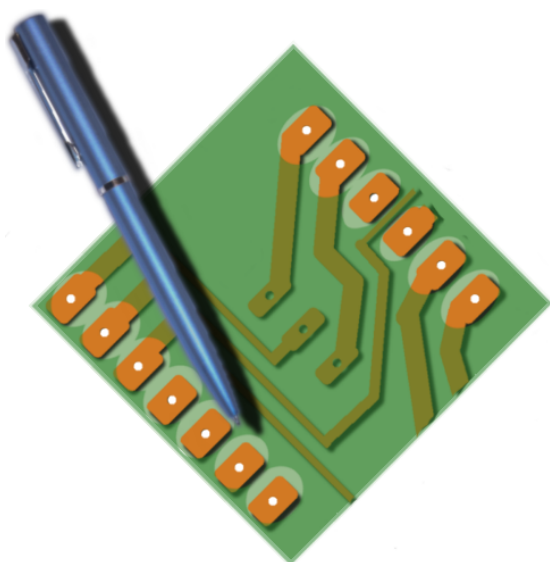


FidoCadJ 0.23

il manuale dell'utente

Davide Bucci



17 dicembre 2009

Quest'opera è soggetta alla Creative Commons Public License versione 3.0 o posteriore. L'enunciato integrale della Licenza in versione 3.0 è reperibile all'indirizzo internet:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.it>.

Si è liberi di riprodurre, distribuire, comunicare al pubblico, esporre, in pubblico, rappresentare, eseguire e recitare quest'opera alle seguenti condizioni:

ATTRIBUZIONE Bisogna attribuire la paternità dell'opera nei modi indicati dall'autore

NON COMMERCIALE Non si può usare quest'opera per fini commerciali.

NON OPERE DERIVATE Non si può alterare o trasformare quest'opera, né usarla per crearne un'altra. Ogni volta che si usa o si distribuisce quest'opera, lo si deve fare secondo i termini di questa licenza, che va comunicata con chiarezza.

In ogni caso si possono concordare con il titolare dei diritti d'autore (Davide Bucci) utilizzi di quest'opera non consentiti da questa licenza.

I nomi commerciali, i loghi, i trademark appartengono ai rispettivi proprietari.

RIASSUNTO

Questo documento è il manuale utente di FidoCadJ. Dopo una breve presentazione della storia del programma e della sua filosofia, verranno fornite le nozioni principali per disegnare un semplice schema elettrico ed un circuito stampato con FidoCadJ. La descrizione procederà poi con alcuni dettagli tecnici finora poco documentati, come la descrizione completa del formato testuale usato da FidoCad e di conseguenza da FidoCadJ...

RINGRAZIAMENTI

Molte persone hanno utilizzato il programma (o una delle sue versioni preliminari) e mi hanno fatto avere le loro impressioni. Ringrazio quindi i frequentatori del newsgroup `it.hobby.elettronica` per i loro consigli. Il programma è stato pazientemente testato sotto Linux da Stefano Martini, che è stato un infaticabile scovatore di bug. Inoltre, ringrazio, Olaf Marzocchi ed Emanuele Baggetta per i test funzionali sotto MacOSX. Ringrazio F. Bertolazzi di non darmi tregua fino a che FidoCadJ non sarà un minimo decente e di aver lavorato per la libreria CadSoft Eagle, utile per esportare i file verso questo programma. Ringrazio Celsius per le prove effettuate su FidoCadJ e sulla libreria PCB ed Andrea D'Amore per i consigli su come migliorare l'aspetto del programma su Apple Macintosh, a partire dalla versione 0.21.1. Un grazie particolare a Roby IZ1CYN per le discussioni sulle librerie e per aver scritto il paragrafo [A.2.2](#) di questo manuale, relativo all'installazione sotto Linux di FidoCadJ. Un grazie anche a Pasu, che si è preso la briga di tradurre in inglese tutto questo manuale e che ha corretto diversi refusi che avevo fatto traducendo l'interfaccia di FidoCadJ in questa lingua.

FIDOCADJ LICENSE

Copyright © 2007-2009 Davide Bucci davbucci@tiscali.it

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3 of the License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

INDICE

1	INTRODUZIONE	1
1.1	La filosofia di FidoCadJ	1
1.2	Storia del programma	1
2	IL DISEGNO CON FIDOCADJ	4
2.1	Gli strumenti di disegno	4
2.2	Disegniamo un semplice schema elettrico	7
2.3	I layer	11
2.4	La griglia	11
2.5	Disegniamo un semplice circuito stampato	11
2.6	Frecce e stili di tratto	16
2.7	Esportazione	18
2.8	Opzioni linea di comando	20
2.9	Gestione librerie	20
3	IL FORMATO DISEGNI, MACRO E LIBRERIE FIDOCADJ	22
3.1	Descrizione dell'intestazione	22
3.2	Il sistema di coordinate	22
3.3	Primitive di disegno	22
3.4	Estensioni di FidoCadJ	27
3.5	Tolleranza agli errori sintattici	27
3.6	Il formato delle librerie	28
3.7	Librerie standard	29
4	CONCLUSIONI	30
A	INFORMAZIONI SPECIFICHE PER PIATTAFORMA	31
A.1	MacOSX	31
A.1.1	Estensioni	31
A.1.2	Come scaricare ed eseguire FidoCadJ con MacO-SX	31
A.2	Linux	32
A.2.1	Estensioni	32
A.2.2	Come scaricare ed eseguire FidoCadJ su un sistema Linux	32
A.2.3	Su qualunque sistema, da terminale	32
A.2.4	Su un sistema grafico	32
A.3	Windows	33
	INDICE ANALITICO	34

ELENCO DELLE FIGURE


Figura 1	Una tipica sessione di lavoro di FidoCadJ sotto MacOSX Tiger. Nell'appendice A, sono descritte le peculiarità della versione specifica per Macintosh. 5
Figura 2	L'aspetto di FidoCadJ con il look and feel Metal. 5
Figura 3	La possibilità di ricerca all'interno delle librerie caricate. 7
Figura 4	La finestra dei parametri di un testo in un disegno FidoCadJ. 7
Figura 5	Ecco il nostro obiettivo: uno specchio di corrente che utilizza transistor NPN. 7
Figura 6	Incominciamo a disegnare un paio di transistor. 8
Figura 7	Selezioniamo e specchiamo con  il transistor a sinistra. 8
Figura 8	Siamo troppo in alto: selezioniamo tutto e spostiamoci più in basso. 9
Figura 9	Il circuito quasi terminato 9
Figura 10	Ecco come viene mostrato il circuito finito 10
Figura 11	Un semplicissimo stadio amplificatore ad un transistor NPN connesso ad emettitore comune. 12
Figura 12	I principali componenti vengono posizionati sulla piastrina. 13
Figura 13	Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni. 13
Figura 14	Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni. 14
Figura 15	Ecco lo stampato quasi terminato. 14
Figura 16	Il lavoro finito, con tutte le serigrafie. 15
Figura 17	Il circuito stampato, così come appare stampato (specchiato) su una pagina in formato ISO-UNI A4. 16
Figura 18	Uno schema elettrico di un GIC in cui sono state utilizzate alcune estensioni proprie a FidoCadJ. 17
Figura 19	La finestra dei parametri di una curva di Bézier in un disegno FidoCadJ. 17
Figura 20	L'aspetto del programma sotto MacOSX, utilizzando il look & feel Motif. 21
Figura 21	La figura 18 come probabilmente apparirebbe se letta con FidoCad. 28
Figura 22	La finestra di impostazione dei permessi, con la distribuzione Ubuntu 8.04. 33

Figura 23	Impostare l'esecuzione con la macchina virtuale Java nella distribuzione Ubuntu 8.04.	33
-----------	---	----

ELENCO DELLE TABELLE

Tabella 1	Riassunto dei comandi di disegno disponibili in FidoCadJ. Il tasto mostrato nella colonna più a sinistra permette di selezionare rapidamente l'operazione da tastiera. Un click destro in una delle modalità di inserimento delle primitive permette di passare direttamente nella finestra di modifica delle proprietà.	6
Tabella 2	Lista dei formati di esportazione disponibili con FidoCadJ.	19
Tabella 3	Presenza delle frecce sulle estremità di un segmento o di una curva di Bézier, a seconda del codice a.	23
Tabella 4	Stile delle frecce, a seconda del codice b.	23
Tabella 5	Funzione dei bit nello stile del testo.	25

INTRODUZIONE

In questo capitolo, faremo rapidamente alcune considerazioni su FidoCadJ. In particolare, vedremo qual è la filosofia che vi sta alla base, per poi ripercorrere brevemente le tappe della sua creazione e sviluppo.

1.1 LA FILOSOFIA DI FIDOCADJ

Per chi non lo conoscesse, FidoCad (senza la J alla fine!) è un software di disegno particolarmente adatto ad essere usato per tracciare semplicemente schemi elettrici e circuiti stampati. Può essere scaricato gratuitamente (per Windows) dalla pagina dell'autore, Lorenzo Lutti:

<http://www.enetsystems.com/lorenzo/fidocad.asp>

Il file generato dal programma contiene solo dei codici in formato testo ed è molto compatto, prestandosi quindi ad essere usato come formato di scambio presso le comunità Usenet. Purtroppo, il software esiste solo nella versione Windows. Chi usa Linux può utilizzare WInE, ma chi come me usa un Macintosh, deve arrangiarsi. Qui ho voluto dare un piccolo contributo scrivendo FidoCadJ (con la J alla fine, questa volta), un editor in Java, capace di visualizzare e modificare i disegni in formato FidoCad.

Chi avesse già utilizzato FidoCad, dovrebbe trovarsi abbastanza a suo agio con FidoCadJ, dato che alcuni comandi sono simili. Naturalmente, le potenzialità del programma non sono ancora del tutto sviluppate; in particolare, la gestione delle macro e delle librerie è ancora piuttosto spartana. Diciamo che il mio scopo è stato quello di fornire una soluzione minimalista alle esigenze di disegno e di sbroglio di semplici circuitini. Il tutto rispettando la filosofia del programma originale, che è quella di fornire un mezzo agile e senza troppi fronzoli. Naturalmente, il mio obiettivo è stato quello di raggiungere una compatibilità totale a livello dei file prodotti fra FidoCadJ e il FidoCad per Windows originale.

Dato che fra i miei interessi compare la tipografia al calcolatore, ed in particolare il \LaTeX , FidoCadJ si distingue dal programma originale per la possibilità di esportare i disegni in diversi formati grafici vettoriali, fra cui il Postscript incapsulato (meglio noto come EPS). A partire dalla versione 0.22, FidoCadJ può esportare anche verso il formato PDF.

L'appendice A descrive brevemente fra le altre cose le procedure di installazione di FidoCadJ nei vari sistemi operativi.

1.2 STORIA DEL PROGRAMMA

Sono un appassionato di elettronica da molto tempo. Quando alcuni anni fa ho iniziato ad affacciarmi al mondo dei newsgroup della gerarchia italiana, mi sono accorto che invece di utilizzare una grafica ASCII, la maggior parte degli schemi elettrici veniva fornita utilizzando il formato del programma FidoCad per Windows. Non usando più questo sistema operativo da molto tempo, ho deciso di fare uno sforzo per colmare la lacuna esistente.

La prima cosa che ho fatto è stata quindi di studiarli in dettaglio il formato utilizzato dal FidoCad e scrivere un'applet Java chiamata FidoReadJ, capace di interpretarlo e mostrare il risultato all'interno

Del resto, penso che sia meglio darsi da fare, piuttosto che lagnarsi che sotto un sistema operativo alternativo a Windows non si trova il tal programma. O no?

di una pagina Web. Ho fatto un po' di reverse engineering, ho poi cercato informazioni un po' dappertutto per capire quali fossero le possibilità e le limitazioni del formato ed infine mi sono scaricato e studiato i sorgenti in C++ (peraltro molto ben fatti e commentati) del FidoCad originale. Questo avveniva più o meno verso marzo 2007. Qualche mese più tardi, l'applet era pubblicata sul sito ed era stata ampiamente testata da una parte della comunità gravitante attorno al gruppo it.hobby.elettronica e it.hobby.fai-da-te.¹

In realtà, era un pallino che avevo da un pezzo. Il primo tentativo che ho fatto di scrivere un CAD 2D risale al 1993.

Disponendo di un interprete del formato, il lavoro restante per arrivare ad un editor completo è stato quello di mettere a punto la gestione dell'interfaccia utilizzatore e delle varie primitive. Il grosso del lavoro si è protratto in varie fasi da gennaio a luglio 2008. FidoCadJ non è quindi un porting di FidoCad per Windows, ma piuttosto una riscrittura totale del programma. A partire dalla versione 0.21 (pubblicata nel gennaio 2009), FidoCadJ propone delle estensioni rispetto al formato FidoCad originale. Molta attenzione è stata fatta per mantenere comunque una compatibilità all'indietro, anche se a partire dalla versione 0.23 di fine dicembre 2009 sono state introdotte delle estensioni per cui è impossibile mantenere una compatibilità con FidoCad. Per la prima volta, FidoCadJ fa cose che FidoCad non può fare.

