qualcosa.

- Le carte di controllo per **variabili** si usano solo quando c'è la possibilità di effettuare misure molto precise e si basano sull'osservazione dei valori continui delle caratteristiche misurabili del processo (come ad esempio tempo, pesi, spessori, diametri, distanze, temperature, ecc che possono essere misurati in frazioni o decimali).

Proviamo ora ad impostare un'analisi della produzione con carte di controllo per varibili utilizzando come campioni i valori contenuti nel nostro database.

Ad esempio, vogliamo controllare se nel mese corrente il processo di produzione associato alla macchina 3 sia sotto controllo o fuori controllo: i parametri presi in considerazione saranno il numero di pezzi giornalieri prodotti. Su questi calcoleremo la carta della media e la carta del range (valore max – valore min).

Per determinare la retta centrale e le rette di controllo occorre stimare, mediante un sufficiente numero di campioni (10-15), la media ed il range ideale della produzione: a tal fine si calcola prima la media ed il range di ogni singolo campione, poi si esegue la media delle medie (X_{mm}) e la media dei range (W_m).

Dopo aver calcolato questi valori, si puo' impostare la retta centrale:

- $y = X_{mm}$

e le due rette di controllo:

- $y_{c+} = X_{mm}+3*W_m/(d_2*sqrt(n))$
- $y_{c-} = X_{mm}-3*W_m/(d_2*sqrt(n))$

Dove d_2 e' un fattore numerico che dipende esclusivamente dalla numerosita' del campione, ricavabile dalla tabella 1 (vedi sotto).

A questo punto e' possibile confrontare i valori di ogni singolo campione con quelli ricavati.

Occorre verificare se:

- Uno o piu' campioni giacciono oltre i limiti di controllo, ovvero se si e' in presenza di una deviazione dello standard di produzione.
- Piu' campioni consecutivi giacciono sopra o sotto la linea centrale.
- Piu' campioni consecutivi tendono ad assumere valori sempre piu' crescenti o decrescenti.
- Si e' in presenza di un comportamento ciclico intorno alla linea centrale.

In caso si dovesse riscontrare almeno uno di questi comportamenti, cio' significherebbe che la produzione non e' piu' sotto controllo. Occorre quindi indagare sulle cause di tale andamento.

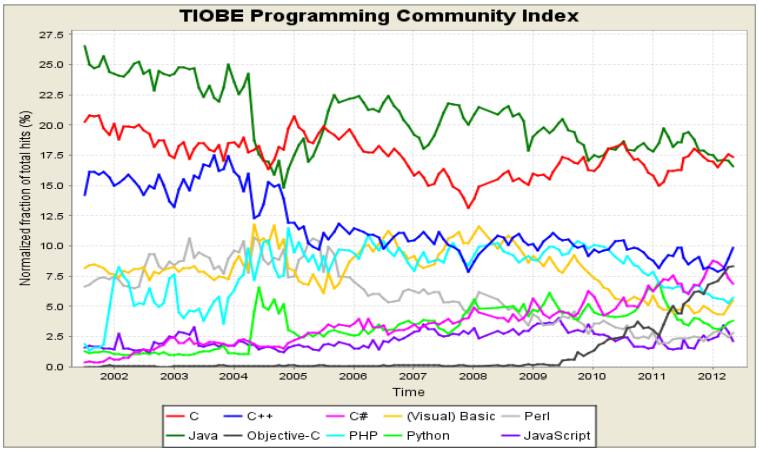
Tabella 1.

