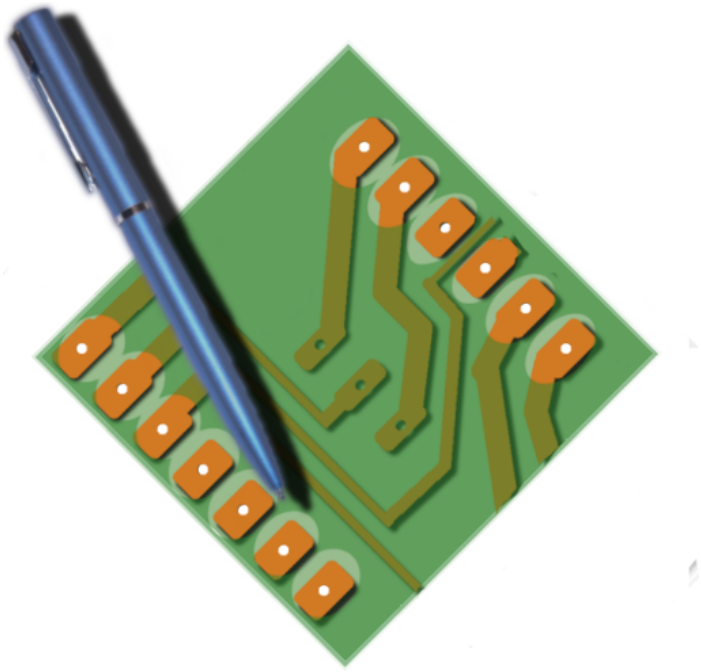
FidoCadJ 0.24.1
manuale dell'utente

Davide Bucci



14 novembre 2014

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 poco documentati altrove, come la descrizione completa del formato testuale usato da FidoCadJ. Infine, verranno fornite alcune indicazioni utili per scaricare ed installare FidoCadJ utilizzando i sistemi operativi più diffusi: Linux, MacOSX e Windows.

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 avermi dato tregua fino a che FidoCadJ non è diventato un minimo decente e di aver lavorato per la libreria CadSoft Eagle. 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 **??** 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. Gli utenti Macintosh possono contare sul buon aspetto di FidoCadJ grazie al look and feel Quaqua, frutto del lavoro di Werner Randelshofer. Werner ha anche fornito molti consigli sull'organizzazione dell'interfaccia utente di FidoCadJ: grazie!

Nell'aprile 2010, FidoCadJ è stato integrato all'interno del portale di elettronica www.electroyou.it. Ringrazio l'admin Zeno Martini, il webmaster Nicolò Martini e tanti utilizzatori per avermi fornito degli ottimi spunti per adattare FidoCadJ a girare in batch per convertire gli schemi sul loro server: è stata aperta una via molto interessante e promettente. Più di recente, la classe FidoReadPHP, capace di leggere ed interpretare i disegni FidoCadJ è stata adottata anche sul forum di www.grix.it grazie agli sforzi di Arniek e di Sstrix, che ringrazio.

FidoCadJ License

Indice

Elenco delle figure

Elenco delle tabelle

1 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

FidoCAD (senza la J alla fine!) era un software di disegno particolarmente adatto ad essere usato per tracciare semplicemente schemi elettrici e circuiti stampati. Può ancora essere scaricato gratuitamente 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. Quindi 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.

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?

Chi avesse già utilizzato FidoCAD, dovrebbe trovarsi abbastanza a suo agio con FidoCadJ. 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 fra FidoCadJ e il FidoCAD per Windows originale. Questo è stato fatto fin dove possibile, ma le esigenze degli utenti hanno ben presto portato ad introdurre delle estensioni rispetto al formato originale.

Fra i miei interessi compare la tipografia al calcolatore, ed in particolare il LaTeX. Per questa ragione, 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). Naturalmente, FidoCadJ può esportare anche verso il notissimo formato PDF.

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

1.2 Storia di FidoCadJ

Sono un appassionato di elettronica da molto tempo. Quando parecchi 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 studiarmi in dettaglio il formato utilizzato dal FidoCAD e scrivere un'applet Java chiamata FidoReadJ, capace di interpretarlo e mostrare il risultato all'interno di una pagina Web. Ho fatto un po' di reverse engineering, ho cercato informazioni dappertutto per capire quali fossero le possibilità e le limitazioni del formato, ed infine mi sono scaricato e studiato i sorgenti in C++ (peraltro 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 mettere a punto la gestione dell'interfaccia utilizzatore e dei vari elementi di disegno. 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. Si è quindi arrivati ad un punto in cui FidoCadJ può fare molte cose che il vecchio FidoCAD non prevedeva.

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 legasse ad una particolare architettura per il futuro. Inoltre, Java era un linguaggio che conoscevo già piuttosto bene, mentre lo stesso non potevo dire di altre soluzioni multipiattaforma. 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 certamente 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 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

[1] FidoReadJ è sempre disponibile all'indirizzo:
http://davbucci.chez-alice.fr/index.php?argument=elettronica/fidoreadj/fidoreadj.inc.

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 lì si possono scaricare gli eseguibili ed anche questo manuale. Si può partecipare attivamente allo sviluppo del programma, scaricandosi il codice sorgente, utilizzando Subversion, oppure con il browser SVN fornito da SourceForge:

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

Per dare una mano allo sviluppo di 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... Del tempo che dedico a FidoCadJ, solo una metà circa è dedicata al lavoro del codice. Il resto è costituito dalle risposte alle domande degli utenti e dal lavoro sulla documentazione. Se vi va, potete anche partecipare al forum di FidoCadJ, scrivere una recensione del programma, suggerire miglioramenti e segnalare bug. Il punto da cui partire è la pagina SourceForge:

http://sourceforge.net/projects/fidocadj/

1.3 FidoCadJ ed il futuro

Nell'aprile 2010, sono capitato quasi per caso in una discussione all'interno del forum del noto portale www.electroyou.it. Un utente, Giancarlo Boletti, peraltro piuttosto famoso in diversi altri ambiti, ha proposto di integrare uno strumento di editing degli schemi elettrici direttamente all'interno del forum, come già era stato fatto con LaTeX. Dato che veniva menzionato FidoCadJ, mi sono iscritto al sito e mi sono offerto di collaborare. Dopo pochissimi giorni, grazie al lavoro del webmaster, il sistema è stato finalizzato: basta fare un copia/incolla in un messaggio a partire dal programma e contrassegnare il codice con un paio di tag. Il motore del forum si occupa di lanciare FidoCadJ in maniera silenziosa sul server e recuperare un'immagine convertita. In questo modo, chi solo legge la discussione non deve avere nulla di installato per vedere l'immagine. Tuttavia, qualora desiderasse modificare l'immagine esistente, con pochi click può ottenere il codice da cui partire a fare le modifiche. La figura **??** mostra una parte di un mio messaggio inviato ad www.electroyou.it. Il sistema ha una praticità ed una flessibilità tale che sono stato il primo ad essere stupito dell'entusiasmo dimostrato dalla comunità di www.electroyou.it.[2]

Il successo che il programma ha avuto su www.electroyou.it ha fatto nascere delle richieste altrove, ed in particolare su www.grix.it, un altro portale italiano molto noto. In questo caso, dato che il server era attrezzato per far funzionare programmi in Java, è stata sviluppata la classe FidoReadPHP, una sorta di interprete scritto in PHP. Il progetto FidoReadPHP è pure lui open source ed è disponibile su SourceForge:

https://sourceforge.net/projects/fidoreadphp/

[2] Per maggiori informazioni: http://www.electroyou.it/darwinne/wiki/fidocadj

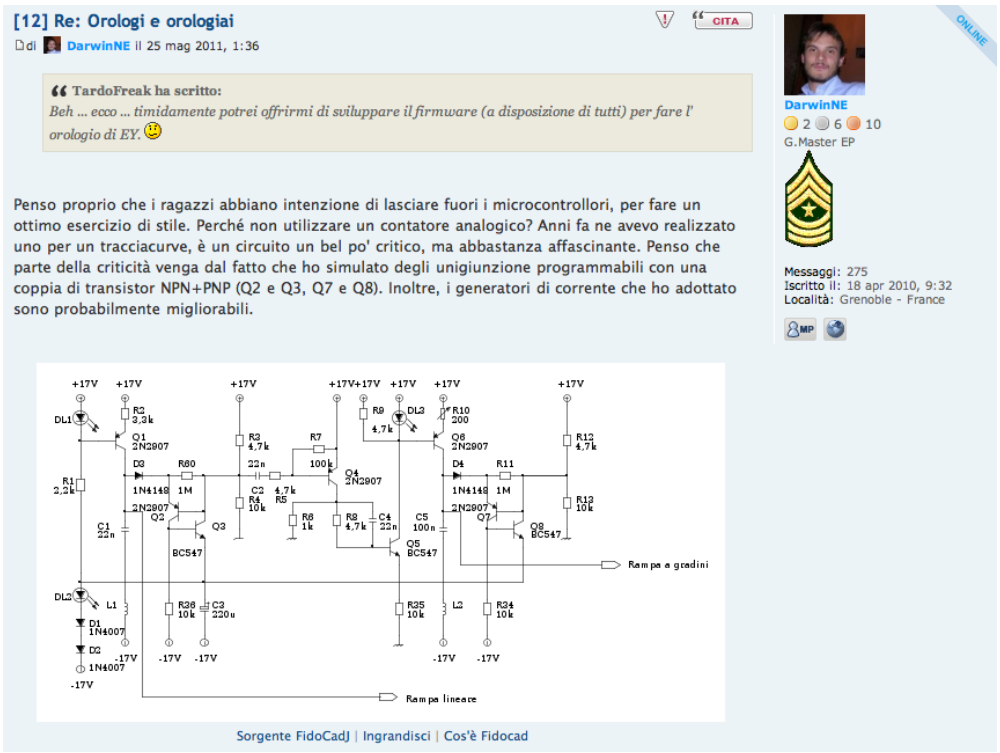


Figura 1.1: Una parte di un mio intervento sul forum di www.electroyou.it. Basta un solo click e si può ingrandire lo schema elettrico. Con un altro, si accede immediatamente al codice pronto da essere nuovamente modificato con FidoCadJ.