La mia scelta di Java è motivata dal fatto che ho cambiato troppe architetture e sistemi operativi negli ultimi anni (per ragioni che non sto qui ad elencare) ed ho scelto uno strumento che non mi leghi ad una particolare architettura per il futuro. Inoltre, è un linguaggio che conoscevo già, rispetto ad altre soluzioni multiplatforma. Lo sforzo di imparare il framework Cocoa sarebbe probabilmente ripagato da un aspetto migliore del programma sotto MacOSX, ma avrebbe reso il programma non portabile, senza contare che NON sono un informatico ed il tempo che dedico a programmare non lo dedico all'elettronica.

Del resto, la struttura del codice rivela che io non sono certo un purista di Java e della programmazione ad oggetti e certe soluzioni sono più pragmatiche che eleganti.

Quello che conta, più che la scelta di questo o quel linguaggio, è secondo me l'impressione d'uso del programma da parte dell'utente finale. Per questo, sono molto attento ai vostri suggerimenti, ed in particolare a scoprire i limiti del programma o i punti in cui è necessario migliorare qualcosa. Per riassumere, non credo che Java sia la panacea di tutti mali e neppure un linguaggio perfetto. Credo tuttavia che molta della sua cattiva nomea sia dovuta a programmi di pessima qualità scritti in quel linguaggio. Senza avere pretese di perfezione e restando nell'ambito delle mie possibilità informatiche che non sono certo illimitate, intendo fare in modo che FidoCadJ NON sia un programma di pessima qualità ed è per questo che ogni commento sull'usabilità del programma e su eventuali problemi individuati sarà quanto mai benvenuto.

A partire da novembre 2009, ho aperto un progetto FidoCadJ su SourceForge. Da questa pagina si possono scaricare gli eseguibili ed i manuali di FidoCadJ. Si può anche partecipare attivamente allo sviluppo del programma, scaricandosi il codice sorgente, eventualmente utilizzando Subversion, oppure con il browser SVN fornito da SourceForge:

<http://fidocadj.svn.sourceforge.net/viewvc/fidocadj/>

¹ FidoReadJ è sempre disponibile, anche se non aggiornatissima, all'indirizzo:
<http://davbucci.cher-alice.fr/index.php?argument=elettronica/fidoreadj/fidoreadj.inc>.

1.2. Storia del programma

Per darmi una mano con FidoCadJ non occorre essere dei programmatori esperti: se volete, potete tradurre il programma in una lingua straniera, rileggere con attenzione i manuali per trovare inconsistenze o errori, curare la coerenza (e la traduzione) della libreria standard... Potete anche partecipare ai forum, scrivere una recensione del programma, suggerire miglioramenti e segnalare bug.

2

IL DISEGNO CON FIDOCADJ

Gli utilizzatori Mac di lunga data noteranno che i menu sono al loro posto!

L'uso del programma dovrebbe essere abbastanza intuitivo per chi abbia già utilizzato un programma di disegno vettoriale. L'aspetto del programma sotto MacOSX è mostrato nella figura 1; sotto altri sistemi operativi cambia qualche dettaglio (per esempio, la figura 2 mostra il risultato con il look and feel Metal di Sun), ma la filosofia resta la medesima. Vedremo quindi quali sono le funzionalità offerte dal programma, nonché gli elementi di disegno di base (le primitive), a partire dai quali ogni disegno FidoCadJ può essere ottenuto.

2.1 GLI STRUMENTI DI DISEGNO

Questo modo di funzionare è ispirato alle vecchie radio a valvole ed ai commutatori in voga fino agli anni 70.

Nella barra dei comandi (in alto), si trovano le funzioni più utilizzate, che permettono di creare e modificare un disegno. La tabella 1 mostra un rapido riassunto delle funzionalità dei comandi disponibili e descrive le azioni possibili. Noterete che una volta che viene premuto un bottone, questo rimarrà premuto fino a quando non si selezionerà un'altra funzione dalla barra. Nella barra, è possibile scegliere quale primitiva di disegno verrà introdotta.¹ Sulla destra, una lista attivabile con un click mostra il layer corrente (riferitevi al paragrafo 2.3 per maggiori informazioni).

La barra dei comandi è parzialmente personalizzabile. In particolare, l'utente può scegliere se mostrare in ogni bottone l'icona più il testo, oppure solo l'icona. Le icone sono inoltre disponibili in due formati selezionabili a piacere. Per scegliere queste impostazioni, basta andare nel menu "Vista/Opzioni".² Le modifiche saranno attive al riavvio successivo del programma, dato che presumibilmente non si tratta di qualcosa che va cambiato tutti i giorni. Le figure 1 e 2 mostrano quanto si ottiene nella barra dei comandi (subito sotto il titolo della finestra), configurata per mostrare il testo e le icone, nella loro versione più piccola. Nella seconda riga, a partire da sinistra si vedono le impostazioni di zoom ed i bottoni "Adatta", "Mostra griglia" e "Blocca sulla griglia". Il primo permette di selezionare automaticamente le impostazioni di zoom più adatte a mostrare la totalità del disegno. Il secondo permette di selezionare se mostrare oppure no la griglia. Il terzo permette all'utente di scegliere se desidera allineare gli elementi alla griglia durante l'introduzione, oppure no.

Sulla destra, sono mostrati sotto forma di albero gli elementi (chiamati macro) delle librerie caricate nel programma. Basta selezionare un elemento della libreria e poi fare click nel disegno per poterlo inserire. Le librerie di FidoCadJ includono tutti i simboli classici degli schemi elettrici, nonché una buona collezione di footprint per circuiti stampati. A partire dalla versione 0.22, un campo di testo con la lente visibile sopra la lista ad albero (vedere figura 3) permette di ricercare rapidamente un elemento fra quelli presenti nelle librerie caricate dal

¹ Per avere maggiori informazioni sul formato con cui gli elementi del disegno vengono memorizzati, riferitevi alla sezione 3.3.

² Tranne su MacOSX, in cui questa voce si trova nel menu FidoCadJ e si chiama "Preferenze".

2.1. Gli strumenti di disegno

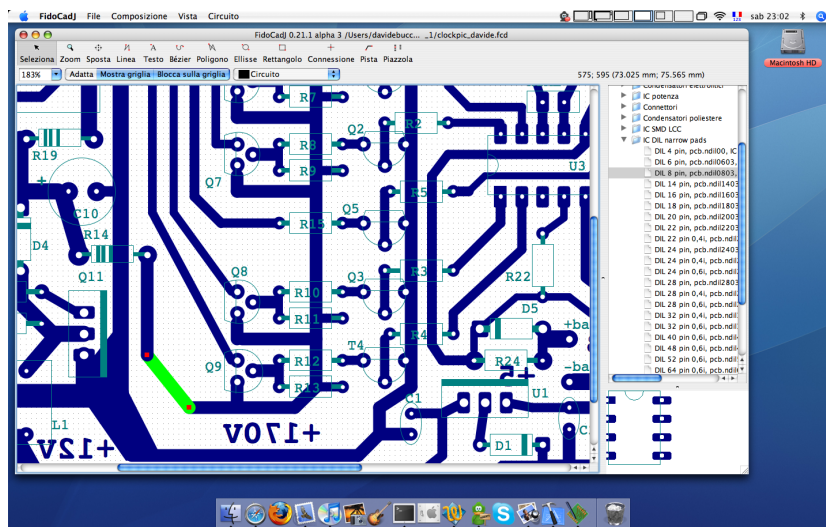


Figura 1: Una tipica sessione di lavoro di FidoCadJ sotto MacOSX Tiger. Nell'appendice A, sono descritte le peculiarità della versione specifica per Macintosh.

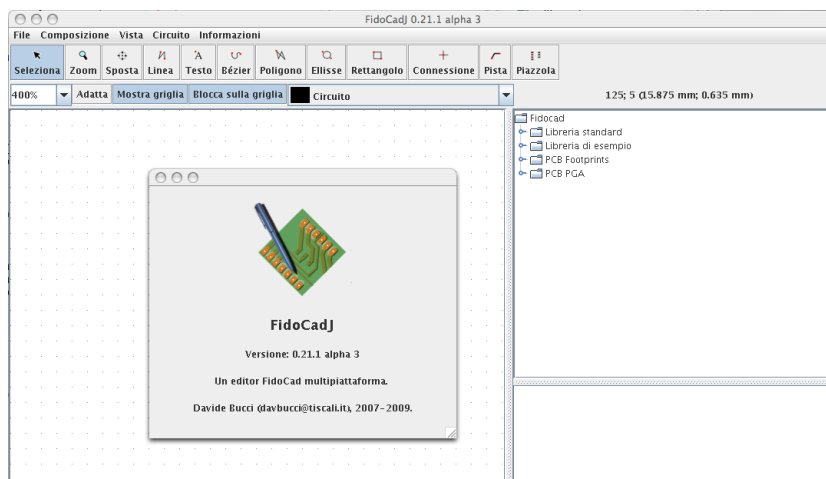


Figura 2: L'aspetto di FidoCadJ con il look and feel Metal.













Tasto	Comando	Uso
A o Spazio	 SELEZIONA	Selezione di uno o più elementi grafici. Premere Control (Command solo sotto MacOSX) per effettuare selezioni multiple o deselezionare un solo elemento. Fare click e trascinare per selezionare più elementi in un'area. Premere R per ruotare gli elementi selezionati. Premere S per specchiare gli elementi selezionati. Fare doppio click su un elemento per modificarne le proprietà.
	 ZOOM	Fare click con il pulsante sinistro per aumentare lo zoom. Fare click con il pulsante destro per diminuirlo.
	 SPOSTA	Cliccare nel disegno e spostare il mouse per fare scorrere il disegno.
L	 LINEA	Introduce una linea o una serie di linee. Premere Esc o fare doppio click con il pulsante sinistro del mouse per terminare l'introduzione.
T	 TESTO	Introduce una stringa di testo.
B	 BÉZIER	Disegna una curva di Bézier.
P	 POLIGONO	Disegna un poligono pieno o vuoto. Fare doppio click o premere Esc per terminare l'introduzione dei vertici.
E	 ELLISSE	Disegna un ellisse pieno o vuoto (tener premuto Control per ottenere un cerchio).
G	 RETTANGOLO	Disegna un rettangolo pieno o vuoto.
C	 CONNESSIONE	Disegna una connessione elettrica in uno schema.
I	 PISTA C. S.	Disegna una pista di circuito stampato. La larghezza di default può essere modificata nella finestra "Vista/Opzioni".
Z	 PIAZZOLA C. S.	Disegna una piazzola per circuito stampato. Le dimensioni di default possono essere modificate nella finestra "Vista/Opzioni".

Tabella 1: Riassunto dei comandi di disegno disponibili in FidoCadJ. Il tasto mostrato nella colonna più a sinistra permette di selezionare rapidamente l'operazione da tastiera. Un click destro in una delle modalità di inserimento delle primitive permette di passare direttamente nella finestra di modifica delle proprietà.

2.2. Disegniamo un semplice schema elettrico

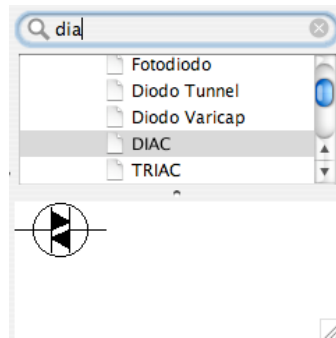


Figura 3: La possibilità di ricerca all'interno delle librerie caricate.

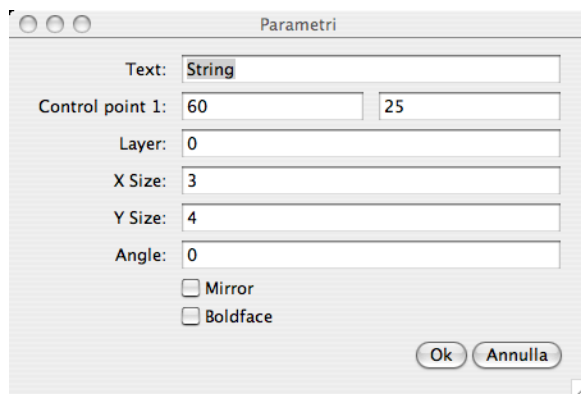


Figura 4: La finestra dei parametri di un testo in un disegno FidoCadJ.

programma. Usando le frecce su e giù, è possibile navigare all'interno di tutti gli elementi trovati.

La figura 4 mostra un esempio di quello che si ottiene facendo doppio click, in modalità selezione, su un elemento del disegno, in questo caso una stringa di testo. All'interno della finestra, è possibile modificare ogni aspetto (coordinate, rotazione...) di ogni primitiva di disegno. Naturalmente, l'aspetto non sarà sempre il medesimo, poiché le informazioni che possono venir modificate dipenderanno dall'elemento grafico che si sta modificando.

2.2 DISEGNAMO UN SEMPLICE SCHEMA ELETTRICO

Per capire come va usato il programma, procederemo a disegnare un semplicissimo schema elettrico, come quello mostrato in figura 5. Per

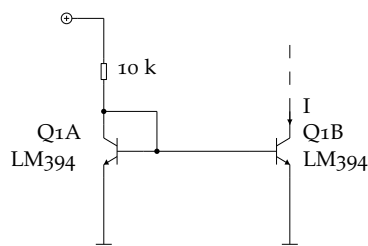


Figura 5: Ecco il nostro obiettivo: uno specchio di corrente che utilizza transistor NPN.