Fattori per la determinazione dei limiti di controllo

n	Fattori per $\bar{X}$			Fattori per s					Fattori per w					
	A	A_1	A_2	c_2	B_1	B_2	B_3	B_4	d_2	d_3	D_1	D_2	D_3	D_4
2	2,121	3,760	1,880	0,5642	0	1,843	0	3,267	1,128	0,853	0	3,686	0	3,267
3	1,732	2,394	1,023	0,7236	0	1,858	0	2,568	1,693	0,888	0	4,358	0	2,575
4	1,500	1,880	0,729	0,7979	0	1,808	0	2,266	2,059	0,880	0	4,698	0	2,282
5	1,342	1,596	0,577	0,8407	0	1,756	0	2,089	2,326	0,864	0	4,918	0	2,115
6	1,275	1,410	0,483	0,8686	0,026	1,711	0,030	1,970	2,534	0,848	0	5,078	0	2,004
7	1,134	1,277	0,419	0,8882	0,105	1,672	0,118	1,882	2,704	0,833	0,205	5,203	0,076	1,924
8	1,061	1,175	0,373	0,9027	0,167	1,638	0,185	1,815	2,847	0,820	0,387	5,307	0,136	1,864
9	1,000	1,094	0,337	0,9139	0,219	1,609	0,239	1,761	2,970	0,808	0,546	5,394	0,184	1,816
10	0,949	1,028	0,308	0,9227	0,262	1,584	0,284	1,716	3,078	0,797	0,687	5,469	0,223	1,777
11	0,905	0,973	0,285	0,9300	0,299	1,561	0,321	1,679	3,173	0,787	0,812	5,534	0,256	1,744
12	0,866	0,925	0,266	0,9359	0,331	1,541	0,354	1,646	3,258	0,778	0,924	5,592	0,284	1,716
13	0,832	0,884	0,249	0,9410	0,359	1,523	0,382	1,618	3,336	0,770	1,026	5,646	0,308	1,692
14	0,802	0,848	0,235	0,9453	0,384	1,507	0,406	1,594	3,407	0,762	1,121	5,693	0,329	1,671
15	0,775	0,816	0,223	0,9490	0,406	1,492	0,428	1,572	3,472	0,755	1,207	5,737	0,348	1,652
16	0,750	0,788	0,212	0,9523	0,427	1,478	0,448	1,552	3,532	0,749	1,285	5,779	0,364	1,636
17	0,728	0,762	0,203	0,9551	0,445	1,465	0,466	1,534	3,588	0,743	1,359	5,817	0,379	1,621
18	0,707	0,738	0,194	0,9576	0,461	1,454	0,482	1,518	3,640	0,738	1,426	5,854	0,392	1,608
19	0,688	0,717	0,187	0,9599	0,477	1,443	0,497	1,503	3,689	0,733	1,490	5,888	0,404	1,596
20	0,671	0,697	0,180	0,9619	0,491	1,433	0,510	1,490	3,735	0,729	1,548	5,922	0,414	1,584
21	0,655	0,679	0,173	0,9638	0,504	1,424	0,523	1,477	3,778	0,724	1,606	5,950	0,425	1,575
22	0,640	0,662	0,167	0,9655	0,516	1,415	0,534	1,466	3,819	0,720	1,659	5,979	0,434	1,566
23	0,626	0,647	0,162	0,9670	0,527	1,407	0,545	1,455	3,858	0,716	1,710	6,006	0,443	1,557
24	0,612	0,632	0,157	0,9684	0,538	1,399	0,555	1,445	3,895	0,712	1,759	6,031	0,452	1,548
25	0,600	0,619	0,153	0,9696	0,548	1,392	0,565	1,435	3,931	0,709	1,804	6,058	0,459	1,541

APPENDICE D: The Java programming language.

OVERVIEW



Java is a programming language originally developed by **James Gosling** at **Sun Microsystems** and released in 1995 as a core component of the Java platform. The language derives much of its syntax from C and C++. Java applications are compiled to **bytecode** that can run on any **Java Virtual Machine** (JVM) regardless of computer architecture. Java is specifically designed to have as few implementation dependencies as possible.

Java is currently one of the most popular programming languages in use, particularly for **client-server web applications**, with more than **10 million users** all over the world.

HISTORY

The Java project initiated in June 1991, by James Gosling and his team. Java was originally designed for interactive television, but it was too advanced for the digital cable television industry at the time.

