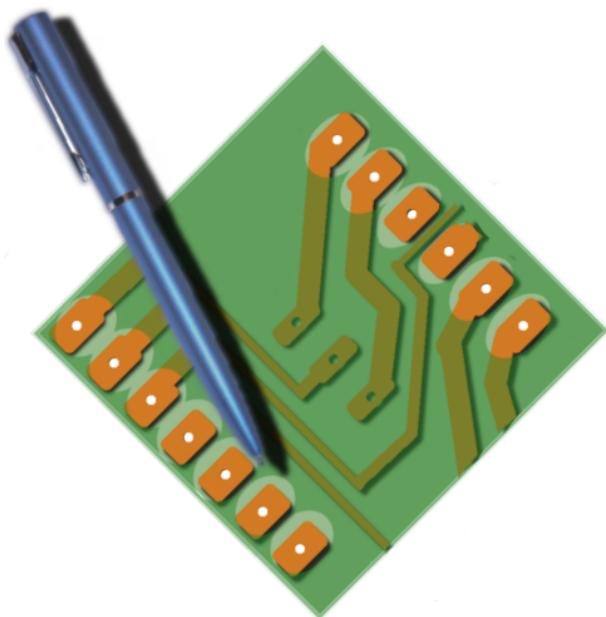


FidoCadJ 0.24.8

Benutzerhandbuch

Davide Bucci



25. Juni 2023

Dieses Handbuch ist unter der Creative Commons Public License Version 2.5 oder höher lizenziert. Der vollständige Text dieser Lizenz ist verfügbar unter <http://creativecommons.org/licenses/by-nc-nd/3.0/>.

Es steht Ihnen frei, dieses Werk unter den folgenden Bedingungen zu reproduzieren, zu verbreiten, zu kommunizieren oder öffentlich auszustellen, darzustellen, aufzuführen und abzuspielen:

Namensnennung Sie müssen das Werk in der vom Autor angegebenen Weise angeben oder Lizenzgeber (aber bedeutet in keiner Weise, dass er Sie oder Ihre Nutzung des Werks befürwortet).

Nichtkommerziell Sie dürfen dieses Werk nicht für kommerzielle Zwecke verwenden.

Keine abgeleiteten Werke Sie dürfen dieses Werk nicht verändern, umwandeln oder darauf aufbauen.

Auf jede der oben genannten Bedingungen kann verzichtet werden, wenn Sie die Erlaubnis des Urheberrechtsinhabers (Davide Bucci) einholen.

Alle in dieser Arbeit erwähnten kommerziellen Namen, Logos und Marken sind von ihren Eigentümern registriert.

Abstrakt

Dieses Dokument ist das FidoCadJ-Benutzerhandbuch. Nach einer kurzen Einführung über die Geschichte und Herkunft dieser Software wird die grundlegende Verwendung von FidoCadJ erläutert. Unser Ziel ist es, Ihnen zu zeigen, wie man sehr einfache elektronische Schaltpläne und Leiterplatten zeichnet. Das Handbuch endet mit einer detaillierten Beschreibung des FidoCadJ-Formats, das zuvor von FidoCAD verwendet wurde. Abschließend geben wir einige Tipps zum Herunterladen und Installieren von FidoCadJ unter den gängigsten Betriebssystemen: Linux, macOS und Windows.

In diesem Dokument werden lediglich das Programm selbst und das Dateiformat vorgestellt. Dabei geht es weder um die Integration von FidoCadJ in Web-sites noch um Code-Organisation und Entwicklungsaspekte. Schauen Sie sich das GitHub-Repository an, wenn Sie an der Entwicklung teilnehmen möchten!

Danksagungen

Mehrere Leute haben diese Software seit den ersten Versionen verwendet und mir mit ihren Ratschlägen weitergeholfen. Deshalb möchte ich den Teilnehmern der it.hobby.elettronica newsgroup für die sehr fruchtbaren Diskussionen danken die wir gemeinsam geführt haben.

Dank der Geduld von Stefano Martini wurde diese Software sehr sorgfältig unter Linux getestet. Er war ein sehr aufmerksamer Alpha- und Betatester! Ich möchte Olaf Marzocchi und Emanuele Baggetta für ihre Tests auf macOS danken.

Ich möchte “F. Bertolazzi”, der sehr nützliche Ratschläge zur Benutzerfreundlichkeit dieser Software gab, danken. Er hat auch die CadSoft Eagle-Kompatibilitätsbibliothek zusammengestellt, die für den Export von FidoCadJ-Zeichnungen nach Eagle nützlich ist. Vielen Dank an “Celsius”, der die Softwarefunktionalitäten für Leiterplatten und zugehörige Bibliotheken getestet hat.