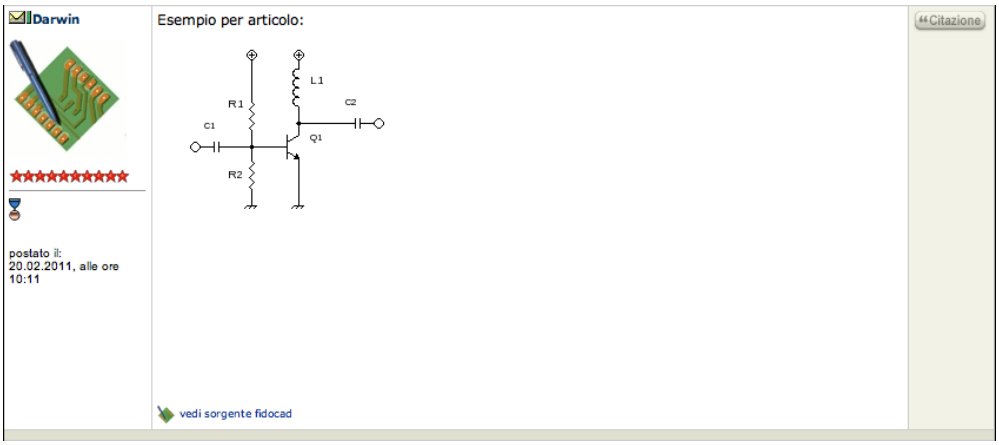


Figura 1.2: Un esempio di integrazione di un disegno FidoCadJ all'interno di un post del forum di www.grix.it. Anche qui si può accedere con un click al sorgente del disegno.

Il risultato è simile a quanto si è ottenuto su www.electroyou.it, anche se le potenzialità grafiche delle librerie del PHP sono parecchio limitate rispetto a quanto offre Java.[3] La figura **??** mostra un esempio di tale tecnica.

Credo che il futuro di FidoCadJ non sia di diventare sempre più complesso alla stregua di altri programmi di CAD elettronico, ma piuttosto vada curata sempre di più la possibilità di integrazione con gruppi di discussione e portali tecnici. L'esperienza ha mostrato che questo è possibile e l'entusiasmo degli utenti suggerisce che questa è la via da percorrere, proprio nello spirito dell'idea originale che ha fatto nascere il primo FidoCAD.

[3] Per maggiori informazioni: http://www.grix.it/viewer.php?page=9335

2 Il disegno con FidoCadJ

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

Usare FidoCadJ dovrebbe essere abbastanza intuitivo per chi abbia già usato un programma di disegno vettoriale. L'aspetto con MacOSX è mostrato nella figura **??**; sotto altri sistemi operativi cambia qualche dettaglio (per esempio, la figura **??** 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 **??** 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 elemento di disegno verrà introdotto.[1] Sulla destra, una lista attivabile con un click mostra il layer corrente (riferitevi al paragrafo **??** per maggiori informazioni).

La barra dei comandi è parzialmente personalizzabile. In particolare, si 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 "File/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 **??** e **??** 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. Una cosa che può essere utile per allineare con precisione degli oggetti senza dover disattivare la griglia è selezionare gli elementi da muovere e poi premere Alt ed i tasti cursore.

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 FidoCAD includono tutti i simboli

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

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

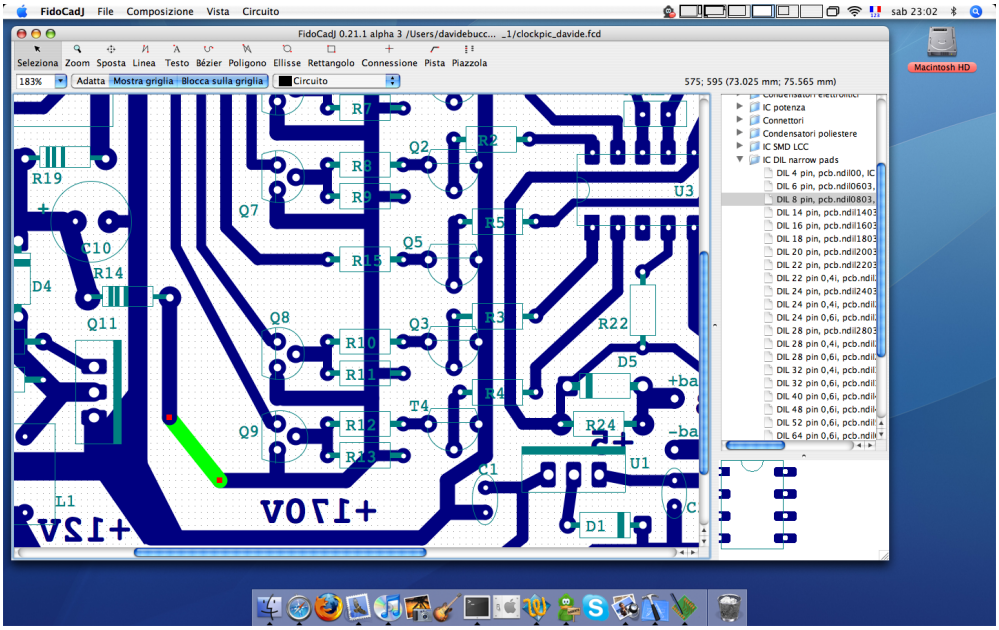


Figura 2.1: Una tipica sessione di lavoro di FidoCadJ sotto MacOSX Tiger. Nell'appendice **??**, sono descritte le peculiarità della versione specifica per Macintosh.

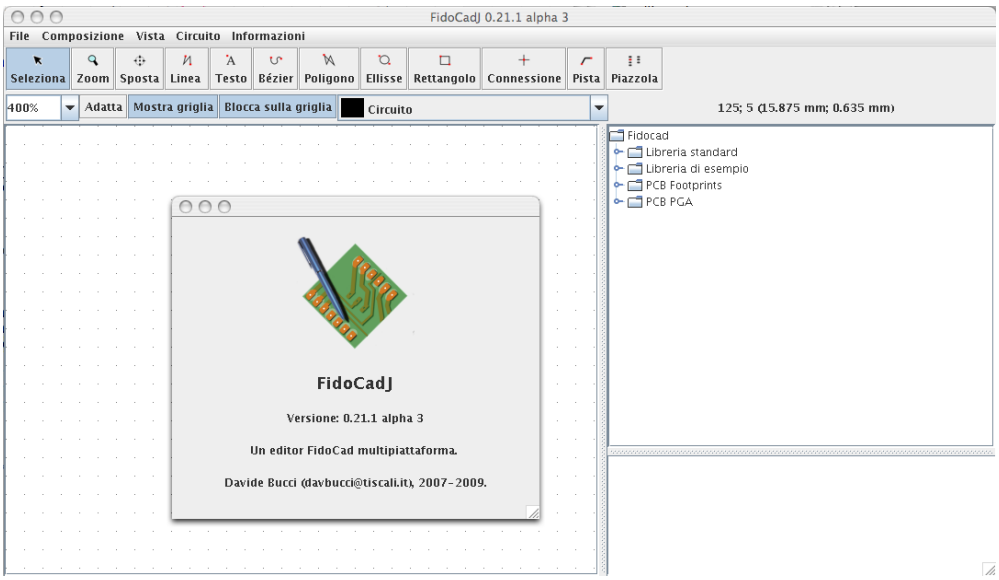


Figura 2.2: L'aspetto di FidoCadJ con il look and feel Metal.

Tasto	Comando	Uso
A o Spazio	Seleziona	Selezione di uno o più elementi grafici. Control o Command con MacOSX, per selezioni multiple o deselezionare un solo elemento. Click e trascinare per selezionare più elementi in un'area. R per ruotare la selezione. S per specchiare la selezione. Doppio click su un elemento per modificarne le proprietà.
	Zoom	Click con il pulsante sinistro per aumentare lo zoom. Click con il pulsante destro per diminuirlo.
	Sposta	Click e spostare il mouse per scorrere la porzione visualizzata.
L	Linea	Linea o una serie di linee. Esc, click destro o doppio click sinistro per terminare l'introduzione.
T	Testo	Riga di testo.
B	Bézier	Curva di Bézier.
P	Poligono	Poligono pieno o vuoto. Doppio click o Esc, per terminare l'introduzione dei vertici.
K	Curva	Curva spline naturale cubica aperta o chiusa. Doppio click o Esc, per terminare l'introduzione dei vertici.
E	Ellisse	Ellisse pieno o vuoto (tener premuto Control per ottenere un cerchio).
G	Rettangolo	Rettangolo pieno o vuoto.
C	Connessione	Connessione elettrica in uno schema.
I	Pista	Pista di circuito stampato. La larghezza di default può essere modificata da "File/Opzioni".
Z	Piazzola	Piazzola per circuito stampato. Le dimensioni di default possono essere modificate da "File/Opzioni".

Tabella 2.1: Riassunto dei comandi di disegno di FidoCadJ. Il tasto nella colonna più a sinistra permette l'attivazione da tastiera. Un click destro permette di modificare l'elemento.

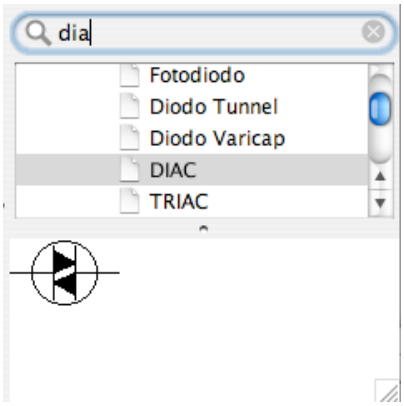


Figura 2.3: La possibilità di ricerca all'interno delle librerie caricate.

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 **??**) permette di ricercare rapidamente un elemento fra quelli presenti nelle librerie caricate dal programma. Usando le frecce su e giù, è possibile navigare all'interno di tutti gli elementi trovati.