2. IL DISEGNO CON FIDOCADJ

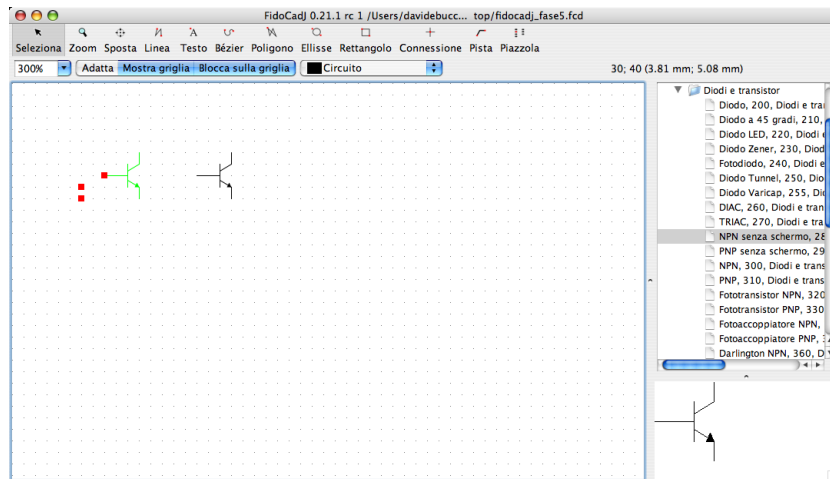


Figura 6: Incominciamo a disegnare un paio di transistor.

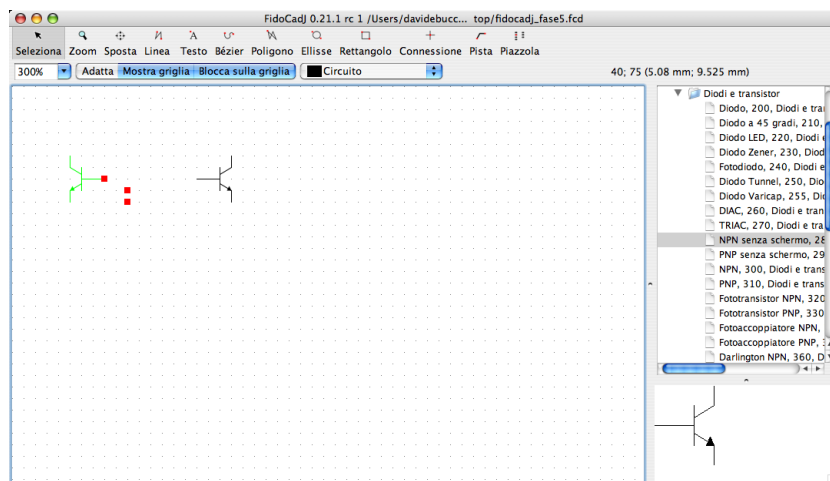


Figura 7: Selezioniamo e specchiamo con **[S]** il transistor a sinistra.

iniziare, lanciamo quindi il programma oppure, se è già in esecuzione, creiamo un nuovo disegno dal menu “File/Nuovo disegno”.

Iniziamo quindi ad introdurre nel disegno i simboli dei due transistor, che sono un po’ il cuore del nostro schema. Per fare questo, dovremmo utilizzare le macro a disposizione nella libreria standard, che è caricata per default e si trova sulla destra del programma. La macro che ci serve è chiamata “NPN senza schermo” e si trova nella categoria “Diodi e transistor”, all’interno della “Libreria standard”. Facendo click per selezionare la macro desiderata, è poi possibile introdurla dove si vuole all’interno del disegno, facendo click una seconda volta nella posizione voluta. A questo punto, dovremmo trovarci davanti a una situazione simile a quella mostrata nella figura 6.

Una cosa che si nota subito è che il transistor bipolare sulla sinistra non è orientato correttamente. Basta fare click su “Seleziona”, nella barra degli strumenti, selezionare il transistor (apparirà verde, con tre punti di controllo identificati da dei quadratini rossi) e poi premere **[S]** per specchiarlo. Otterremo quindi il risultato visibile in figura 7.

Utilizzando lo strumento “Linea” nella barra degli strumenti, potremo completare qualche connessione elettrica, fino ad accorgerci di aver cominciato lo schema un po’ troppo vicino ai bordi dell’area di

2.2. Disegniamo un semplice schema elettrico

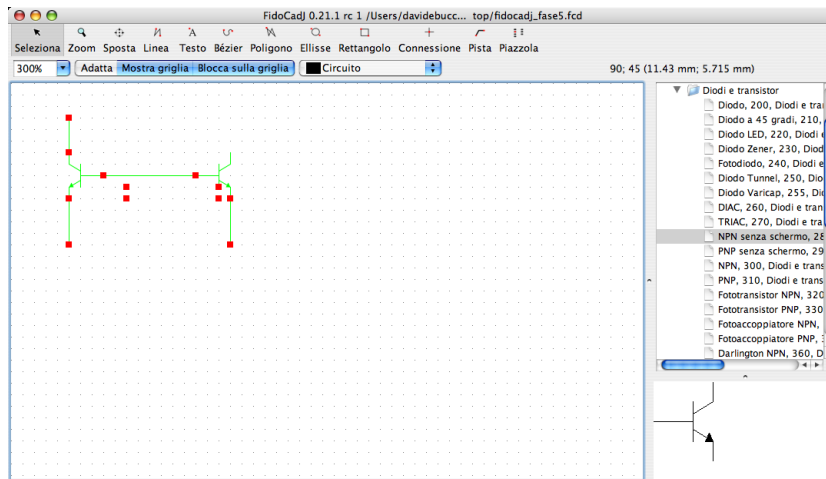


Figura 8: Siamo troppo in alto: selezioniamo tutto e spostiamoci più in basso.

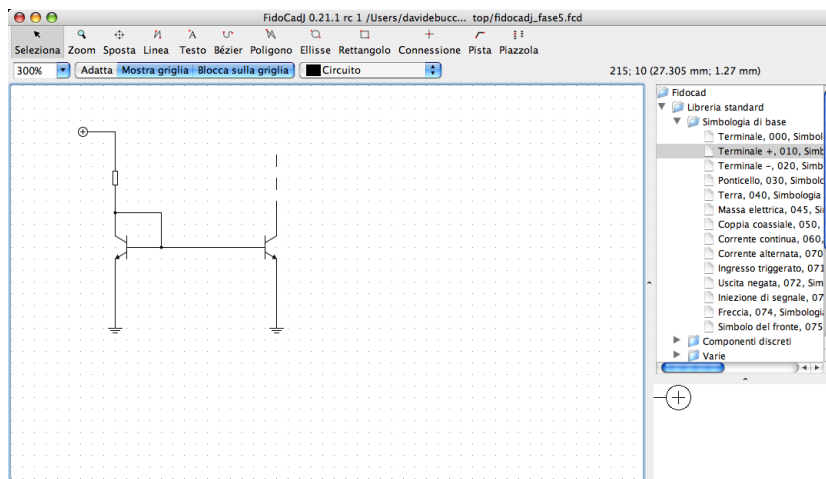


Figura 9: Il circuito quasi terminato

disegno. Poco male: in modalità “Seleziona”, potremo fare click in alto a sinistra e, sempre tenendo premuto il pulsante sinistro del mouse, trascinarlo in basso a destra. Apparirà un rettangolo con i bordi verdi, che indicherà che stiamo cercando di selezionare d’un colpo tutti gli elementi al suo interno. Dato che dobbiamo spostare tutto quello che abbiamo disegnato fino ad ora, dovremmo selezionarli dapprima tutti (come si vede in figura 8). A questo punto, basterà, sempre in modalità selezione, fare click su un elemento qualsiasi selezionato e trascinare tutto lo schema nella posizione voluta.

Continuiamo quindi ad introdurre le parti mancanti del circuito, in particolare un resistore (Libreria standard/Componenti discreti/Resistore) ed il terminale di alimentazione positiva (Libreria standard/Simbologia di base/Terminale +). Quest’ultimo, andrà ruotato per fargli assumere l’orientazione voluta. Basterà selezionarlo e premere **R** fino ad ottenere il risultato desiderato. Dovremmo quindi avere qualcosa di simile alla figura 9.

Manca ora solo l’introduzione delle stringhe di testo e della freccia rappresentante il verso della corrente. Per quest’ultima, c’è la macro “Freccia”, in “Libreria standard/Simbologia di base”. Per il testo, bi-

2. IL DISEGNO CON FIDOCADJ

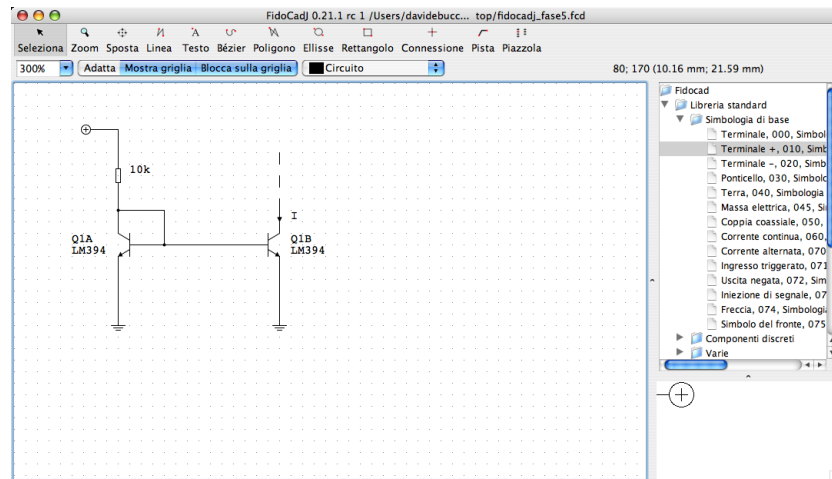


Figura 10: Ecco come viene mostrato il circuito finito

sogna usare il bottone “Testo” nella barra dei comandi e cliccare nel disegno alla posizione desiderata. Verrà introdotto un testo “String”, di cui potranno essere modificate le caratteristiche, facendovi doppio click in modalità selezione. Riferitevi alla figura 4. Il nome e il modello del transistor utilizzato (una coppia di transistor, in realtà) sono specificati all’interno dei campi “Name” e “Value” che si ottengono facendo doppio click in modalità “Selezione” sulla macro.³ Una dimensione adatta per lavorare con i circuiti è 4 unità in verticale e 3 in orizzontale. Il circuito finito è mostrato in figura 10.

Per chi fosse curioso, ecco a cosa assomiglia il codice che descrive il circuito. Per vederlo, basta selezionare “Testo del circuito” dal menu “Circuito”. Il tutto è pronto per essere copiato ed incollato su un messaggio di posta elettronica, in un newsgroup, oppure un forum.

```
[FIDOCAD]
MC 95 65 0 0 280
FCJ
TY 115 60 4 3 0 0 0 * Q1B
TY 115 65 4 3 0 0 0 * LM394
MC 55 65 0 1 280
FCJ
TY 20 60 4 3 0 0 0 * Q1A
TY 20 65 4 3 0 0 0 * LM394
LI 55 65 95 65 0
LI 40 75 40 95 0
LI 110 75 110 95 0
LI 40 40 40 55 0
MC 40 30 0 0 115
LI 40 15 40 30 0
LI 30 15 40 15 0
MC 30 15 2 0 010
LI 40 50 60 50 0
LI 60 50 60 65 0
SA 60 65 0
SA 40 50 0
LI 110 45 110 55 0
LI 110 35 110 40 0
```

³ La possibilità di associare un nome ed un valore ad una macro, ovvero ad un simbolo di un componente, è in realtà un’estensione di FidoCadJ non presente nel FidoCad originale. Vedere il paragrafo 3.4 per maggiori informazioni sulla compatibilità.

```
LI 110 25 110 30 0
MC 40 95 0 0 040
MC 110 95 0 0 040
TY 45 30 4 3 0 0 0 * 10 k
TY 115 50 4 3 0 0 0 * I
MC 110 50 1 0 074
```

Se siete interessati a comprendere il formato, in questo manuale c'è una descrizione dettagliata al capitolo 3.

Non è comunque obbligatorio passare per la finestra "Testo del circuito"; basta infatti selezionare il disegno nell'editor ed incollarlo nel testo che si sta scrivendo: apparirà il codice automaticamente.

2.3 I LAYER

Una maniera di immaginare un layer è quello di un disegno fatto su un acetato trasparente. Il disegno finale sarà dato dalla sovrapposizione di più layer, che verranno sovrapposti come fogli di acetato. Ogni layer è distinto da un colore e può essere disegnato, oppure no. Questo modo di lavorare è comune a molti sistemi di disegno elettronico, perché permette facilmente di rappresentare le diverse parti che poi verranno sovrapposte, per esempio in un circuito stampato.

FidoCadJ permette di utilizzare 16 layer, che sono numerati da 0 a 15. La funzione di ogni layer è basata su una convenzione ed in particolare, il layer zero sarà quello utilizzato per gli schemi elettrici, il layer 1 per il rame lato saldature, il layer 2 per il rame lato componenti ed il layer 3 per le serigrafie. I restanti layer non sono associati ad una specifica funzione e potete usarli come meglio credete. Il nome ed il colore di ogni layer si può specificare attraverso il menu "Vista/Layer". Dallo stesso menu si può decidere se disegnare o meno un certo layer a schermo, oppure in stampa.

L'ordine dei layer è importante. In particolare, i layer con numero più basso vengono disegnati per primi. Disegni presenti su layer successivi potranno coprire quindi quanto presente sui layer più bassi.