Vielen Dank auch an Andrea D’Amore für seinen Rat zur FidoCadJ-Benutzeroberfläche auf dem Apple Macintosh. Ich möchte Roby IZ1CYN dafür danken, dass er Bibliotheken besprochen hat, Abschnitt A.2 dieses Handbuchs geschrieben und es unter Linux installiert hat. Vielen Dank an Max2433BO für Ratschläge zu Linux und für seine geduldige Arbeit an GitHub-Problemen.

Macintosh-Benutzer können FidoCadJ verwenden, das dank des Quaquaa-Look-and-Feel von Werner Randelshofer und neuerdings dank des VAqua7-Look-and-Feel gut in das Look-and-Feel ihres Betriebssystems integriert ist. Werner hat ebenfalls hilfreiche Beiträge geleistet Ratschläge zur Benutzeroberfläche von FidoCadJ: Danke dafür!

Ein Teil dieses Handbuchs wurde von “Pasu” ins Englisch übersetzt. Ich möchte ihm für seine harte Arbeit danken. Ich möchte auch Miles “qhg007” für die sorgfältige Überprüfung und die sehr hilfreichen Kommentare danken. Das englische Handbuch wurde von Joop Nijenhuis als Dankeschön für FidoCadJ ins Niederländische übersetzt. Das niederländische Handbuch wurde von Joop Nijenhuis mit u.a. Google ins Deutsche übersetzt.

Im April 2010 wurde FidoCadJ in die bekannte italienische Website www.electroyou.it integriert. Es läuft nun unbeaufsichtigt auf dem Server und konvertiert Zeichnungen automatisch die im Forum gepostet wurden. Ich möchte dem Administrator Zeno Martini und dem Webmaster Nicolò Martini sowie allen Benutzern dieser Website meinen Dank aussprechen. Sie haben mir viele sehr nützliche Ideen zur Stapelsteuerung FidoCadJ gegeben: dort Es ist ein vielversprechender Weg vorgezeichnet der es verdient umfassend erforscht zu

werden. FidoCadJ wird auch auf <http://www.matematicamente.it> verwendet. Dank der Bemühungen von Arniek, Sstrix und Stan wurde es von der beliebten Website www.grix.it verwendet. Großes Kompliment an sie!

Hilfreiche Hinweise zur FidoCadJ-Website kamen von Federica Garin und Emanuele Baggetta. Auch Sergio Juanez hat viel dazu beigetragen, die technischen Aspekte und die Organisation zu verbessern.

FidoCadJ-Lizenz

Copyright © 2007-2023 das FidoCadJ-Entwicklungsteam.

Diese Software ist kostenlos: Sie können sie unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation, Version 3 der Lizenz, veröffentlicht, weiterverbreiten und/oder ändern.

Dieses Programm wird in der Hoffnung verbreitet, dass es nützlich ist, jedoch OHNE JEGLICHE GARANTIE; ohne die stillschweigende Garantie der MARKTGÄNGIGKEIT oder EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Weitere Einzelheiten finden Sie in der GNU General Public License.

Sie sollten zusammen mit diesem Programm eine Kopie der GNU General Public License erhalten haben. Wenn nicht, schauen Sie nach <http://www.gnu.org/licenses/>.

Inhaltsverzeichnis

1 Einführung	1
1.1 Die Philosophie von FidoCadJ	1
1.2 Geschichte von FidoCadJ	2
1.3 FidoCadJ und die Zukunft	4
2 Zeichnen mit FidoCadJ	6
2.1 Zeichenwerkzeuge	6
2.2 Ein einfaches Schema	10
2.3 Die Schichten	15
2.4 Das Gitter	15
2.5 Eine einfache Leiterplatte	15
2.6 Verwendung des Lineals	22
2.7 Pfeil- und Strichstile	22
2.8 Exportieren	24
2.9 Befehlszeilenoptionen	26
2.10 Bibliotheksverwaltung	29
2.10.1 Verwendung von Bibliotheksdateien	29
2.10.2 Neue Symbole definieren	30
2.10.3 Vorhandene Symbole ändern	33
3 Zeichnungsformat, Makros & Bibliothek	34
3.1 Header-Beschreibung	34
3.2 Koordinatensystem	34
3.3 Elemente zeichnen	35
3.4 FidoCadJ-Erweiterungen	41
3.4.1 Ebenenaufbau	42
3.4.2 Elektrischer Anschlussaufbau	43
3.4.3 Linienbreite	43
3.5 Toleranz für Syntaxfehler	43
3.6 Bibliotheksformat	44
3.7 Standardbibliotheken	44
4 Conclusie	46
A Plattformspezifische Informationen	47
A.1 macOS	47
A.1.1 So laden Sie FidoCadJ unter macOS herunter und führen es aus	47