La figura **??** 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 Disegniamo un semplice schema elettrico

Per capire come va usato il programma, procederemo a disegnare un semplicissimo schema elettrico, come quello mostrato in figura **??**. Per 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 (il programma mostra un'anteprima), facendo click una seconda volta nella posizione voluta. A questo punto, dovremmo trovarci davanti una situazione simile a quella mostrata nella figura **??**.

Una cosa che si nota subito è che il transistor bipolare sulla sinistra non è orientato

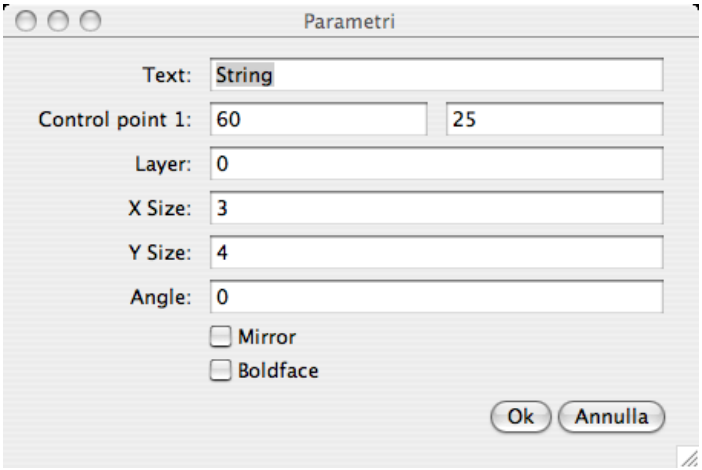


Figura 2.4: La finestra dei parametri di un testo in un disegno FidoCadJ.

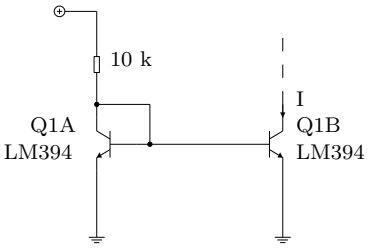


Figura 2.5: Ecco il nostro obiettivo: uno specchio di corrente che utilizza transistor NPN.

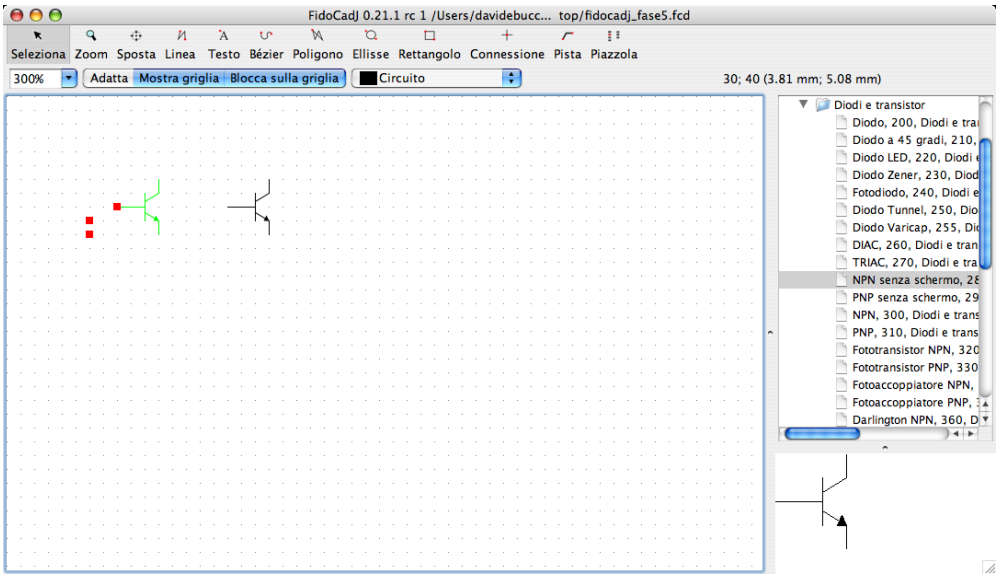


Figura 2.6: Incominciamo a disegnare un paio di transistor.

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. Un'altra possibilità è quella di premere il tasto S prima di introdurre il transistor nel disegno, quando questo è già stato selezionato nella lista dei simboli. Otterremo quindi il risultato visibile in figura **??**.

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 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 **??**). 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 **??**.

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 stan-

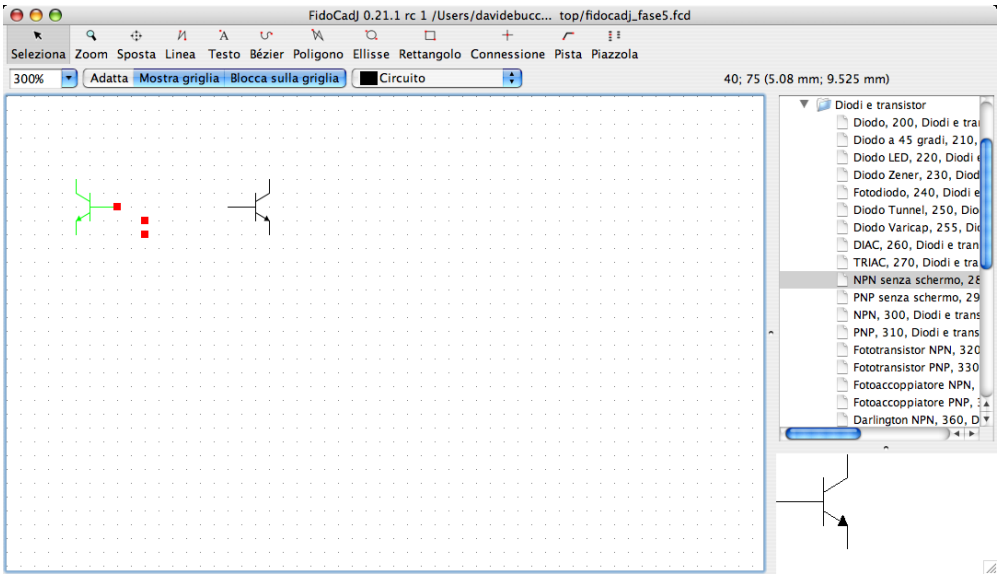


Figura 2.7: Selezioniamo e specchiamo con S il transistor a sinistra.

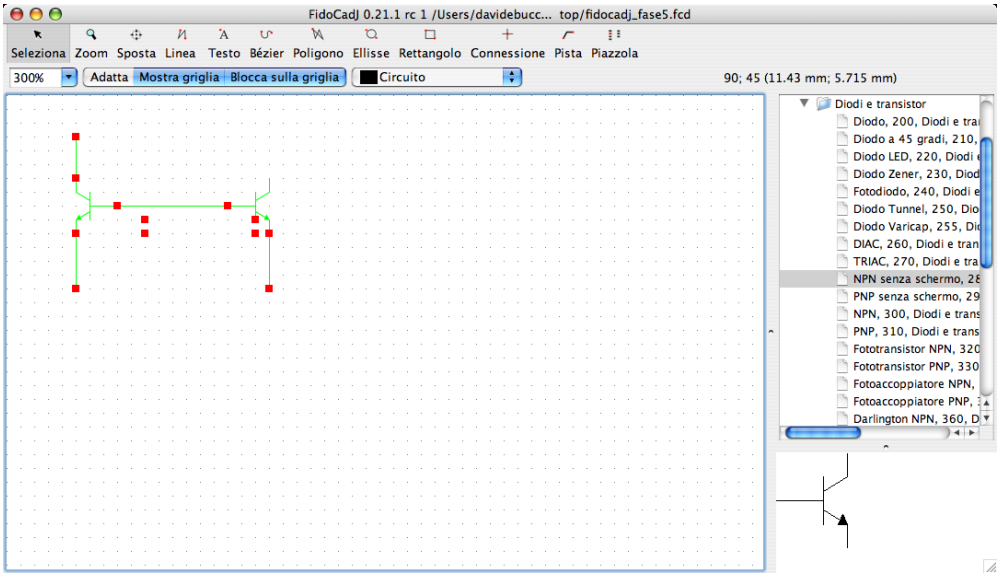


Figura 2.8: Siamo troppo in alto: selezioniamo tutto e spostiamoci più in basso.

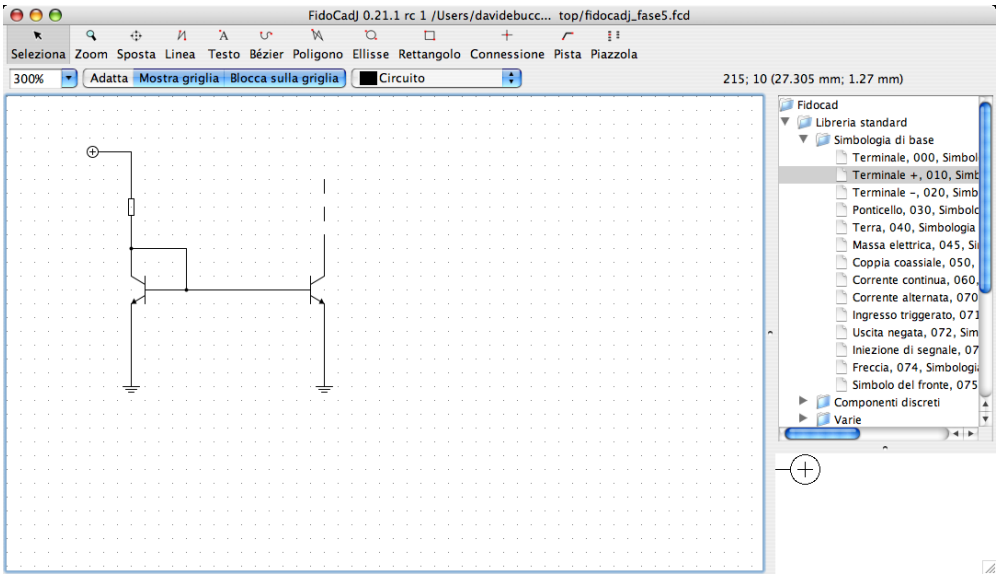


Figura 2.9: Il circuito quasi terminato

dard/Simbologia di base". Per il testo, bisogna 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 **??**. 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 **??**.