2.4 LA GRIGLIA

L'unità logica di FidoCadJ è di 5 mils (127 micron) e non si possono avere "mezze unità", nel senso che le coordinate di qualsiasi elemento grafico devono per forza essere intere. Questo permette di ottenere una risoluzione sufficiente per disegnare uno schema elettrico e la maggior parte dei circuiti stampati. Per comodità, comunque, il programma può impostare una griglia più larga e fare in modo che ogni operazione fatta con il mouse venga allineata al punto della griglia più vicino. Per questa ragione, sono presenti due bottoni, "Mostra griglia" e "Blocca sulla griglia", che permettono appunto di mostrare il reticolato utilizzato e di attivare o meno l'operazione di aggiustamento sui punti della griglia. Il passo della griglia può essere scelto all'interno della finestra che si attiva dal menu Vista e poi Opzioni.

2.5 DISEGNAMO UN SEMPLICE CIRCUITO STAMPATO

Per impratichirsi con tutto quanto visto fino ad ora, il modo migliore è vedere come fare a disegnare un circuito stampato usando FidoCadJ. A differenza di altri programmi di CAD elettrico che sono indubbiamente

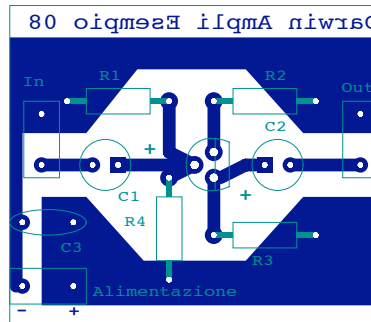


Figura 11: Un semplicissimo stadio amplificatore ad un transistor NPN connesso ad emettitore comune.

Il lettore si faccia animo: un po' di ragionamenti fatti con carta e matita (e moltissime gomme) permettono a conti fatti di guadagnare tempo per ottenere un'idea precisa poi da sviluppare sul calcolatore.

molto potenti, ma anche piuttosto difficili da usare, FidoCadJ fornisce sostanzialmente una versione elettronica dei vecchi trasferibili R41. Ovviamente, il fatto di lavorare su un calcolatore permette di beneficiare di tutta la flessibilità offerta dal mezzo.

Bisogna notare che il progetto di un circuito stampato, specie se complesso, non è affatto un compito semplice. Esistono, è vero, autoplayer ed autorouter che promettono miracoli nei depliant illustrativi dei grandi CAD, ma è indubbio che si tratti di un lavoro in cui l'intelligenza e l'esperienza di chi lo compie ha ancora un'importanza fondamentale. FidoCadJ costituisce un sistema molto rapido ed immediato per disegnare piccoli circuiti stampati su scala casalinga. Vedremo qui brevemente come fare per disegnarne uno semplicissimo, ma completo.

Quello che consiglierai a chi si accinge a fare questo lavoro è di avere già un'idea abbastanza precisa di dove piazzare i componenti e come far passare le piste di modo che si incrocino il meno possibile.

Qui bariamo un po' ed iniziamo direttamente dal risultato che vogliamo ottenere, mostrato in figura 11. Si tratta di un semplice circuito di un amplificatore ad emettitore comune realizzato attorno ad un NPN, tipo BC547 o similari. Sarà utile immaginarsi la piastra come se la vetroresina fosse trasparente, guardando da sopra il circuito, dal lato componenti. A questo proposito, è utile la serigrafia dei componenti che aiuterà a non perdersi nei collegamenti, anche se magari in un progetto casalingo non verrà riportata sulla vetroresina.

La prima cosa che consiglierai di fare è di piazzare approssimativamente i componenti, nel nostro caso il transistor (biblioteca "PCB footprints/Semiconduttori 3 terminali/TO92"), i resistori ("PCB footprints/Resistori/Resistore 1/4 W 0,4 i"), i condensatori elettrolitici ("PCB footprints/Condensatori elettrolitici/Vert. diam. 5 passo 2,5"). Può essere utile marcare la dimensione totale desiderata del circuito, introducendo un rettangolo vuoto sul layer delle serigrafie (il 3). Per fare questo, basta utilizzare la primitiva rettangolo avendo dapprima selezionato il layer in cui questo va introdotto. Si dovrebbe ottenere il risultato mostrato in figura 12.⁴

Si possono quindi introdurre le regioni ramate che forniranno l'alimentazione positiva e negativa. Per fare ciò, bisognerà introdurre dei poligoni con la primitiva apposita e poi farci sopra doppio click in modalità selezione, per specificare all'interno della finestra di dialogo che

⁴ FidoCadJ è attualmente distribuito con una nazionalizzazione italiana, italiana con varianti svizzere, francese ed inglese. È molto semplice aggiungere una nuova lingua. Nel caso, fatevi sentire! C'è anche bisogno di lavoro per tradurre le librerie (almeno quelle standard), nonché questo manuale.

2.5. Disegniamo un semplice circuito stampato

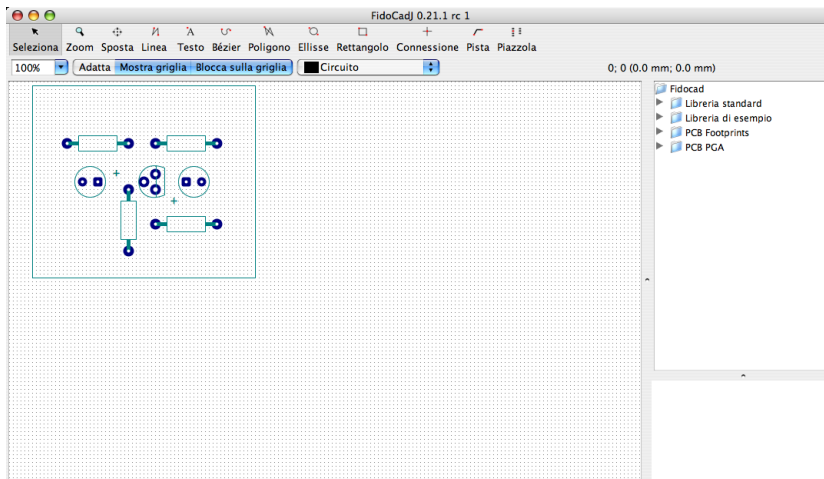


Figura 12: I principali componenti vengono posizionati sulla piastrina.

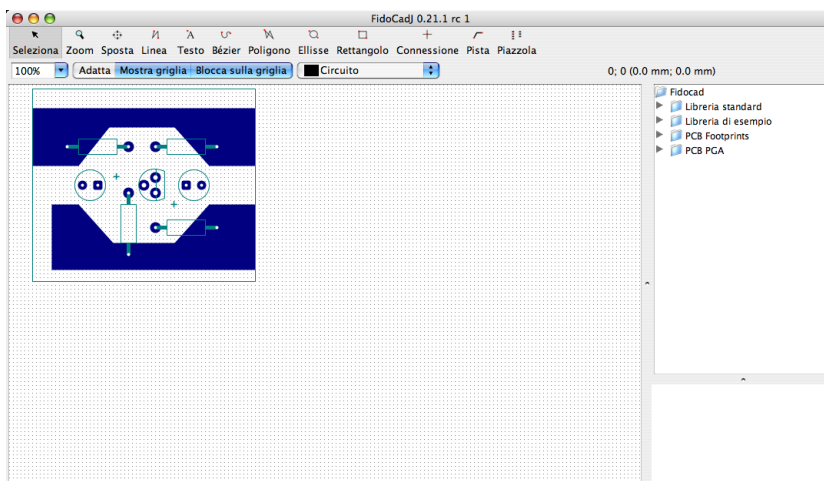


Figura 13: Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni.

apparirà che quello che vogliamo è un poligono pieno. Fate attenzione a specificare anche il layer corretto, che è il numero 1, ovvero il rame lato saldature. L'uso di un poligono per i piani di alimentazione può essere utile per essere sicuri di disporre di collegamenti a bassa impedenza parassita. Con un po' di attenzione, dovrebbe essere possibile ottenere il risultato mostrato in figura 13.

Bisognerà poi completare le connessioni elettriche utilizzando la primitiva PCB line. Ho scelto di utilizzare uno spessore di 10 unità (1,27 mm), che è una larghezza comoda, per esempio per facilitare le saldature.

Ci rendiamo conto di aver bisogno di alcuni connettori: per far entrare il segnale, per prelevare e per l'alimentazione. Possiamo tranquillamente utilizzare un footprint previsto per un condensatore al poliestere. Avrà probabilmente la dimensione corretta. Ricordiamoci in fondo che FidoCadJ nasce dai trasferibili...

Già che ci siamo, scriviamo un + ed un - sul lato rame e mettiamo un condensatore ceramico in parallelo all'alimentazione. Mettiamo anche la scritta sul lato superiore. Per scrivere sul lato rame, oltre a selezionare il layer corretto, sarà necessario specchiare le scritte. Questo

Attenzione alle larghezze delle piste: quella che sembra un'autostrada sullo schermo si rivelerà nella realtà dannatamente stretta e pronta a scollarsi dalla vetroresina durante la saldatura.

2. IL DISEGNO CON FIDOCADJ

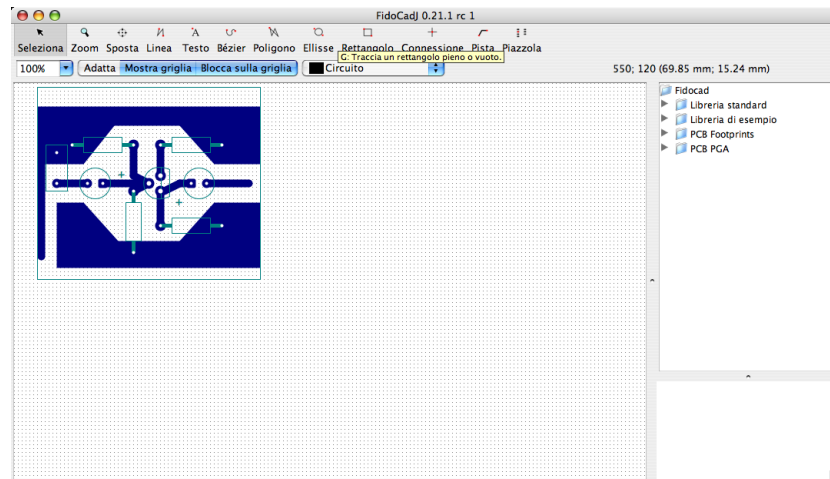


Figura 14: Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni.

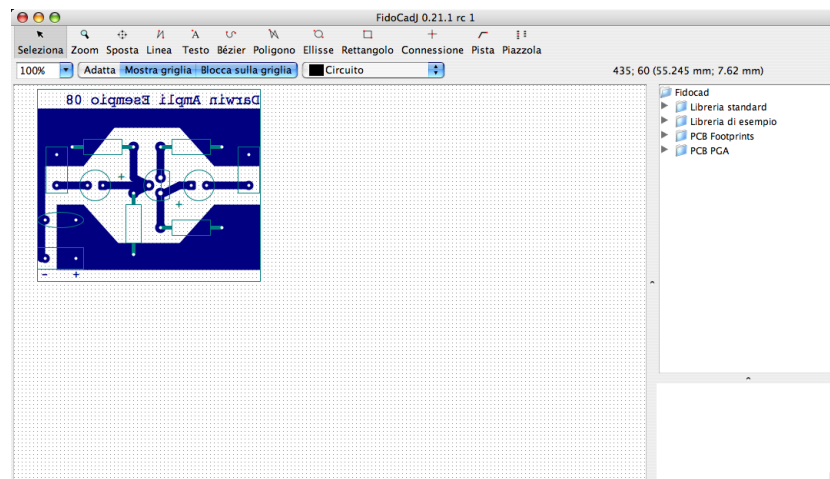


Figura 15: Ecco lo stampato quasi terminato.

lo si fa agevolmente all'interno della solita finestra delle proprietà a cui si accede facendo doppio click in modalità selezione sulla stringa da modificare. Sarà necessario fare qualche prova per ottenere le dimensioni dei caratteri corrette. Per avere un'idea, conviene mantenere un rapporto approssimativo di $3/4$ tra le dimensioni orizzontali e verticali dei caratteri. La figura 15 mostra il risultato che si è ottenuto con dimensioni del testo di 11 unità in orizzontale e 18 in verticale.

A questo punto, l'unica cosa che manca è il testo con i nomi dei componenti, che si potrà introdurre sul layer 3, dedicato alla serigrafia. Il programma con il circuito terminato è mostrato in figura 16.

Una volta concluso il lavoro, bisognerà probabilmente stampare il risultato su lucido, per poi utilizzare il bromografo, oppure metodi tipo stira e ammira. Per fare questo, bisogna dapprima rendere invisibili tutti i layer che non si vogliono stampare. Questo lo si fa dalla finestra di dialogo a cui si accede tramite il menu "Vista/Layer". Nel nostro caso, basterà rendere invisibile il layer 3, con le serigrafie. Il programma mostrerà solo il rame lato saldature.