The language was initially called Oak because of an oak tree that stood outside Gosling's office; then it was renamed in Java, from the Java coffee, said to be consumed in large quantities by the language's creators.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular.

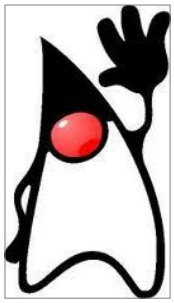
On November 13, 2006, Sun released much of Java as free and open source software, under the terms of the GNU General

Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available.

Nowadays, Java software runs from laptops to data centers, from game consoles to scientific supercomputers. There are 930 million JRE downloads each year and 3 billion mobile phones run Java.

PRINCIPLES

There were five primary goals in the creation of the Java language:



It should be "***simple***, ***object-oriented*** and ***familiar***"
It should be "***robust*** and ***secure***"
It should be "***architecture-neutral*** and ***portable***"
It should execute with "***high performance***"
It should be "***interpreted***, ***threaded***, and ***dynamic***"

CHARACTERISTICS

Java Bytecode: *the Java language code is compiled to an intermediate representation called Java bytecode, instead of directly to platform-specific machine code.*

PROs:

- **Portability**: computer programs written in the Java language must run similarly on any hardware/operating-system platform.

CONs:

- **Performance**: the overhead of bytecode interpretation means that interpreted programs almost always run more slowly than programs compiled to native executables would. However, Java programs' execution speed improved significantly with the introduction of Just-in-time compilation and HotSpot optimization. Currently, microbenchmarks show Java 7 is approximately 1.5 times slower than C.

Automatic memory management: *Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use .*

PROs:

- **Simplicity**: programmers can be spared the burden of having to perform manual memory management.

- **Code quality**: the garbage collector prevent programs from memory leaks. The memory leak occurs when a program (or a programmer) doesn't deallocate the memory no longer needed.

CONs:

- **Unpredictability:** The garbage collection could happen at any time, causing some milleconds of delay. This may be crucial in certain type of real-time application.

Object oriented paradigm: *Java uses the Object-oriented programming paradigm. A common program in Java is a series of class definition, everyone of them has a specific task to accomplish. Unlike C++, Java was built almost exclusively as an object-oriented language. All code is written inside a class, and everything is an object, with the exception of the primitive data types which are not classes for performance reasons.*

PROs:

- **Simplicity**: OOP programs are usually easier to design, write and maintain than the others.

CONs:

- **Verbosity**: OOP programs could be longer (in term of number of line of code) than the others.

EDITIONS:
Sun has defined and supports four editions of Java targeting different application environments and segmented many of its APIs so that they belong to one of the platforms. The platforms are:

- **Java Card** for smartcards.
- **Micro Edition** (Java ME) — targeting environments with limited resources.
- **Standard Edition** (Java SE) — targeting workstation environments.
- **Enterprise Edition** (Java EE) — targeting large distributed enterprise or Internet environments.

CURIOSITY:

Google and Android, Inc. have chosen to use Java as a key pillar in the creation of the Android operating system, an open-source smartphone operating system. Besides the fact that the operating system, built on the Linux 2.6 kernel, was written largely in Java, the Android SDK uses Java to design applications for the Android platform.

BIBLIOGRAFIA

AAVV, Java (Programming Language), Wikipedia, 2012.

AAVV, Arduino (Hardware), Wikipedia, 2012.

AAVV, Reference Manual, Arduino.cc, 2012.

AAVV, Javadoc Class Documentation, Oracle inc, 2012.

AAVV, Forum Discussions, Html.it, 2012.

Paul Vixie, Crontab manual, Crontab.org, 1994.

Anna Maria Gambotto Manzone/Claudia Susara Longo, Inferenza Statistica e Ricerca operativa, Tramontana, 2012.

Si desidera ringraziare tutte le persone che hanno contribuito allo svolgimento di questo progetto; in particolare i professori P. Ollari, R. Moretti, A. Melej e R. Massera per il loro fondamentale apporto su alcune parti specifiche.

Andrea Valenti