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
```

[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 **??** per maggiori informazioni sulla compatibilità.

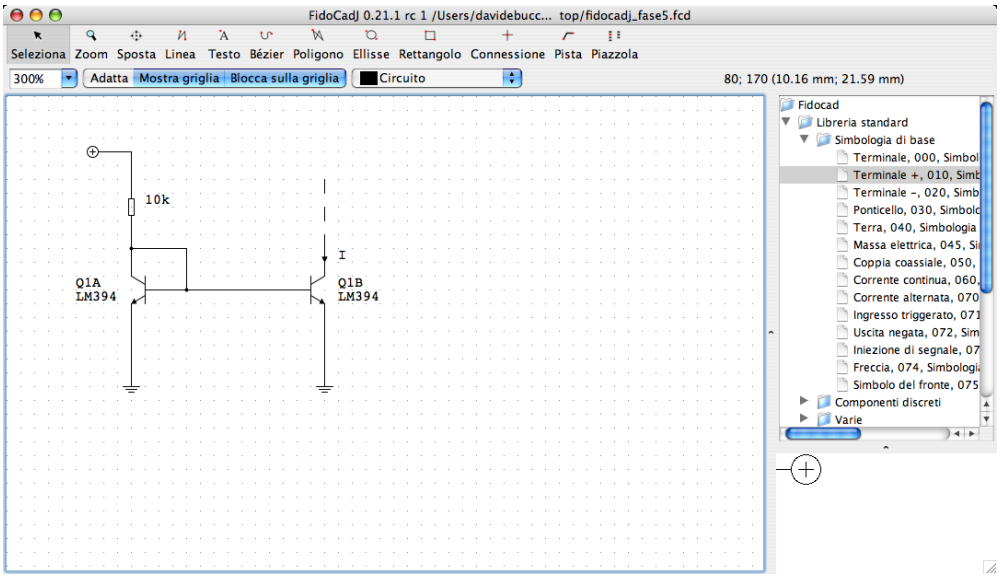


Figura 2.10: Ecco come viene mostrato il circuito finito

```
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
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 **??**.

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.

A seconda delle configurazioni attive, FidoCadJ può applicare una traslazione in diagonale, di altezza e larghezza (x e y) uguali al passo della griglia, quando si esegue un copia/incolla degli elementi. Il comportamento per default è di applicare questa traslazione per differenziare gli elementi copiati da quelli originali. Dato che in qualche situazione ciò può essere inutile o fastidioso, questo comportamento può essere disattivato nelle "Opzioni".

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.

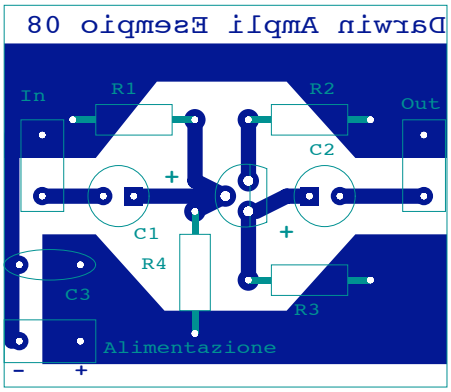


Figura 2.11: Un semplicissimo stadio amplificatore ad un transistor NPN connesso ad emettitore comune.

2.5 Disegniamo 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 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.

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.

Bisogna notare che il progetto di un circuito stampato, specie se complesso, non è affatto un compito semplice. Esistono, è vero, autoplacer 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 giocano 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 consiglierei 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 **??**. 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 consiglierei di fare è di piazzare approssimativamente i componenti, nel nostro caso il transistor (biblioteca "PCB footprints/Semiconduttori 3

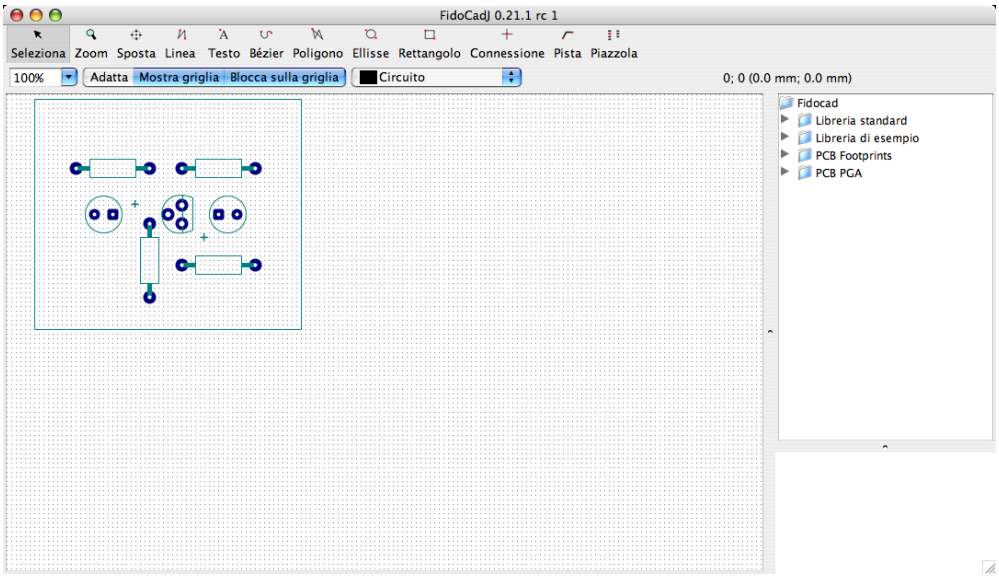


Figura 2.12: I principali componenti vengono posizionati sulla piastrina.

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 **??**.[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 fare doppio click sul perimetro in modalità selezione, per specificare all'interno della finestra di dialogo che 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. Avrà probabilmente la dimensione corretta. Ricordiamoci in fondo che FidoCadJ nasce dai trasferibili... Con un po' di attenzione, dovrebbe essere possibile ottenere il risultato mostrato in figura **??**.

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.

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.

[4] FidoCadJ è attualmente distribuito con una nazionalizzazione in italiano, francese, spagnolo, tedesco, inglese, cinese ed olandese. 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.

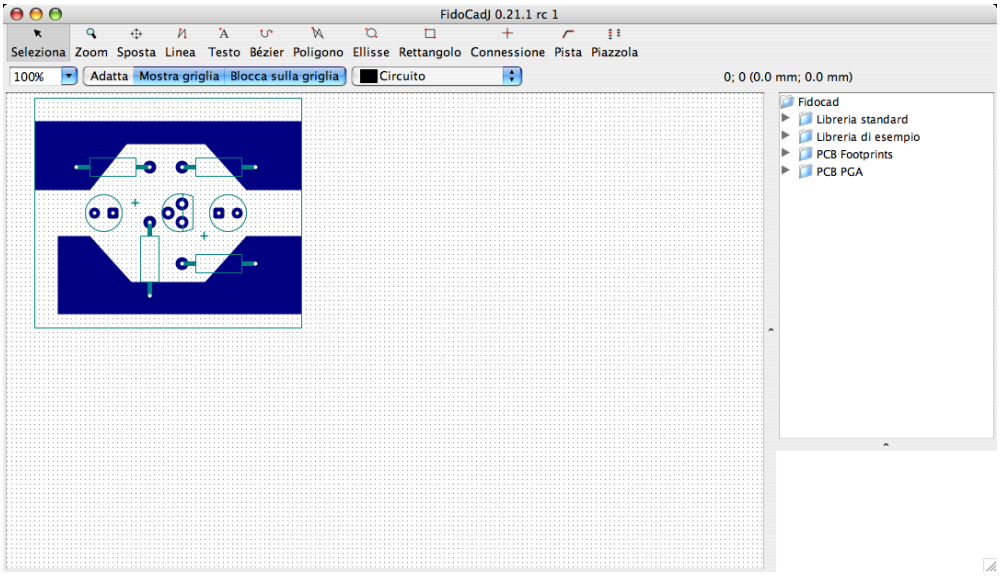


Figura 2.13: Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni.

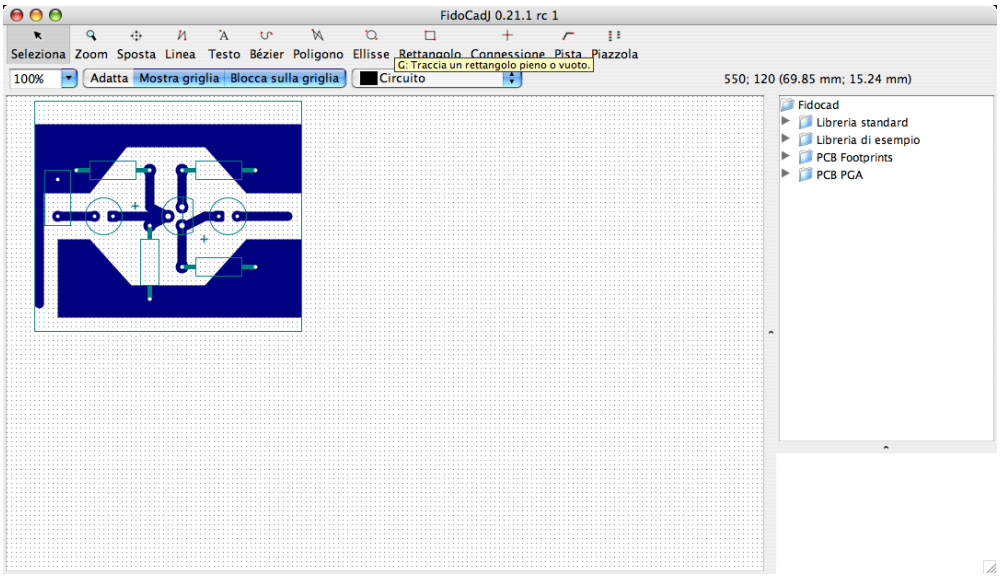


Figura 2.14: Abbiamo aggiunto le connessioni di massa e di alimentazione positiva con dei poligoni.