Bisognerà quindi stampare quanto si vede sullo schermo (NON adattarlo alla pagina, ovviamente, per rispettare le dimensioni fissate)

2.5. Disegniamo un semplice circuito stampato

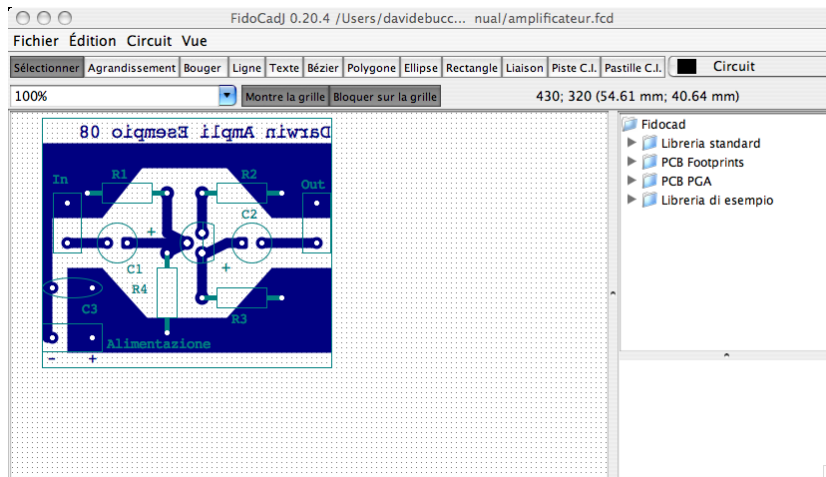


Figura 16: Il lavoro finito, con tutte le serigrafie.

e selezionando la stampa in bianco e nero per fare in modo che il programma stampi in colore nero pieno tutto il layer. Potrà essere utile specchiare il disegno, a seconda della tecnica che userete poi per fabbricare il circuito stampato. Dato che il nostro stampato è relativamente piccolo, a grandezza naturale occuperà solo un angolino di un foglio in formato standard ISO-UNI A4, come si può vedere in figura 17.

Per informazione, ecco il codice del circuito stampato che abbiamo usato negli esempi visti sopra (attenzione alle righe un po' lunghe!):

```
[FIDOCAD]
TY 320 10 18 11 0 4 1 * Darwin Ampli Esempio 08
TY 85 240 12 8 0 5 1 * +
TY 44 239 12 8 0 5 1 * -
PL 35 90 35 225 10 1
PL 55 130 95 130 10 1
PL 250 130 305 130 10 1
PL 215 130 230 130 10 1
PL 195 140 215 130 10 1
PL 115 130 175 130 10 1
MC 155 220 3 0 PCB.R01
MC 75 80 0 0 PCB.R01
MC 270 185 2 0 PCB.R01
MC 270 80 2 0 PCB.R01
MC 230 130 3 0 PCB.CE00
MC 115 130 1 0 PCB.CE00
MC 40 175 0 0 PCB.CC50
PL 190 80 190 120 10 1
PL 190 140 190 185 10 1
PL 155 80 155 120 10 1
PL 155 120 175 130 10 1
PL 155 140 175 130 10 1
PP 30 30 30 105 90 105 130 55 215 55 260 105 320 105
   320 30 1
PP 320 240 320 155 260 155 215 205 135 205 90 155 55
   155 55 240 1
MC 190 120 0 0 PCB.T092
MC 305 90 1 0 PCB.CPBX352
MC 55 90 1 0 PCB.CPBX352
MC 80 225 2 0 PCB.CPBX352
TY 290 65 12 8 0 0 3 * Out
```

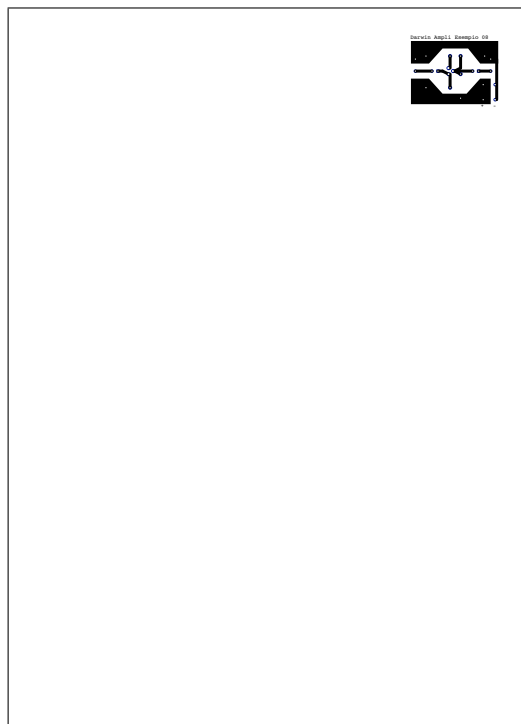



Figura 17: Il circuito stampato, così come appare stampato (specchiato) su una pagina in formato ISO-UNI A4.

```

TY 40 60 12 8 0 0 3 * In
TY 95 225 12 8 0 0 3 * Alimentazione
TY 70 190 12 8 0 0 3 * C3
TY 230 95 12 8 0 0 3 * C2
TY 115 150 12 8 0 0 3 * C1
TY 120 170 12 8 0 0 3 * R4
TY 220 200 12 8 0 0 3 * R3
TY 230 55 12 8 0 0 3 * R2
TY 100 55 12 8 0 0 3 * R1
RV 30 5 320 255 3

```

2.6 FRECCHE E STILI DI TRATTO

A partire dalla versione 0.23 di FidoCadJ, delle possibilità sono state aggiunte per quanto riguarda la possibilità di aggiungere delle frecce alle estremità dei segmenti e delle curve di Bézier. FidoCadJ offre la possibilità di specificare a quale estremità deve essere introdotta la freccia desiderata, così come lo stile con cui disegnarla. Vengono introdotti anche alcuni stili di tratto tratteggiati, per facilitare i disegni tecnici. La figura 18 mostra un esempio di uno schema elettrico in cui si è racchiuso uno stadio (un GIC) in un rettangolo tratteggiato e si è usata una freccia su un'estremità di una curva di Bézier. Facendo doppio click su quest'elemento, la finestra dei parametri che viene mostrata è visibile in figura 19. Si nota che l'opzione "Arrow at start" (freccia all'inizio) è spuntata. Il programma disegnerà una freccia orientata correttamente nel primo punto che definisce la curva. Vi sono alcuni stili di freccia e di tratto diversi nelle liste attivabili in corrispondenza delle voci "Arrow style" e "Dash style". Fate qualche prova per capire come funziona la cosa.

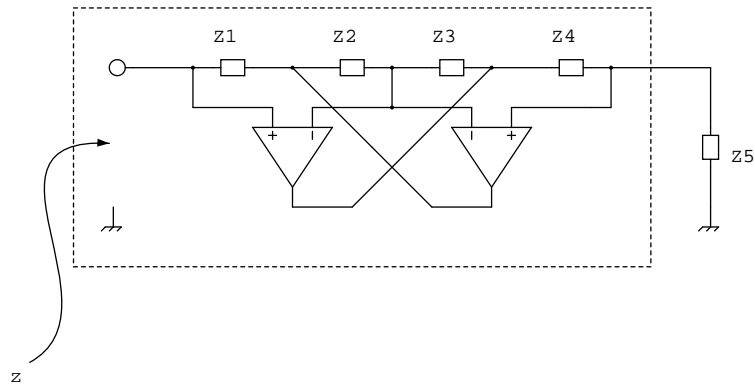


Figura 18: Uno schema elettrico di un GIC in cui sono state utilizzate alcune estensioni proprie a FidoCadJ.

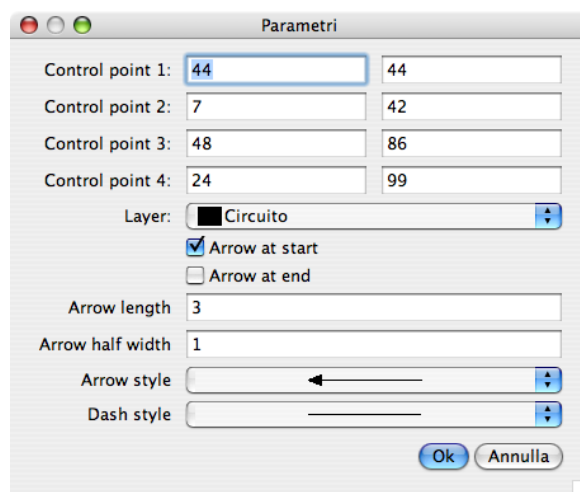


Figura 19: La finestra dei parametri di una curva di Bézier in un disegno FidoCadJ.

La possibilità di scegliere un tipo di tratto e di aggiungere delle frecce alle linee non era mai stata prevista nel formato FidoCad. Questo purtroppo condiziona la compatibilità all'indietro con il FidoCad per Windows. Quando si sceglie di utilizzare un'estensione FidoCadJ bisogna avere ben chiaro in testa cosa si sta facendo. Se si ha bisogno a tutti i costi di mantenere una compatibilità con altri utenti che utilizzano solo FidoCad e non FidoCadJ, si può attivare l'opzione "Modalità di compatibilità con FidoCad" all'interno del tab "Estensioni FidoCadJ" della finestra "Preferenze di FidoCadJ" prima di iniziare a lavorare. In questo modo, sarà impossibile introdurre elementi grafici non previsti da FidoCad e si otterranno disegni perfettamente compatibili con quest'ultimo. Per maggiori informazioni, consultate il paragrafo 3.4.

2.7 ESPORTAZIONE

Una delle cose che mi interessa maggiormente di FidoCadJ è la possibilità di creare piccoli schemi ad uso tipografico. Per questa ragione, ho cercato di permettere l'esportazione di disegni anche sotto diversi formati grafici.

Per esportare il disegno corrente, basta selezionare la voce "Esporta verso file grafico" del menu "File". La tabella 2 mostra una lista dei formati grafici attualmente disponibili per l'esportazione. Per ogni formato, viene specificato espressamente se il formato è vettoriale, oppure bitmap (all'interno della tabella 2, ma anche nella finestra di dialogo mostrata da FidoCadJ). Qualora fosse possibile, conviene prediligere sistematicamente l'utilizzo di formati vettoriali per l'esportazione, di modo da ottenere i migliori risultati possibili in tutte le situazioni.⁵

Per i formati bitmap, può essere utile attivare l'opzione "Anti aliasing", per limitare l'effetto di scalettatura che si ottiene nei bordi di linee oblique. La risoluzione utilizzata e l'opzione "Anti aliasing" non vengono utilizzate durante l'esportazione in un formato vettoriale.

L'opzione "Bianco e nero" permette di stampare ogni layer visibile in nero pieno. Questo è importante quando si debbano per esempio preparare delle pellicole ad uso tipografico, oppure da passare al bromografo.

C'è una cosa però da dire sull'esportazione PDF fatta da FidoCadJ. Il programma non può includere font all'interno del file creato. Questo vuole dire che verranno esportati in maniera identica all'originale solo i disegni che utilizzano alcuni fra i font standard tipo 1 a 14 punti:

- Courier: Font con grazie a spaziatura fissa. Vengono convertiti automaticamente in Courier anche le scritte effettuate in Courier New.
- Times: Font con grazie a spaziatura proporzionale. Vengono convertiti in Times anche le scritte in Times New Roman o Times Roman.
- Helvetica: Font senza grazie a spaziatura proporzionale. Vengono convertite anche le scritte fatte in Arial.
- Symbol: contiene simboli vari.

Tra l'altro, questi font sono praticamente di sicuro disponibili ovunque, quindi è una buona idea utilizzarli sistematicamente.

⁵ La struttura del codice di FidoCadJ permette di aggiungere abbastanza agevolmente formati vettoriali di esportazione. Se ve la sentite di partecipare al progetto, contattatemi.

*Un formato vettoriale
memorizza le
primitive di disegno.
Un formato bitmap
lavora su una matrice
di punti.*

Formato	Commento
JPG	Diffusissimo formato bitmap. Per il fatto che la compressione utilizzata è lossy, non si presta per l'esportazione di schemi come quelli di FidoCadJ.
PNG	Formato bitmap compresso, adatto all'esportazione di schemi e grafici. In mancanza di un formato vettoriale, questo è il formato migliore per esportare un disegno FidoCadJ.
SVG	Formato vettoriale standard del W3C. Alcuni browser Internet (come le versioni recenti di Safari) permettono di visualizzarlo in una pagina web. Permette di usare programmi come Inkscape per ritoccare i disegni forniti da FidoCadJ. Attualmente, ci sono alcune limitazioni per quanto riguarda l'esportazione del testo ruotato o specchiato.
EPS	Formato vettoriale Postscript incapsulato. Molto utile per chi utilizzi programmi di grafica professionale, oppure voglia includere delle figure in documenti \LaTeX . Questa è la tecnica per ottenere la figura 11, a pagina 12 (passando per una conversione in PDF, dato che utilizzo PDF \LaTeX).
PGF	Formato vettoriale da utilizzare direttamente all'interno di un file \LaTeX , che utilizzi il pacchetto <i>pgf</i> , disponibile nell'archivio CTAN. Questa modalità di esportazione fornisce uno script abbastanza interpretabile e modificabile a mano. Questo permette di introdurre codice \LaTeX direttamente all'interno del disegno ed è la tecnica adottata per ottenere la figura 5, a pagina 7.
SCR	FidoCadJ, a partire dalla versione 0.21, permette di esportare un disegno verso uno script per CadSoft Eagle. Per utilizzare questa possibilità, bisogna installare nella directory <code>1br</code> dell'installazione di Eagle la libreria <code>FidoCadJLIB.1br</code> , scaricabile dal sito di FidoCadJ. Attualmente, l'esportazione è possibile solo per i simboli elettrici più comuni. Non sono esportati le piazzole e le piste, che saranno dunque assenti nello script Eagle.
PDF	Il celebre formato vettoriale Portable Document Format, di Adobe. Vedi il testo per alcune limitazioni sull'uso dei font.