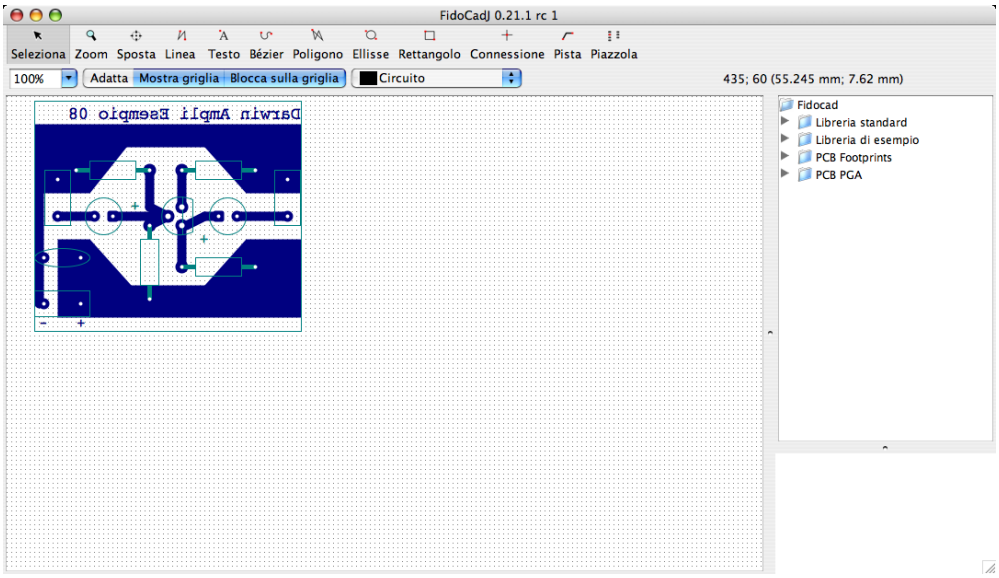


Figura 2.15: Ecco lo stampato quasi terminato.

Ci rendiamo conto di aver bisogno di alcuni connettori: per far entrare il segnale, per prelevarlo e per l'alimentazione. Possiamo tranquillamente utilizzare un footprint previsto per un condensatore al poliestere.

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 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 **??** 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 **??**.

Una volta concluso il lavoro, bisognerà probabilmente stampare il risultato su lucido da proiezione, 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) e selezionando la stampa in bianco e

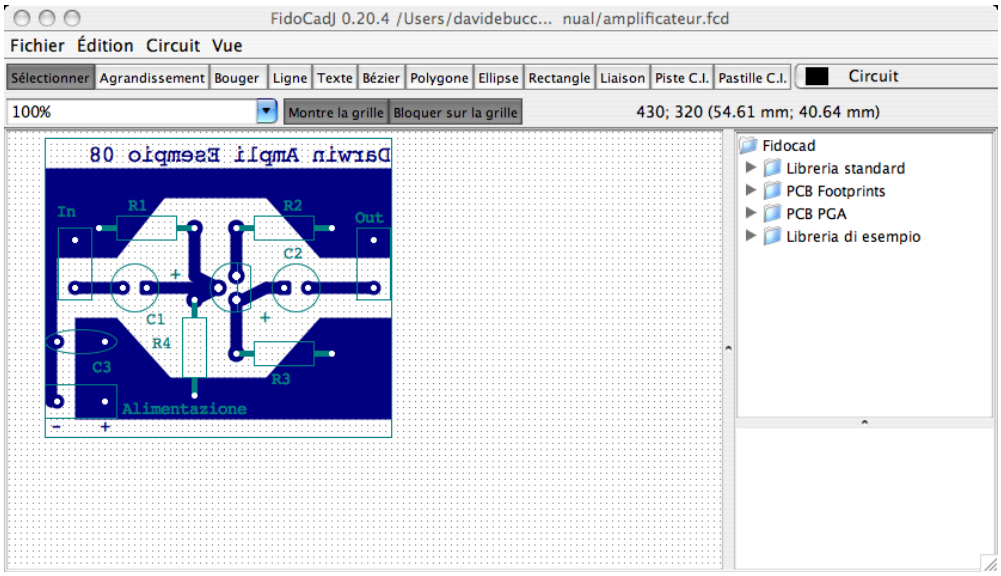


Figura 2.16: Il lavoro finito, con tutte le serigrafie.

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 **??**.

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
```

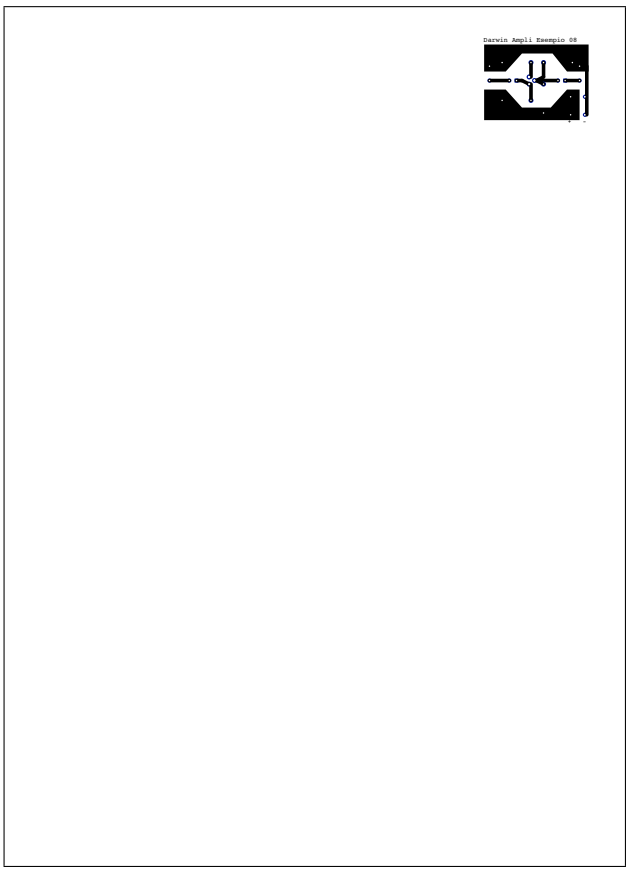


Figura 2.17: Il circuito stampato, così come appare stampato (specchiato) su una pagina in formato ISO-UNI A4.

```
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.TO92
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
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 Usiamo un righello

Durante il disegno di un circuito stampato, è spesso utile misurare con precisione distanze all'interno del proprio progetto. Per esempio, si può voler controllare la larghezza di una pista, la distanza tra due zone ramate o la dimensione totale di una piastra. FidoCadJ offre (a partire dalla versione 0.23.2) la possibilità di piazzare arbitrariamente un righello nella zona di lavoro per facilitare queste cose. Basta fare click con il tasto destro del mouse e trascinare per vedere apparire un righello, come mostrato in figura **??**. Se l'operazione di click destro e trascinamento è scomoda con il vostro sistema, si può alternativamente fare click con il tasto sinistro e trascinare, mantenendo però premuto il tasto Shift. La lunghezza totale misurata da FidoCadJ è mostrata in unità logiche ed anche in millimetri. Questo è utile quando il disegno viene stampato in scala 1:1, com'è il caso per i circuiti stampati.

2.7 Frecce 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

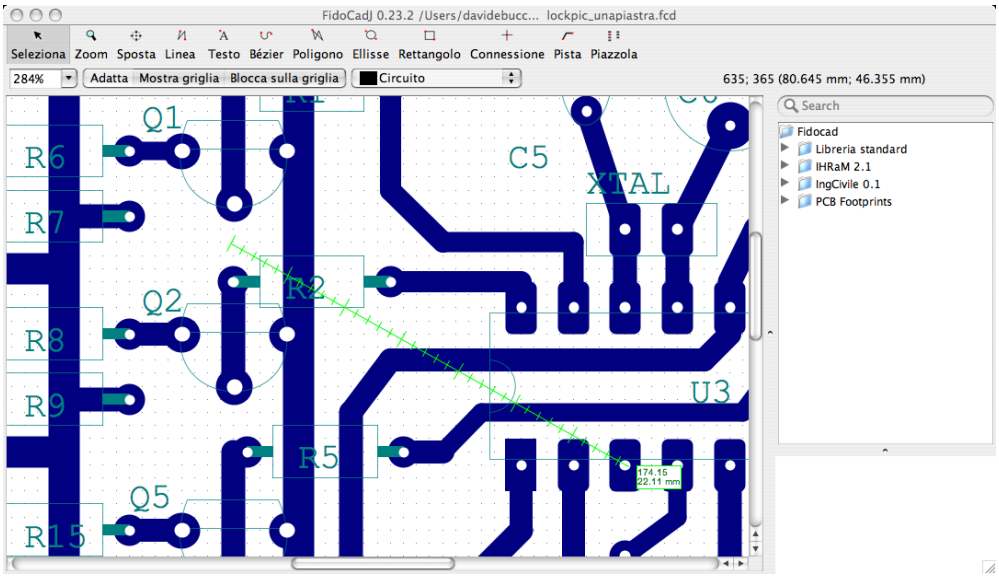


Figura 2.18: Fare click destro e trascinare per vedere apparire un righello.

figura **??** 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 **??**. 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.

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 **??**.

2.8 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

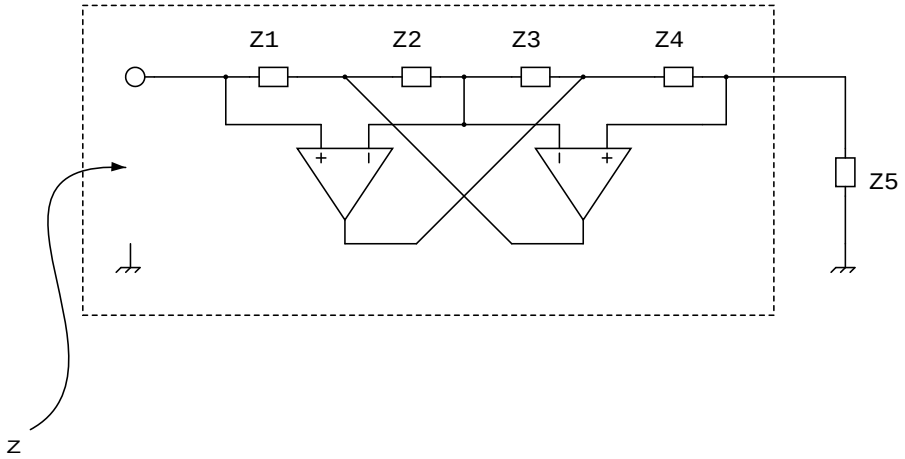


Figura 2.19: Uno schema elettrico di un GIC in cui sono state utilizzate alcune estensioni proprie a FidoCadJ.

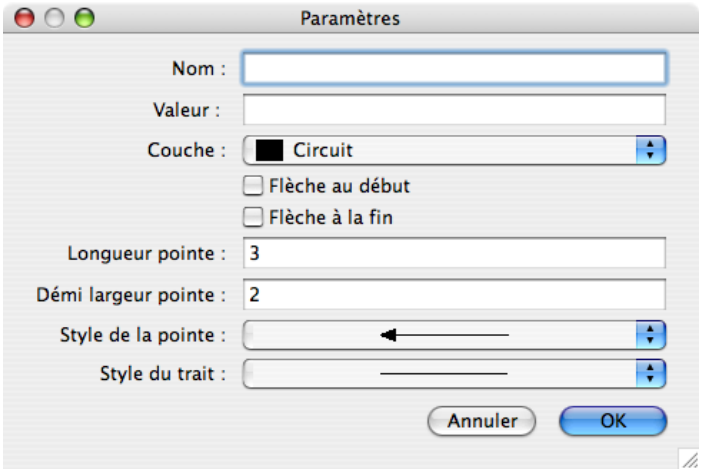


Figura 2.20: La finestra dei parametri di una curva di Bézier in un disegno FidoCadJ. Si noti la possibilità di specificare un nome ed un valore per quasi ogni elemento del disegno.

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 **??** 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 **??**, 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]

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

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, mentre invece diventa possibile decidere un fattore di scala.

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.

2.9 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 1.5, che è in giro ormai da diversi anni.

[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.

[6] Salvo nella versione per Macintosh, per la quale vi è un'applicazione a sé stante.

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 **??**, a pagina **??** (passando per una conversione in PDF, dato che utilizzo pdfLaTeX).
PGF	Formato vettoriale da utilizzare direttamente all'interno di un file LaTeX, che utilizzi il pacchetto *pgf*, 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 **??**, a pagina **??**.
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 `lbr` dell'installazione di Eagle la libreria `FidoCadJLIB.lbr`, 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 2.2: Lista dei formati di esportazione disponibili con FidoCadJ.

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` (sempreché questi esista). Esistono però altre cose interessanti che FidoCadJ può fare. L'opzione `-h` mostra un elenco delle possibilità a disposizione dell'utente:

```
[davidebucci@Darwin-iMac-G5]$ java -jar fidocadj.jar -h

This is FidoCadJ, version 0.24.1 gamma.
By Davide Bucci, 2007-2012.

Use: java -jar fidocadj.jar [-options] [file]
where options include:

 -n     Do not start the graphical user interface (headless mode)

 -d     Set the extern library directory
        Usage: -d dir
        where 'dir' is the path of the directory you want to use.

 -c     Convert the given file to a graphical format.
        Usage: -c sx sy eps|pdf|svg|png|jpg|fcd|sch outfile
        If you use this command line option, you *must* specify a FidoCadJ
        file to convert.
        An alternative is to specify the resolution in pixels per logical unit
        by preceding it by the letter 'r' (without spaces), instead of giving
        sx and sy.

 -s     Print the size  of the specified file in logical coordinates.

 -h     Print this help and exit.

 -t     Print the time used by FidoCadJ for the specified operation.

 -p     Do not activate some platform-dependent optimizations. You might try
        this option if FidoCadJ hangs or is painfully slow.

 -l     Force FidoCadJ to use a certain locale (the code might follow
        immediately or be separated by an optional space).

 [file] The optional (except if you use the -d or -s options) FidoCadJ file to
        load at startup time.

Example: load and convert a FidoCadJ drawing to a 800x600 pixel png file
        without using the GUI.
  java -jar fidocadj.jar -n -c 800 600 png out1.png test1.fcd

Example: load and convert a FidoCadJ drawing to a png file without using the
        graphic user interface (the so called headless mode).
        Each FidoCadJ logical unit will be converted in 2 pixels on the image.
  java -jar fidocadj.jar -n -c r2 png out2.png test2.fcd

Example: load FidoCadJ forcing the locale to simplified chinese (zh).
  java -jar fidocadj.jar -l zh
```

```
[davidebucci@Darwin-iMac-G5]$
```

L'opzione più semplice è `-n` ("meno enne"), con cui il programma... non fa nulla, ovvero non fa partire un'interfaccia utente e non fa *sua sponte* niente di interessante. C'è da rilevare che attivando il programma in questo modo, l'opzione `java.awt.headless` viene automaticamente impostata a `true`. Ovviamente, l'opzione `-n` non serve a molto da sola, ma è interessante combinata con altre che vedremo tra un poco. Essa infatti permette di lanciare FidoCadJ su un server, oppure all'interno di uno script per effettuare operazioni automatiche.

L'opzione `-d` permette di specificare una directory dove pescare delle librerie aggiuntive e, da ultima, l'opzione `-c` permette di convertire un file FidoCAD in un formato grafico a scelta fra quelli disponibili. Si tratta quindi dell'opzione che ci interessa maggiormente, soprattutto se utilizzata congiuntamente all'opzione `-n`, che evita al programma di far partire un'interfaccia utente grafica. Prendiamo l'esempio riportato nell'help:

```
java -jar fidocadj.jar -n -c 800 600 png out.png test.fcd
```

Il programma viene lanciato senza che l'interfaccia grafica venga mai attivata, esportando in png il file test.fcd. Il file prodotto sarà `out.png` ed avrà una dimensione di 800 per 600 pixel.

C'è una versione alternativa dell'opzione `-c` che permette di specificare in quanti pixel dev'essere convertita una unità logica. Scegliere un paio di pixel per unità è un modo sicuro per cui gli schemi saranno sempre leggibili (anche se in alcuni casi un po' troppo piccoli). Fattori di conversione non interi sono possibili. Se l'esportazione avviene verso un file vettoriale, `-c` permette di moltiplicare tutte le coordinate per il fattore specificato, che chiameremo r_p. Questo fattore può non essere intero, come nell'esempio seguente:

```
java -jar fidocadj.jar -n -c r1.25 png out2.png test2.fcd
```

There is an alternative version of the `-c` option, which will allow to specify how many pixels should be used to convert one logical unit (we will call this factor r_p). FidoCadJ does not deal with half logical units (they are always integers), by choosing, let's say, r_p equal to two pixels per logical unit ensures that schematics will be always understandable (even if probably a little bit small). The r_p factor can be non integer, as in the following example:

```
java -jar fidocadj.jar -n -c r1.25 png out2.png test2.fcd
```

Per conoscere la dimensione in unità logiche di uno schema, basta usare l'opzione `-s`. Bisogna ricordarsi però che FidoCadJ inserisce un margine di $t_\mathrm{margin} = 3$ unità logiche per ogni lato quando il disegno viene esportato. Per questa ragione, se t_w è la larghezza del disegno in unità logiche ottenuta tramite l'opzione `-s`, ci si può aspettare che p_w, la larghezza in pixel sia quanto fornito dall'espressione seguente:

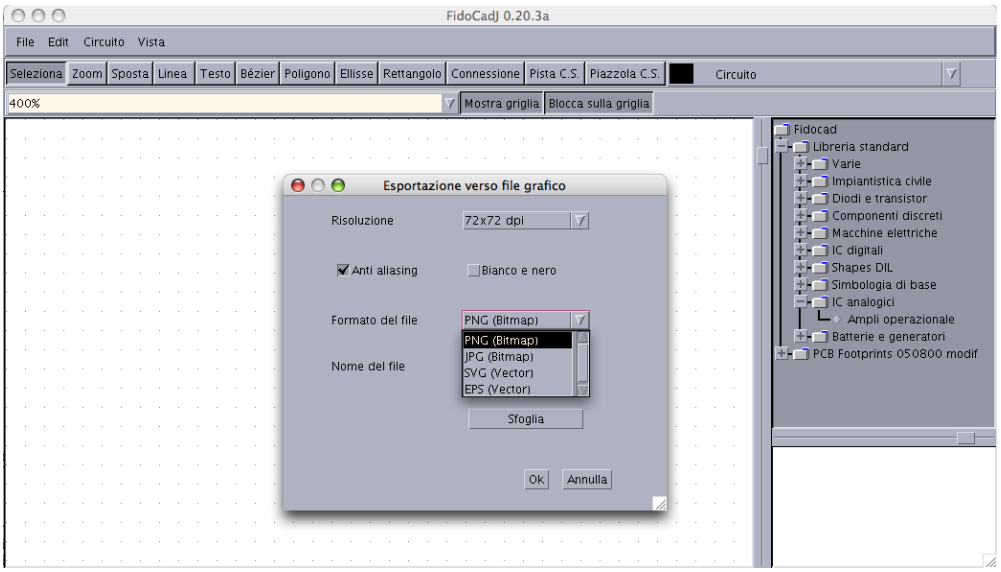


Figura 2.21: L'aspetto del programma sotto MacOSX, utilizzando il look & feel Motif.

$$p_{\mathrm{w}} = r_{\mathrm{p}}(t_{\mathrm{w}} + 2t_{\mathrm{margin}}) \tag{2.1}$$