Tabella 2: Lista dei formati di esportazione disponibili con FidoCadJ.

2.8 OPZIONI LINEA DI COMANDO

Il programma viene distribuito sotto forma di file `.jar`, ovvero un archivio Java.⁶ In molti sistemi operativi, può essere sufficiente fare doppio click sul file per lanciare il programma, a patto di avere una versione recente di Java installata sulla macchina. Nella terminologia Sun, quello che serve è il cosiddetto JRE, ovvero il Java Runtime Environment, che è tutto quello che ci vuole per far girare un programma scritto in Java (ma non per scriverlo: per quello ci vuole l'SDK...). La versione minima di Java necessaria per utilizzare FidoCadJ è la 1.4, che è in giro ormai da diversi anni.

In qualche caso, potrà essere utile far partire FidoCadJ dalla linea di comando (il terminale dei sistemi Unix, oppure il Prompt MS-DOS per quelli Windows). Per quello, basta utilizzare il comando `java`, con l'opzione `-jar`:

```
java -jar fidocadj.jar
```

Se un file è specificato nella linea di comando, il programma tenta di aprirlo. Per esempio (su un sistema Unix):

```
java -jar fidocadj.jar ~/FidoCadJ/test.fcd
```

FidoCadJ verrà avviato e cercherà di aprire il file `~/FidoCadJ/test.fcd` (semprech  questi esista).

Un'altra possibilit  interessante, sebbene a stretto rigore sia pi  una caratteristica di Java che di FidoCadJ,   la possibilit  di modificare l'aspetto del programma (che in gergo Java si chiama *look & feel*).

Potete giocare comunque con il tipo di aspetto che preferite da linea di comando, senza modificare di una virgola il codice. Ecco qualcosa che alcuni utilizzatori Linux apprezzeranno, il look GTK+:

```
java -Dswing.defaultlaf=com.sun.java.swing.plaf.gtk.
GTKLookAndFeel -jar fidocadj.jar
```

Oppure, il classico e non privo di fascino look & feel Motif, che   mostrato in figura 20:

```
java -Dswing.defaultlaf=com.sun.java.swing.plaf.motif.
MotifLookAndFeel -jar fidocadj.jar
```

Ovviamente, i comandi sopra elencati vanno usati da terminale, trovandosi nella stessa directory in cui si trova il file `fidocadj.jar` e scrivendo tutto di seguito sulla stessa riga.

2.9 GESTIONE LIBRERIE

Il programma permette di specificare una directory all'interno della quale sono caricati tutti i file di libreria (estensione `.fcl`). Per farlo, basta andare nel menu "Vista/Opzioni" e specificarla nella riga apposita. Se   presente un file di nome `FCDstdlib.fcl`, i suoi contenuti si sostituiscono alla libreria standard disponibile nel programma. Se   presente un file di nome `PCB.fcl`, i suoi contenuti si sostituiscono alla libreria PCB disponibile nel programma.⁷ A partire dalla versione 0.23, grazie all'accordo di Roby IZ1CYN, ho potuto includere la libreria IHRaM 2.1

⁶ Salvo nella versione per Macintosh, per la quale vi   un'applicazione a s  stante.

⁷ Attenzione all'utilizzo delle lettere maiuscole, in particolare se il vostro sistema operativo le differenzia dalle minuscole nella gestione dei file.

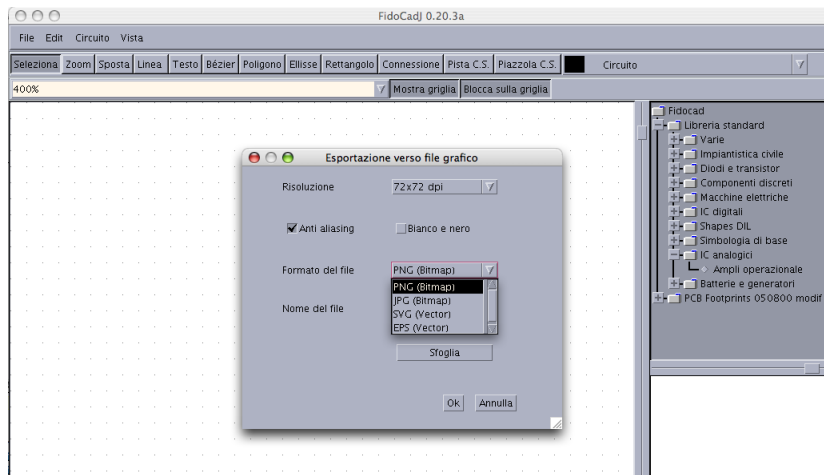


Figura 20: L'aspetto del programma sotto MacOSX, utilizzando il look & feel Motif.

direttamente nel pacchetto contenente FidoCadJ. Questo perché fra le varie librerie messe a punto dagli utilizzatori di FidoCad e di FidoCadJ mi è parsa la migliore, la più completa e razionale. Se comunque all'interno della directory esterna contenente le librerie è disponibile un file chiamato IHRAM.FCL, questo verrà caricato al posto della versione contenuta in FidoCadJ.

Altri file con estensione .fcl sono considerati alla stregua di librerie aggiuntive e caricati in memoria oltre alla libreria standard. Tutto questo è fatto solo all'avvio, quindi per far prendere in conto delle modifiche, il programma va riavviato.

Dalla versione 0.23, FidoCadJ permette (come del resto il FidoCad originale) di suddividere le macro non standard, per permettere anche a chi non avesse la libreria in questione installata di vedere lo stesso il disegno. Per attivare quest'opzione, basta andare nelle preferenze del programma ed attivare la voce corrispondente nel tab "Estensioni di FidoCadJ", a seconda che si voglia suddividere le macro durante un salvataggio di un file, oppure durante il copia/incolla. Quest'ultima possibilità può essere utile per esempio quando si mostrano i disegni su un forum o su un gruppo di discussione riportandovi il codice.

FidoCad aveva l'opzione "Splitta le macro non standard". FidoCadJ può fare esattamente lo stesso, ma in italiano.

3

IL FORMATO DISEGNI, MACRO E LIBRERIE FIDOCADJ

In questo capitolo, sarà descritto in dettaglio il formato utilizzato da FidoCad e di conseguenza da FidoCadJ per memorizzare i disegni. Si tratta di un semplice formato testuale che ha il pregio di essere molto compatto ed efficiente. Dato che il formato non è mai stato descritto dettagliatamente in un documento esaustivo, cerco qui di riassumere tutto quello che ho imparato durante lo studio che ho compiuto. Nel corso dello sviluppo di FidoCadJ, mi sono trovato a dover aggiungere dei dettagli al formato originale. Descriverò qui anche le mie aggiunte.

3.1 DESCRIZIONE DELL'INTESTAZIONE

Tutti i file contenenti un disegno in formato FidoCad devono iniziare con il tag `[FIDOCAD]`. Un programma può quindi riconoscere la presenza di comandi FidoCad riconoscendo quindi questo tag. A questo proposito, FidoCadJ è più tollerante del FidoCad originale e riconosce ed interpreta correttamente un file privo dell'intestazione standard. Anche comandi frammisti a testo vengono comunque riconosciuti ed interpretati correttamente, a meno che il numero di righe consecutive non corrette non superi un valore fissato internamente al programma (intorno al centinaio). Ciò evita che FidoCadJ macini inutilmente per diversi minuti per esempio tentando di aprire un file binario di grandi dimensioni.

3.2 IL SISTEMA DI COORDINATE

FidoCadJ lavora con un sistema di coordinate molto semplice. In pratica, si ha a disposizione un'area molto grande, identificata però da delle coordinate intere e positive. La lunghezza di ogni unità in x ed in y è fissata a 127 μm , valore che permette di ottenere una buona risoluzione per i package SMD più piccoli, senza tuttavia essere troppo fine per gli utilizzi di tutti i giorni.

Il FidoCad originale prevedeva due modalità diverse: PCB e schema elettrico. In FidoCadJ questa differenza è attenuata e compare solamente al momento di stampare un disegno: sarà infatti opportuno indicare al programma di ridimensionare uno schema elettrico per occupare al meglio la dimensione del foglio, altrimenti ne verrà fuori un francobollo.

3.3 PRIMITIVE DI DISEGNO

In FidoCadJ, esistono 11 primitive di disegno e sono le seguenti:

- Linea
- Rettangolo pieno o vuoto
- Testo semplice (obsoleta)
- Testo avanzato

a	Freccia
0	nessuna
1	all'inizio
2	alla fine
3	entrambe le estremità

Tabella 3: Presenza delle frecce sulle estremità di un segmento o di una curva di Bézier, a seconda del codice a.

b	Stile della freccia
0	freccia piena normale
1	freccia piena con trattino
2	freccia vuota
3	freccia vuota con trattino

Tabella 4: Stile delle frecce, a seconda del codice b.

- Poligono pieno o vuoto
- Ellisse piena o vuota
- Curva di Bézier
- Connessione elettrica
- Piazzola per circuito stampato
- Pista per circuito stampato
- Macro

Procederemo ad analizzarne il formato una per una. In generale, ogni primitiva è identificata da un comando e da una serie di parametri (di solito numeri interi, oppure stringhe testuali) che si trovano sulla stessa riga, separati da uno spazio.

Linea

La primitiva linea, è identificata dal comando **LI** e richiede per la sua definizione solamente le coordinate iniziali, finali ed il layer:

```
LI x1 y1 x2 y2 l
```

il punto (x_1, y_1) rappresenta le coordinate iniziali, (x_2, y_2) quelle finali e l è il layer, indicato con un numero da 0 a 15.

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, **LI** può essere seguito nella riga successiva da un'estensione:

```
FCJ a b c d e
```

dove a è un intero rappresentante la presenza o l'assenza di frecce agli estremi del segmento (vedere la tabella 3), b è un intero identificante lo stile della freccia da utilizzare (vedere la tabella 4). I parametri c e d forniscono rispettivamente la lunghezza totale e la semilarghezza della freccia da disegnare, mentre e è un intero che fornisce lo stile del tratteggio.

*I matematici
troverebbero
probabilmente più
appropriato il termine
"segmento".*

Rettangolo pieno o vuoto

Un rettangolo pieno o vuoto è indicato rispettivamente dai comandi **RP** e **RV** seguiti dalle coordinate di due vertici sulla diagonale e dal layer.

```
RP x1 y1 x2 y2 l
RV x1 y1 x2 y2 l
```

il punto (x_1, y_1) rappresenta il primo vertice sulla diagonale, (x_2, y_2) il secondo vertice, e l è il layer, indicato con un numero da 0 a 15.

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, **RP** e **RV** possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e
```

dove e è un intero che fornisce lo stile del tratteggio.

Testo semplice (obsoleta)

Il testo semplice è stata la prima primitiva di testo messa a disposizione dalle prime versioni di FidoCad. FidoCadJ la riconosce e scrive semplicemente il testo in corpo 12, qualunque sia il livello di zoom utilizzato.

Dato che questa primitiva è stata considerata obsoleta da Lorenzo Lutti, il creatore di FidoCad, FidoCadJ fa esattamente lo stesso e, seppure questa venga interpretata correttamente, non è presente nella barra delle primitive. FidoCadJ memorizzerà questo elemento esattamente come se fosse un testo avanzato e fra l'altro utilizzerà il comando **TY** nel momento di salvare il file.

Il comando è **TE** ed il formato è il seguente:

```
TE x1 y1 testo da scrivere
```

il punto (x_1, y_1) è dove la stringa "testo da scrivere" verrà posizionata. Si noti come l'indicazione del layer è assente. FidoCadJ tratterà quest'oggetto come se fosse posizionato sul layer zero (circuito).

Testo avanzato

La primitiva testo avanzato permette una flessibilità molto maggiore rispetto alla primitiva testo semplice presentata precedentemente.

Essa è identificata dal comando **TY**, seguito da svariati parametri, atti a determinare l'orientamento del testo (testo ruotato o specchiato), nonché le dimensioni in x e y del font utilizzato. Data la quantità di informazioni da fornire, la stringa di comando è abbastanza articolata:

```
TY x1 y1 sy sx a s l f testo da scrivere
```

il punto (x_1, y_1) è dove la stringa "testo da scrivere" verrà posizionata. Il valore di s_y e s_x indica la dimensione verticale ed orizzontale del testo in unità logiche. Ho scelto di fare in modo che FidoCadJ rispetti la dimensione verticale del testo a partire da quella orizzontale, e che deformi il font solo se strettamente necessario. La rotazione del testo è resa possibile dal termine a , espresso in gradi sessagesimali, mentre il valore di s determina lo stile del testo, secondo la tabella 5. Il layer è rappresentato dal solito termine l , mentre f indica il font da utilizzare, oppure è un asterisco, per indicare l'utilizzo del font standard Courier New (con grazie, a spaziatura fissa). Qualora il nome del font avesse degli spazi, questi vengono rimpiazzati dal simbolo $+$.

Bit	Peso	Funzione
0	1	Testo in neretto
2	4	Testo specchiato

Tabella 5: Funzione dei bit nello stile del testo.

La lunghezza massima del testo è di una ottantina di parole. Il conto è fatto in parole e non in caratteri perché nella struttura interna del programma le parole vengono contate separatamente al momento di interpretare la linea.

Poligono pieno o vuoto

Un poligono pieno o vuoto è indicato rispettivamente dai comandi **PP** e **PV**, seguiti dalle coordinate dei vertici che definiscono il poligono e dal layer.