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

Potete giocare 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[7] look & feel Motif, che è mostrato in figura **??**:

```
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 `fidocad.jar` e scrivendo tutto di seguito sulla stessa riga.

[7] La cosa può far sorridere chi è abituato ad interfacce molto curate come il look Aqua utilizzato con MacOSX. Tuttavia, ho visto un sistema di controllo di un sincrotrone funzionare con quest'interfaccia. Posso assicurare che è cosa che incute un certo rispetto.

2.10 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 "File/Opzioni" e specificarla nella riga apposita. Se è presente un file di nome `FCDstdlib.fcl`, i suoi contenuti si sostituiscono alla libreria standard disponibile nel programma. Se è presente un file di nome `PCB.fcl`, i suoi contenuti si sostituiscono alla libreria PCB disponibile nel programma.[8] Grazie all'accordo di Roby IZ1CYN, ho potuto includere la libreria IHRaM 3.1 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 all'avvio o quando l'utente configura FidoCadJ per leggere le librerie aggiuntive in un'altra directory. Si può forzare il programma a rileggere le librerie attraverso l'opzione "Aggiorna librerie" del menu "Circuito".

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

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. Attenzione che il comportamento è leggermente diverso in modalità normale, oppure quando la modalità di compatibilità con FidoCAD è attiva. Infatti normalmente sono considerati standard gli elementi delle librerie `stdlib`, `pcb`, `elettrotecnica` e `ihram`. Quando la modalità di compatibilità con FidoCAD è attiva, solo la libreria `stdlib` viene considerata come standard.

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

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. 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. Prima di cominciare, ricordo che a partire dalla versione 0.23.4, FidoCadJ utilizza unicamente l'encoding UTF-8 su tutte le piattaforme.

3.1 Descrizione dell'intestazione

Tutti i file contenenti un disegno in formato FidoCadJ iniziano con il tag `[FIDOCAD]`. Un programma può quindi riconoscere la presenza di comandi FidoCadJ 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 $127\,\mu$m, 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. Detto in termini tipografici, la risoluzione con cui lavora FidoCadJ è 200 dpi.

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 sche-

ma elettrico per occupare al meglio la dimensione del foglio, altrimenti ne verrà fuori un francobollo.

3.3 Elementi di disegno

In FidoCadJ, esistono 12 elementi di disegno:

- Linea
- Rettangolo pieno o vuoto
- Testo semplice (obsoleta)
- Testo avanzato
- Poligono pieno o vuoto
- Ellisse piena o vuota
- Curva di Bézier
- Spline naturale cubica
- Connessione elettrica
- Piazzola per circuito stampato
- Pista per circuito stampato
- Macro

Procederemo ad analizzarne il formato una per una. In generale, ogni elemento del disegno è identificato 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

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

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: il comando `LI` può essere seguito nella riga successiva da un'estensione:

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

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

Tabella 3.1: 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 3.2: Stile delle frecce, a seconda del codice b.

dove a è un intero rappresentante la presenza o l'assenza di frecce agli estremi del segmento (vedere la tabella **??**), b è un intero identificante lo stile della freccia da utilizzare (vedere la tabella **??**). 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. Se nv è uguale a 1, la linea `FCJ` dev'essere seguita da due comandi `TY` che specificano rispettivamente il nome ed il valore associati a quest'elemento.

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: i comandi `RP` e `RV` possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e nv
```

dove e è un intero che fornisce lo stile del tratteggio. Se nv è uguale a 1, la linea `FCJ` dev'essere seguita da due comandi `TY` che specificano rispettivamente il nome ed il valore associati a quest'elemento.

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

Tabella 3.3: Funzione dei bit nello stile del testo.

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 **??**. 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 +.

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: i comandi PP e PV possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e nv
```

dove e è un intero che fornisce lo stile del tratteggio. Se nv è uguale a 1, la linea FCJ dev'essere seguita da due comandi TY che specificano rispettivamente il nome ed il valore associati a quest'elemento.

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: EP e EV possono essere seguiti nella riga successiva da un'estensione:

```
FCJ e nv
```

dove e è un intero che fornisce lo stile del tratteggio. Se nv è uguale a 1, la linea FCJ dev'essere seguita da due comandi TY che specificano rispettivamente il nome ed il valore associati a quest'elemento.

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:

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

dove $t \in [0, 1]$ è un parametro.

Estensione FidoCadJ: `BE` può essere seguito nella riga successiva da un'estensione:

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

dove a è un intero rappresentante la presenza o l'assenza di frecce agli estremi della curva (vedere la tabella **??**), b è un intero identificante lo stile della freccia da utilizzare (vedere la tabella **??**). 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. Se nv è uguale a 1, la linea `FCJ` dev'essere seguita da due comandi `TY` che specificano rispettivamente il nome ed il valore associati a quest'elemento.

Curva complessa, spline

Una spline naturale cubica viene creata a partire da un certo numero di vertici. La curva passa in tutti i vertici e viene calcolata in modo che il suo percorso sia molto liscio.[1] All'interno di FidoCadJ, una spline cubica è identificata dai comandi `CV` e `CP`:

```
CV aa x1 y1 x2 y2 ... l
CP aa x1 y1 x2 y2 ... l
```

il parametro aa è uguale a 1 se la curva è chiusa, o a 0 altrimenti. I punti (x_1, y_1), $(x_2, y_2)\ldots$ sono i vertici che definiscono la spline 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 (come con i poligoni) è fissato arbitrariamente ad un centinaio, per evitare di avere a che fare con linee troppo lunghe. Questa primitiva è stata introdotta da FidoCadJ a partire dalla versione 0.24 e non è disponibile in FidoCAD.

Estensione FidoCadJ: `CV` o `CP` possono essere seguiti nella riga successiva da un'estensione:

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

dove a è un intero che indica la presenza o l'assenza di frecce agli estremi della curva (vedere la tabella **??**), b è un intero identificante lo stile della freccia da utilizzare (vedere la tabella **??**). 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. Se nv è uguale a 1, la linea `FCJ` dev'essere seguita da due comandi `TY` che specificano il nome ed il valore associati.

[1] http://www.cse.unsw.edu.au/~lambert/splines/natcubic.html

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 può essere modificato nel menu "Opzioni".

Estensione FidoCadJ: SA può essere seguito nella riga successiva da un'estensione:

```
FCJ
```

Se FCJ è presente, i due comandi successivi devono essere TY che specificano il nome ed il valore da associare a quest'elemento.

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.

Estensione FidoCadJ: PA può essere seguito nella riga successiva da un'estensione:

```
FCJ
```

Se FCJ è presente, i due comandi successivi devono essere TY che specificano il nome ed il valore da associare a quest'elemento.

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.

Estensione FidoCadJ: PL può essere seguito nella riga successiva da un'estensione:

```
FCJ
```

Se FCJ è presente, i due comandi successivi sono TY che specificano il nome ed il valore da associare.

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'orientamento è definito dal valore di o (angolo moltiplicato per 90° in senso orario) 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.

Estensione FidoCadJ: MC può essere seguito nella riga successiva da un'estensione FCJ. In questo caso, i due comandi successivi devono essere TY che specificano il nome ed il valore da associare alla macro.

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. Ecco un esempio:

```
[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
```

La presenza del comando FCJ indica che i campi indicanti il nome ed il valore sono specificati dai due comandi TY nelle linee che seguono immediatamente FCJ. Solo le coordinate ed i font dei due comandi TY sono utilizzati.

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 **??**).

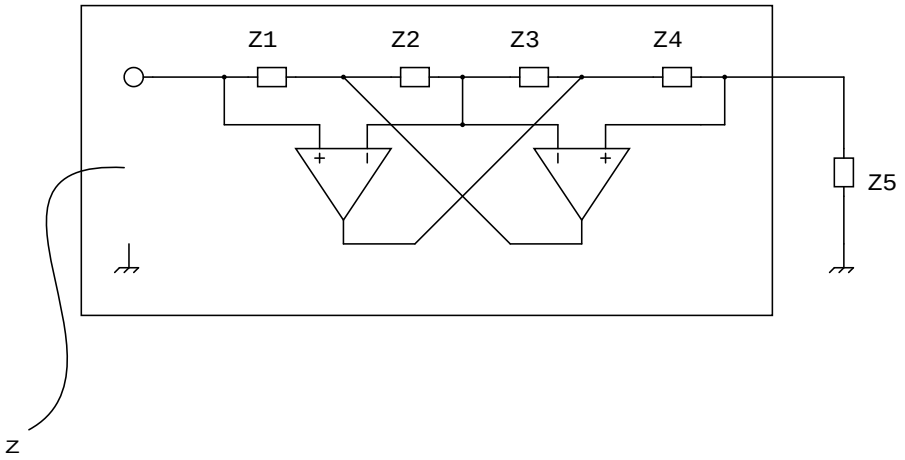


Figura 3.1: La figura **??** come probabilmente apparirebbe letta con FidoCAD.

FidoCadJ dispone di una modalità "compatibilità con FidoCAD" all'interno della quale verranno disabilitate le estensioni, di modo da risultare compatibile con il FidoCAD originale. In caso contrario, FidoCAD leggerà ancora (fornendo un messaggio di errore) i file prodotti da FidoCadJ, ma le informazioni addizionali di cui fa uso FidoCadJ non saranno prese in conto nei disegni. Naturalmente, questo limita di molto le possibilità di FidoCadJ, perché si rimane bloccati a quello che poteva fare il vecchio FidoCAD.[2] Una seconda difficoltà è che FidoCadJ utilizza sempre ed in ogni caso l'encoding dei caratteri UTF-8. Questo costituisce purtroppo una fonte d'incompatibilità con FidoCAD, basato sul vecchio sistema CP-1252. Alcune lettere accentate potranno non visualizzarsi correttamente nei file scambiati fra i due programmi. 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 **??** dovrebbe riassumere quello che si ottiene fornendo a FidoCAD il codice usato per disegnare la figura **??**. 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.

A differenza del FidoCAD originale, FidoCadJ permette di salvare un certo numero di informazioni relative alla configurazione del programma. Il comando utilizzato è `FJC` ed è solitamente emesso verso l'inizio del file. I casi trattati sono descritti nei prossimi paragrafi.

[2] Se la modalità "compatibilità con FidoCAD" viene attivata con un disegno già presente, il disegno non sarà modificato, ma le eventuali estensioni non saranno salvate con il codice. C'è quindi il rischio di perdere qualche configurazione.

3.4.1 Configurazione dei layer

La configurazione dei layer è fatta attraverso il comando `FJC L`. Vengono salvaguardate delle informazioni solamente se il layer è stato modificato rispetto alla configurazione standard di FidoCadJ.