```
PP x1 y1 x2 y2 ... l
PV x1 y1 x2 y2 ... l
```

i punti (x_1, y_1) , (x_2, y_2) ... sono i vertici che definiscono il poligono e l è il layer, indicato con un numero da 0 a 15. La lunghezza della linea può quindi variare a seconda del numero di vertici presenti. Il numero massimo di vertici disponibili è fissato arbitrariamente a 20, per evitare di avere a che fare con linee troppo lunghe.

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, **PP** e **PV** possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e
```

dove e è un intero che fornisce lo stile del tratteggio.

Ellisse piena o vuota

Un'ellisse piena o vuota è indicata rispettivamente dai comandi **EP** e **EV**, seguiti dalle coordinate di due vertici sulla diagonale e dal layer.

```
EP x1 y1 x2 y2 l
EV x1 y1 x2 y2 l
```

il punto (x_1, y_1) rappresenta il primo vertice sulla diagonale, (x_2, y_2) il secondo vertice, e l è il layer, indicato con un numero da 0 a 15.

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, **EP** e **EV** possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e
```

dove e è un intero che fornisce lo stile del tratteggio.

Curva di Bézier

Una curva di Bézier, nella sua variante cubica è identificata da quattro vertici, che vengono quindi richiesti dal comando **BE**:

```
BE x1 y1 x2 y2 x3 y3 x4 y4 l
```

i punti $P_1 \equiv (x_1, y_1)$, $P_2 \equiv (x_2, y_2)$, $P_3 \equiv (x_3, y_3)$ e $P_4 \equiv (x_4, y_4)$ sono i quattro punti di controllo della curva di Bézier, mentre l è il layer,

indicato con un numero da 0 a 15. Dati i quattro punti sopra definiti, la curva viene calcolata con la combinazione:

$$B(t) = (1-t)^3P_1 + 3t(1-t)^2P_2 + 3t^2(1-t)P_3 + t^3P_4 \quad (3.1)$$

dove $t \in [0, 1]$ è un parametro.

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, [LI](#) può essere seguito nella riga successiva da un'estensione:

```
FCJ a b c d e
```

dove a è un intero rappresentante la presenza o l'assenza di frecce agli estremi del segmento (vedere la tabella 3), b è un intero identificante lo stile della freccia da utilizzare (vedere la tabella 4). I parametri c e d forniscono rispettivamente la lunghezza totale e la semilarghezza della freccia da disegnare, mentre e è un intero che fornisce lo stile del tratteggio.

Connessione elettrica

La primitiva connessione elettrica è semplicemente un circoletto pieno di dimensione costante e va utilizzata per rappresentare una connessione in uno schema elettrico. È identificata dal comando [SA](#) e richiede unicamente le coordinate ed il layer:

```
SA x1 y1 l
```

Con FidoCadJ, il diametro del circoletto è fissato internamente al programma a due unità logiche.

Piazzola per circuito stampato

Una piazzola per circuito stampato è identificata dal comando [PA](#) ed è caratterizzata dal suo stile (rotondo, rettangolare, rettangolare con spigoli arrotondati) e dalla dimensione del foro interno:

```
PA x1 y1 dx dy si st l
```

Il punto (x_1, y_1) rappresenta la posizione della piazzola, d_x è la larghezza secondo x della piazzola, d_y l'altezza lungo y. Il valore di s_i è il diametro interno del foro e s_t è lo stile della piazzola:

- 0 piazzola ovale
- 1 piazzola rettangolare
- 2 piazzola rettangolare con spigoli arrotondati

Il valore di l indica invece il layer all'interno del quale si trova la piazzola.

Pista per circuito stampato

La pista di circuito stampato è sostanzialmente un segmento, di cui si può specificare la larghezza. Gli estremi del segmento sono sempre arrotondati, per semplificare la connessione con altre piste del circuito stampato. Il comando da utilizzare è [PL](#), con il formato seguente:

```
PL x1 y1 x2 y2 di l
```

La pista viene tracciata tra i punti (x_1, y_1) e (x_2, y_2) , con spessore totale d_i . Il layer utilizzato è l.

Chiamata di una macro

Una macro è un disegno o un simbolo presente all'interno di una libreria. Tipicamente, i simboli elettrici più frequenti sono rappresentati in questa maniera. Il comando per realizzare una chiamata di una macro è **MC**, e la chiamata avviene nella maniera seguente:

```
MC x1 y1 o m n
```

La macro viene disegnata nel punto (x_1, y_1) , l'orientazione è definito dal valore di *o* (moltiplicato per 90°) e se *m* è uguale a 1, la macro viene specchiata. L'ultimo parametro, *n* è il nome della macro all'interno di una libreria, specificata come libreria.codice.

L'utilizzo è il seguente:

```
[FIDOCAD]
MC 40 30 0 0 080
FCJ
TY 50 35 4 3 0 0 0 * R1
TY 50 40 4 3 0 0 0 * 47k
```

Estensione FidoCadJ: a partire dalla versione 0.23 di FidoCadJ, **LI** può essere seguito nella riga successiva da un'estensione: La presenza del comando **FCJ** indica che la definizione della macro non è terminata, ma che i campi indicanti il nome ed il valore sono specificati dai due comandi **TY** nelle linee che seguono immediatamente **FCJ**. Attualmente, solo le coordinate dei due comandi **TY** sono utilizzate.

Questo modo di procedere ha il vantaggio che se un file contenente un'estensione di questo tipo viene letto da FidoCad, quest'ultimo ignorerà le linee contenenti **FCJ** (emettendo un messaggio di errore), ma il disegno ottenuto sarà in qualche modo simile a quello che appare su FidoCadJ (vedere il paragrafo 3.4).

3.4 ESTENSIONI DI FIDOCADJ

A partire dalla versione 0.21, FidoCadJ ha introdotto delle estensioni al formato FidoCad originale, che nel codice sono identificate dal comando **FCJ**. La presenza di un comando **FCJ** indica in generale che il comando precedente non è terminato, ma che ulteriori informazioni restano da prendere in conto.

FidoCadJ dispone di una modalità "compatibilità con FidoCad" all'interno della quale verranno disabilitate le estensioni, di modo da risultare perfettamente compatibile con il FidoCad originale. In caso contrario, FidoCad leggerà ancora (fornendo un messaggio di errore) i file prodotti da FidoCadJ, ma le informazioni aggiuntive di cui fa uso FidoCadJ non saranno prese in conto nei disegni. Il risultato dipenderà dal contesto: una riga tratteggiata apparirà continua, le frecce eventualmente presenti non saranno disegnate. Il testo associato al nome ed al valore di una macro sarà invece mostrato correttamente. La figura 21 dovrebbe riassumere quello che si ottiene fornendo a FidoCad il codice usato per disegnare la figura 18. FidoCad protesterà un po' per via dei comandi che non comprenderà, mancano dei dettagli (il tratteggio e la freccia), ma in linea di massima il disegno è ancora utilizzabile.

3.5 TOLLERANZA AGLI ERRORI SINTATTICI

FidoCadJ è progettato per tollerare eventuali errori o comandi malformati che vengano forniti al programma. Ovviamente, a meno che non

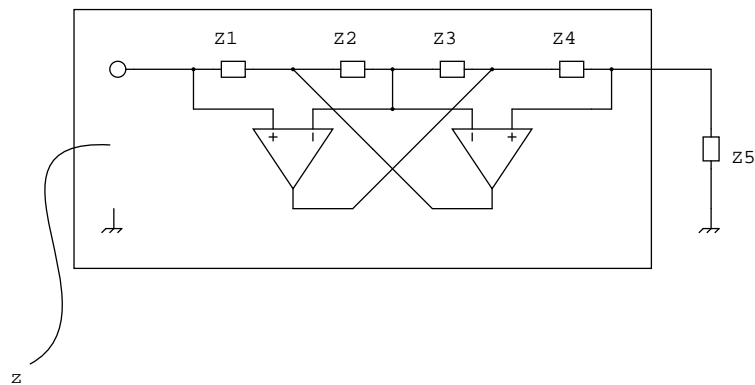


Figura 21: La figura 18 come probabilmente apparirebbe se letta con FidoCad.

abbiate una sfera di cristallo collegata come periferica USB, il programma non potrà correggere gli errori e si limiterà a saltare (ed eliminare) tutte le linee che li contengono.

Un'eccezione a questo comportamento è dato dal fatto che, per ragioni di compatibilità con le prime versioni di FidoCad, alcune primitive possono essere specificate omettendo il layer (verranno in quel caso considerate parte del layer zero, dedicato ai circuiti).

3.6 IL FORMATO DELLE LIBRERIE

La struttura di un file di libreria è molto semplice:

```
[FIDOLIB Libreria standard]
{Simbologia di base}
[000 Terminale]
LI 100 100 102 100
EV 102 98 106 102
[010 Terminale +]
LI 100 100 102 100
EV 102 98 106 102
LI 103 100 105 100
LI 104 99 104 101
[020 Terminale -]
LI 100 100 102 100
EV 102 98 106 102
LI 103 100 105 100
...
```

La prima linea indica fra parentesi quadre il nome della libreria (preceduto da FIDOLIB). La seconda linea fornisce, fra parentesi graffe, la categoria della libreria all'interno della quale riunire le macro, fornite successivamente.

Ogni macro è formata da un'intestazione (fra parentesi quadre) ed una sequenza di comandi. L'intestazione è composta dal codice (che DEVE essere unico all'interno della libreria) e dalla descrizione. Il codice sarà utilizzato all'interno del codice FidoCad, mentre la descrizione serve solo all'utente per capire di che macro si tratta. I comandi non sono altro che dei disegni FidoCad, attorno al punto (100,100), utilizzato come origine. Questo punto sarà poi quello che verrà rimappato

nel momento in cui la macro verrà introdotta. La macro verrà poi richiamata con la primitiva MC, con il codice "libreria.macro".

Nulla impedisce di richiamare una macro all'interno di un'altra macro. Quello che bisogna evitare, ovviamente, è la ricorsione, ovvero di scrivere una macro che richiama sé stessa.

3.7 LIBRERIE STANDARD

FidoCadJ contiene al suo interno alcune librerie tradizionalmente fornite con FidoCad. In particolare, si tratta della libreria standard e della libreria dedicata ai simboli per i circuiti stampati (PCB).

È tuttavia possibile modificare il sostituire il contenuto delle librerie interne specificando al programma (menu "Vista/Opzioni/Directory Librerie") una directory contenente le librerie da caricare. Se un file di nome `FCDstdlib.fcl` è presente in questa directory, il suo contenuto sarà utilizzato al posto della libreria standard. Se un file di nome `PCB.fcl` è presente, il suo contenuto verrà utilizzato al posto della libreria contenente i simboli da usare nei circuiti stampati. Librerie contenute in file con un altro nome e con estensione `fcl` saranno caricate a fianco delle librerie standard.

4

CONCLUSIONI

In questo manuale, si è visto come utilizzare FidoCadJ per disegnare uno schema elettrico oppure un semplice circuito stampato. A questo punto, il lettore dovrebbe possedere tutti gli elementi necessari per utilizzare proficuamente FidoCadJ per le proprie esigenze.

Non bisogna credere che FidoCadJ sia uno strumento dedicato solo all'elettronica. Preparando librerie specifiche, diventa possibile creare quasi ogni tipo di disegno bidimensionale e può essere utile in molte situazioni.

Il vantaggio di un programma libero è quello di essere a disposizione di una comunità. Per questo, è molto importante che gli utilizzatori si facciano sentire (se non altro per capire se il progetto merita di essere continuato, ed in che direzione). Non esitate quindi a contattarmi (davbucci@tiscali.it, ma non mandate allegati di nessun tipo!).

INFORMAZIONI SPECIFICHE PER PIATTAFORMA

A.1 MACOSX

A.1.1 Estensioni

Una delle critiche più diffuse alle prime versioni di FidoCadJ da parte degli utilizzatori Macintosh (come me, del resto) consisteva nella cattiva integrazione del programma sotto MacOSX. A partire dalla versione 0.21.1, FidoCadJ fa degli sforzi specifici per integrarsi all'interno dell'aspetto e del funzionamento delle applicazioni native. Per questa ragione, alcuni dettagli del funzionamento del programma sono leggermente diversi quando FidoCadJ si rende conto di essere eseguito su una piattaforma Apple:

- FidoCadJ utilizza di default il look and feel Quaqua ¹ quando lo si utilizza a partire dall'applicazione completa (FidoCadJ.app e non il file fidocadj.jar). Dato che Quaqua potrebbe rallentare le prestazioni su macchine non recentissime, vi è la possibilità di disattivarlo attraverso le impostazioni del menu Preferenze.
- La barra dei menu è mostrata al suo posto, ovvero nella parte alta dello schermo.
- Le voci "Preferenze" e "Informazioni su FidoCadJ" si trovano al loro posto, ovvero all'interno del menu FidoCadJ.
- Il programma dichiara al sistema operativo la sua disponibilità ad aprire file di tipo .fcd. Ad essi viene pure associata un'icona che richiama il simbolo del programma e che dovrebbe essere abbastanza evocativa.

A.1.2 Come scaricare ed eseguire FidoCadJ con MacOSX

FidoCadJ può funzionare con una versione di MacOSX superiore o eguale alla 10.3.9 (Panther). Il motivo è presto detto: FidoCadJ richiede per funzionare almeno la versione 1.4 di Java, che Apple fornisce con il suo sistema operativo.

Sebbene sia possibile utilizzare il file `fidocadj.jar` come sotto gli altri sistemi operativi, con MacOSX conviene utilizzare l'applicazione preparata specificatamente per questo sistema operativo. Tutto avviene come con le applicazioni native: basta scaricare l'immagine disco contenente il programma all'indirizzo: <http://davbucci.cher-alice.fr/elettronica/fidocadj/FidoCadJ.dmg>, aprire l'immagine disco e spostare FidoCadJ.app all'interno della cartella Applicazioni da cui sarà poi disponibile.

Per disinstallare FidoCadJ, sarà sufficiente prelevare FidoCadJ.app dalla cartella Applicazioni e spostarlo nel cestino.

¹ <http://www.randelshofer.ch/quaqua/>

A. INFORMAZIONI SPECIFICHE PER PIATTAFORMA

A.2 LINUX

A.2.1 Estensioni

Non sono state attualmente implementate estensioni dedicate esclusivamente a questa piattaforma.

A.2.2 Come scaricare ed eseguire FidoCadJ su un sistema Linux

di Roby Pozzato IZ1CYN

Presupposto: avere installato il JRE 6 di Sun e/o OpenJDK 6 JRE (o versioni precedenti compatibili con le specifiche del programma). Nel paragrafo [A.2.3](#) descriveremo come installare il programma adoperando esclusivamente comandi da terminale. Nel paragrafo [A.2.4](#), ci baseremo invece sull'interazione tramite un sistema grafico.

A.2.3 Su qualunque sistema, da terminale

Scarichiamo il programma utilizzando il comando `wget`:

```
$ wget http://davbucci.chez-alice.fr/elettronica/fidocadj/fidocadj.jar
--23:51:22-- http://davbucci.chez-alice.fr/elettronica/fidocadj/fidocadj.jar
=> 'fidocadj.jar'
Risoluzione di davbucci.chez-alice.fr in corso... 212.27.63.127
Connessione a davbucci.chez-alice.fr [212.27.63.127:80]... connesso.
HTTP richiesta inviata, aspetto la risposta... 200 OK
Lunghezza: 276,694 (270K) [application/x-java-archive]

100%[=====] 276,694 71.75K/s ETA 00:00
23:51:26 (71.59 KB/s) - "fidocadj.jar" salvato [276694/276694]
```

Creiamo una directory (prima diventiamo root con `su` o `sudo -s`)

```
# mkdir /usr/bin/fidocadj
```

... e ci spostiamo il file scaricato (sostituire <user> con l'utente che ha scaricato il file):

```
# mv /home/<user>/fidocadj.jar /usr/bin/fidocadj
```

Rendiamo il file eseguibile:

```
# chmod +x /usr/bin/fidocadj/fidocadj.jar
```

Non dimentichiamo di tornare utenti normali!

```
# exit
```

Ed ora possiamo eseguire il programma:

```
$ /usr/bin/fidocadj/fidocadj.jar
```

A.2.4 Su un sistema grafico

Nell'esempio è una Ubuntu 8.04, ma per versioni precedenti o successive, o altri sistemi, non cambiano i concetti:

- Scarichiamo il file dal browser, o con Gwget, o simili)
- Lanciamo il nostro File Manager (Nautilus, Konqueror, ecc.) come root (se non c'è un comando apposito nel menu basta lanciarlo da terminale con `sudo nautilus`), creiamo la directory `/usr/bin/fidocadj` e spostiamoci il file appena scaricato (sarà in `/home/<user>/`)

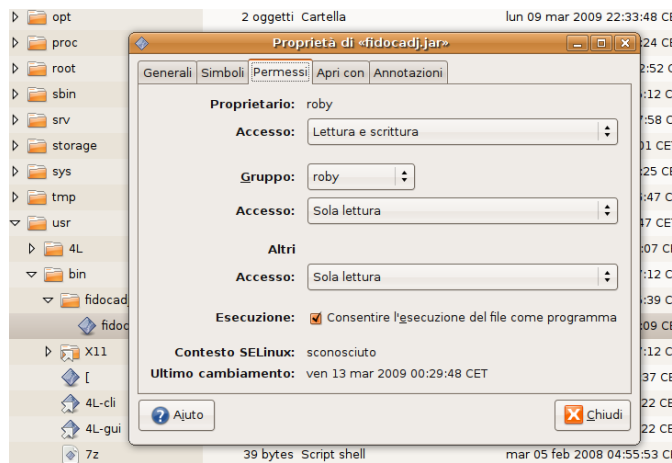


Figura 22: La finestra di impostazione dei permessi, con la distribuzione Ubuntu 8.04.

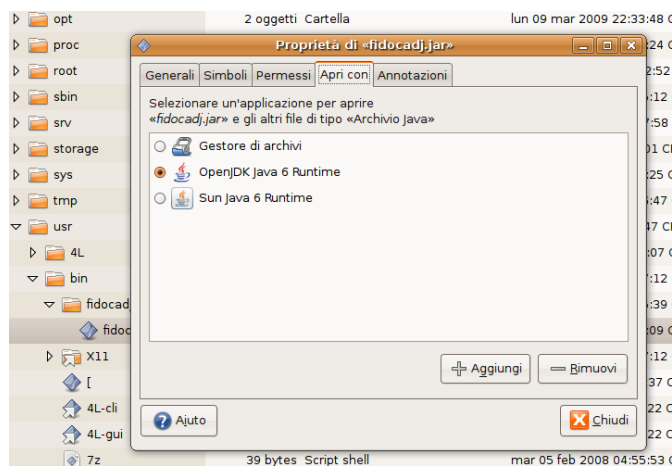


Figura 23: Impostare l'esecuzione con la macchina virtuale Java nella distribuzione Ubuntu 8.04.

- Click destro sul file, nella finestra selezioniamo il tab “Permessi”, aggiungiamo un segno di spunta a fianco della dicitura “Consentire l'esecuzione del file come programma”, come mostrato nella figura 22
- Selezioniamo la scheda “Apri con” e scegliamo “OpenJDK Java 6 Runtime” oppure “Sun Java 6 Runtime”, come visibile nella figura 23.²
- Click su “Chiudi”, e siamo pronti ad eseguire FidoCadJ: un doppio click sull'eseguibile, oppure lo aggiungiamo al menu; il comando da inserire è semplicemente `/usr/bin/fidocadj/fidocadj.jar`.

A.3 WINDOWS

A partire dalla versione 0.23, se FidoCadJ viene fatto girare su una piattaforma Windows, viene utilizzato di default il Look and Feel

² N.D.R. Mi è stato segnalato che utilizzando il runtime OpenJDK si possono incontrare dei problemi nella stampa di disegni su stampanti adoperanti driver CUPS. In caso di problemi, fate qualche prova con Sun JRE 1.4.2, se disponibile.

“Windows”.

INDICE ANALITICO

- [+](#), [24](#)
- [A4](#), [15](#)
- [Arial](#), [18](#)
- [asterisco](#), [24](#)
- [autoplacer](#), [12](#)
- [autorouter](#), [12](#)
- [Bézier](#), [6](#), [16](#), [23](#), [25](#)
- [barra degli strumenti](#), [8](#)
- [barra dei comandi](#), [4](#)
- [BE](#), [25](#)
- [bromografo](#), [14](#)
- [CadSoft](#), [19](#)
- [circuito](#), [24](#)
- [circuito stampato](#), [11](#), [12](#), [14](#), [15](#)
 - [footprint](#), [4](#)
 - [PCB](#), [22](#)
 - [SMD](#), [22](#)
- [Cocoa](#), [2](#)
- [codice](#), [10](#), [11](#)
- [colore](#), [11](#)
- [compressione lossy](#), [19](#)
- [connessione](#), [6](#), [23](#), [26](#)
- [Courier](#), [18](#)
- [Courier New](#), [24](#)
- [dimensioni dei caratteri](#), [14](#)
- [disegno vettoriale](#), [4](#)
- [Eagle](#), [v](#), [19](#)
- [ellisse](#), [6](#), [23](#), [25](#)
- [EP](#), [25](#)
- [EPS](#), [1](#), [19](#)
- [esportazione](#), [18](#)
- [estensione FidoCadJ](#), [10](#), [27](#)
- [estensioni](#), [2](#)
- [EV](#), [25](#)
- [FCJ](#), [27](#)
- [FidoCad](#), [1](#), [2](#), [4](#), [10](#), [18](#), [21](#), [22](#), [24](#), [27–29](#)
- [FidoCadJ](#), [22](#)
- [FidoCadJ.app](#), [31](#)
- [fidocadj.jar](#), [31](#)
- [FIDOLIB](#), [28](#)
- [FidoReadJ](#), [1](#), [2](#)
- [font](#), [18](#), [19](#), [24](#)
- [formato bitmap](#), [18](#)
- [formato vettoriale](#), [18](#)
- [forum](#), [10](#)
- [francese](#), [12](#)
- [freccia](#), [16](#)
- [GIC](#), [16](#)
- [griglia](#), [11](#)
- [GTK+](#), [20](#)
- [Helvetica](#), [18](#)
- [incrocio piste](#), [12](#)
- [inglese](#), [12](#)
- [Inkscape](#), [19](#)
- [interprete](#), [2](#)
- [intestazione](#), [22](#), [28](#)
- [it.hobby.elettronica](#), [v](#), [2](#)
- [it.hobby.fai-da-te](#), [2](#)
- [italiano](#), [12](#)
- [jar](#), [20](#)
- [Java](#), [1](#), [2](#), [20](#)
- [JPG](#), [19](#)
- [JRE](#), [20](#), [32](#)
- [L^AT_EX](#), [19](#)
- [layer](#), [11](#), [13](#), [28](#)
- [LI](#), [23](#), [26](#), [27](#)
- [libreria](#), [1](#), [20](#)
 - [libreria standard](#), [20](#)
- [libreria PCB](#), [29](#)
- [libreria standard](#), [8](#), [29](#)
- [linea](#), [6](#), [22](#), [23](#)
- [linea di comando](#), [20](#)
- [Linux](#), [1](#), [20](#)
- [linux](#), [v](#)
- [look & feel](#), [20](#)
- [Lorenzo Lutti](#), [1](#), [24](#)
- [lunghezza massima testo](#), [25](#)
- [Macintosh](#), [5](#), [20](#)
- [MacOSX](#), [v](#), [2](#), [4](#), [5](#), [21](#)
- [macro](#), [1](#), [4](#), [8](#), [23](#), [27](#), [28](#)
- [MC](#), [27](#), [29](#)
- [Metal](#), [4](#), [5](#)
- [Motif](#), [20](#)
- [nazionalizzazione](#), [12](#)
- [newsgroup](#), [v](#), [1](#), [10](#)
- [numero massimo di vertici](#), [25](#)
- [obsoleta](#), [24](#)
- [OpenJDK](#), [32](#)
- [PA](#), [26](#)

- parametri, 23
- parentesi quadre, 28
- PCB, v
- PCB line, 13
- PDF, 1, 19
- PDFLATEX, 19
- PGF, 19
- piazzola c. s., 6
- piazzola per c. s., 23, 26
- pista c. s., 6
- pista per c. s., 23, 26
- PL, 26
- PNG, 19
- poligono, 6, 12–14, 23, 25
- posta elettronica, 10
- Postscript, 19
- PP, 25
- primitiva, 2, 4, 23
- programmazione ad oggetti, 2
- prompt MS-DOS, 20
- punto di controllo, 8
- PV, 25
- Quaqua, 31
- resistore, 9
- rettangolo, 6, 12, 22, 24
- ricerca in librerie, 4
- ricorsione, 29
- risoluzione, 11
- root, 32
- RP, 24
- RV, 24
- SA, 26
- Safari, 19
- saldature, 11
- schema elettrico, 22
- schemi elettrici, 11
- SCR, 19
- segmento, 23
- selezione, 6, 7, 12
- serigrafia, 11, 12
- sfera di cristallo, 28
- sistema di coordinate, 22
- specchiatura scritte, 13
- sposta, 6
- stampa, 14
- stile del testo, 24
- stira e ammira, 14
- Sun, 4, 20, 32
- SVG, 19
- Symbol, 18
- TE, 24
- terminale, 20
- testo, 6, 10, 22
 - ruotato, 24
 - specchiato, 24
- testo avanzato, 24
- testo semplice, 24
- Times, 18
- Times New Roman, 18
- Times Roman, 18
- tolleranza agli errori, 27
- transistor, 8
- trasferibili R41, 12, 13
- tratteggio, 16
- TY, 24
- Ubuntu, 32, 33
- unità logica, 11, 24
- Unix, 20
- Windows, 1, 20
- WInE, 1
- zoom, 6, 24

Questo manuale è stato redatto utilizzando PDF^LA^TE_X sotto MacOSX, adottando la classe `classicthesis`. I listati sono stati composti utilizzando il pacchetto `listings`. Il pacchetto `pgf` è stato utilizzato per la figura 5. I pacchetti utilizzati sono disponibili nell'archivio CTAN.

Il lavoro è composto con il font *Palatino*, di Hermann Zapf.