```
FJC L n xxxx yy
```

dove n rappresenta il numero del layer (da 0 a 15), $xxxx$ è un intero a 32 bit contenente le informazioni sul colore RGB. La componente rossa è contenuta nei bit 16-23, quella verde nei bit 8-15 ed il blu nei bit 0-7. Il valore, decimale, a singola precisione, yy rappresenta la trasparenza del layer, compresa fra 0,0 (completamente trasparente) e 1,0 (completamente opaco).

Una seconda informazione importante è il nome del layer, specificato (se necessario) come segue:

```
FJC N n aaaaa
```

dove n rappresenta il numero del layer (da 0 a 15), mentre `aaaaa` è il nome del layer da assegnare. Se questa linea non è utilizzata, il nome del layer è quello assegnato per default all'avvio di FidoCadJ e che dipende dalla lingua e dalle configurazioni locali utilizzate dal sistema operativo.

3.4.2 Configurazione connessione elettrica

La dimensione del bollino utilizzato per indicare una connessione elettrica può essere modificata dal programma. Quando le estensioni FidoCadJ sono attive, il valore scelto viene salvato nel file con il comando `FJC C` nel modo seguente:

```
FJC C aaaa
```

dove *aaaa* è un valore decimale, a doppia precisione, rappresentante la taglia del bollino in unità logiche.

3.4.3 Configurazione larghezza tratto

La larghezza del pennino usato per il disegno degli schemi elettrici può essere modificata tramite il comando `FJC A`:

```
FJC A aaaa
```

dove *aaaa* è una costante decimale, a doppia precisione, che specifica la larghezza del tratto (in unità logiche) da utilizzare per gli elementi grafici (linee, rettangoli, ovali, spline, Bézier, poligoni). La larghezza di default è fissata internamente a 0,5 unità logiche. Prima della versione 0.23.6, FidoCadJ faceva differenza tra la larghezza del tratto dei segmenti e delle curve (indicata quest'ultima con `FJC B bbbb`). Questa differenza è stata eliminata nelle versioni più recenti del programma.

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 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 dal tag 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. Una libreria *non deve* contenere nelle sue macro alcuna informazione di

configurazione di FidoCadJ. In altre parole, i comandi `FJC` non devono *mai* apparire all'interno di un file di libreria.

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).

É tuttavia possibile modificare il sostituire il contenuto delle librerie interne specificando al programma (menu "File/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 sul forum di Sourceforge! [1]

[1] https://sourceforge.net/projects/fidocadj/forums/forum/997486

A 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 dell'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 1.5 di Java, che Apple forniva con il suo sistema operativo. Per ragioni probabilmente di economia di mercato, Apple non sembra molto propensa a fornire Java con le ultime release del suo sistema operativo MacOSX. Questo è un componente essenziale per far funzionare FidoCadJ, quindi se vi manca scaricatevelo ed installatelo. Fra l'altro, Apple non permette di distribuire tramite App Store programmi GPL o basati su tecnologie software che non le sono simpatiche. La mancanza di Java è

[1] http://www.randelshofer.ch/quaqua/

peraltro una delle ragioni per cui è improbabile che FidoCadJ possa girare su iPad ed iPhone.

Sebbene sia possibile utilizzare il file `fidocad.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://sourceforge.net/projects/fidocadj/files/FidoCadJ_ MacOSX.dmg/download

aprite l'immagine disco e spostate `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.

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). Una installazione insoddisfacente di Java si traduce in performance scadenti di FidoCadJ[2]. Nel paragrafo **??** descriveremo come installare il programma adoperando esclusivamente comandi da terminale. Nel paragrafo **??**, 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://downloads.sourceforge.net/project/fidocadj/fidocadj.jar?use_mirror=garr
--00:48:18--  http://downloads.sourceforge.net/project/fidocadj/fidocadj.jar?use_mirror=garr
          => 'fidocadj.jar?use_mirror=garr'
Risoluzione di downloads.sourceforge.net in corso... 216.34.181.59
Connessione a downloads.sourceforge.net|216.34.181.59:80... connesso.
HTTP richiesta inviata, aspetto la risposta... 302 Found
Posizione: http://garr.dl.sourceforge.net/project/fidocadj/fidocadj.jar [segue]
--00:48:24--  http://garr.dl.sourceforge.net/project/fidocadj/fidocadj.jar
          => 'fidocadj.jar'
Risoluzione di garr.dl.sourceforge.net in corso... 193.206.140.34
Connessione a garr.dl.sourceforge.net|193.206.140.34:80... connesso.
HTTP richiesta inviata, aspetto la risposta... 200 OK
Lunghezza: 343,207 (335K) [application/java-archive]

100%[====================================>] 343,207      422.48K/s

00:48:30 (420.55 KB/s) - "fidocadj.jar" salvato [343207/343207]
$
```

[2] N.D.C. Giurin giuretta! E non venite a dirmi che è colpa mia se la vostra distribuzione ha una versione di Java orrenda!

In alternativa, o in caso di problemi, è ovviamente possibile scaricare il file con qualunque browser dall'indirizzo:
http://sourceforge.net/projects/fidocadj/files/fidocadj.jar/download
e salvarlo nella propria `/home/`
Creiamo una directory (prima diventiamo root con `su` o `sudo -s`):

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

...e ci spostiamo il file scaricato (sostituite `<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 il comando:

  ```
  sudo nautilus
  ```

  creiamo poi la directory `/usr/bin/fidocadj` e spostiamoci il file appena scaricato (sarà in `/home/<user>/`)
- 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 **??**
- Selezioniamo la scheda "Apri con" e scegliamo "OpenJDK Java 6 Runtime" oppure "Sun Java 6 Runtime", come visibile nella figura **??**.[3]

[3] 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.

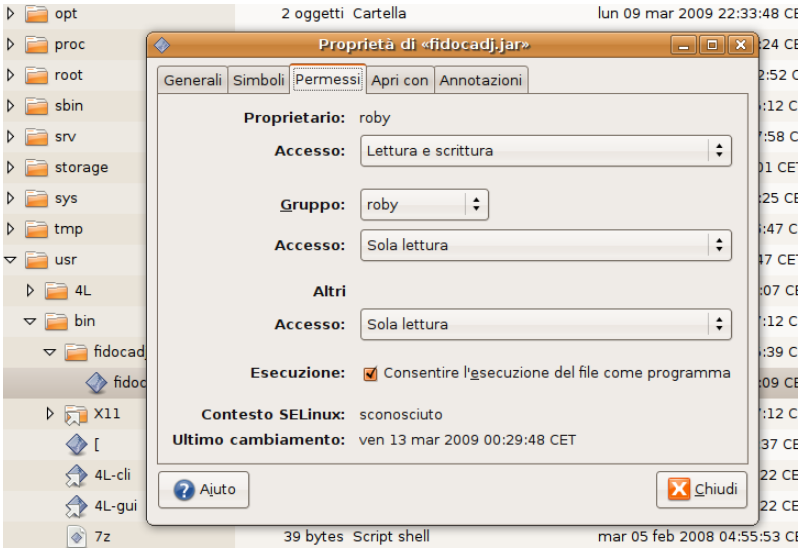


Figura A.1: La finestra di impostazione dei permessi, con la distribuzione Ubuntu 8.04.

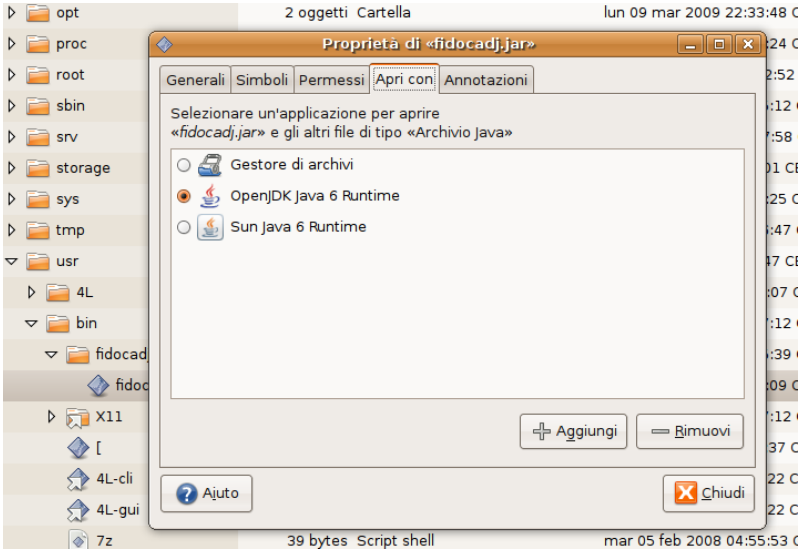


Figura A.2: Impostare l'esecuzione con la macchina virtuale Java nella distribuzione Ubuntu 8.04.

- 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.3.1 Estensioni

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

A.3.2 Come scaricare ed eseguire FidoCadJ con MacOSX

In molte situazioni classiche, se avete già una macchina virtuale Java installata[4], basta scaricare il file `fidocadj.jar` e far doppio click su di esso. Se scaricandolo il vostro navigatore lo mostrerà come un archivio `zip`, probabilmente non avete installato Java sul vostro computer. Oracle permette di scaricare gratuitamente una versione aggiornata di Java:
http://www.java.com/it/download/

[4] Apple non la fornisce più per default a partire da MacOSX 10.7 Lion, mentre prima incoraggiava l'uso di Java...

Questo manuale è stato redatto utilizzando pdfLaTeX sotto MacOSX. I listati sono stati composti utilizzando il pacchetto listings. Il pacchetto pgf è stato utilizzato per la figura **??**. I pacchetti utilizzati sono disponibili nell'archivio CTAN.