A.2	Linux	48
A.2.1	Auf jeder Plattform, vom Terminal aus	48
A.2.2	Auf einem grafischen System	49
A.2.3	“Portable” Installation	51
A.2.4	Raspberry Pi 400	51
A.3	Windows	52
A.3.1	So laden Sie FidoCadJ herunter und führen es aus	53
A.4	Compileren van sources	53
A.5	Android	53
B	FidoCadJ-Beispiele	55
C	FidoCadJ art	61

Abbildungsverzeichnis

1.1	Teile einen meiner Beiträge www.electroyou.it . Sie können den Zeitplan mit nur einem Klick vergrößern. Mit einem zweiten Klick erhalten Sie sofort den Quellcode, den Sie in FidoCadJ einfügen können um ihn zu ändern.	4
1.2	Ein Beispiel einer FidoCadJ-Zeichnung die in einen www.grix.it -Forumsbeitrag integriert ist. Den Quellcode der Zeichnung erhalten Sie mit nur einem Klick.	5
2.1	Die drei Menüleisten von FidoCadJ 0.24.8.	6
2.2	Eine FidoCadJ 0.24.8-Sitzung, die auf einem macOS 10.13 High Sierra läuft. Anhang A beschreibt die Besonderheiten der Version für Macintosh-Computer.	7
2.3	Eine FidoCadJ 0.24.8-Sitzung, die auf einem Raspberry Pi mit Looks and Feel von Metal läuft.	7
2.4	Die Schnellsuchfunktion für installierte Bibliotheken.	8
2.5	Dialogfeld für die Parameter eines Symbols in einer FidoCadJ-Zeichnung. Das Betriebssystem ist Raspberry Pi OS, auf Deutsch. Drücken Sie die Taste unten links, um Symbole einzugeben oder Hinweise zu erhalten.	10
2.6	Das Referenzschema: Stromspiegel mit NPN-Transistoren.	10
2.7	Wir beginnen mit der Zeichnung einiger Transistoren.	11
2.8	Wählen Sie den Transistor auf der linken Seite aus und spiegeln Sie ihn, indem Sie [S] drücken.	11
2.9	Wir sind zu nah am oberen Rand des Blattes: Wählen Sie die gesamte Zeichnung aus und verschieben Sie sie in die Mitte.	12
2.10	Die Schaltung ist fast fertig.	13
2.11	Die komplette Schaltung.	13
2.12	Eine sehr einfache Verstärkerstufe mit einem NPN-Transistor in Emitterschaltung.	16
2.13	Die wichtigsten Teile sind auf der Leiterplatte untergebracht.	17
2.14	Stromanschlüsse mit Polygonlinien hinzugefügt.	17
2.15	Die verbleibenden Leiterplattenanschlüsse wurden hinzugefügt.	18
2.16	Die Platine ist fast fertig.	18
2.17	Mit dem Siebdruck ist die Arbeit abgeschlossen.	19
2.18	Die Leiterplatte, wie sie aussieht wenn sie auf ein ISO-UNI-A4-Blatt gedruckt (gespiegelt) wird.	20
2.19	Klicken Sie mit der rechten Maustaste und ziehen Sie, um das FidoCadJ-Lineal zu aktivieren.	22

2.20 Eine elektrische Zeichnung (ein GIC von Antoniou) unter Verwendung einiger FidoCadJ-Erweiterungen.	23
2.21 Das in den Schemata der Abbildung verwendete Parameterfenster der Bézier-Kurve 2.20.	23
2.22 Das Setup-Menü für den Export in ein Bild	24
2.23 Das Aussehen von FidoCadJ unter Verwendung des Motif-Look & Feel.	29
2.24 Das per Rechtsklick erscheinende Popup-Menü ermöglicht die Umwandlung von Zeichenelementen in ein Symbol	31
2.25 Der neue Symboldefinitionsdialog. Hier können Sie alle wichtigen Eigenschaften des Symbols einstellen. Beachten Sie den durch die beiden roten Achsen definierten Ursprung.	31
2.26 Das neu erstellte Symbol, angezeigt in der Symbolliste und in der Zeichnung. Auf der linken Seite befindet sich noch die Zeichnung, die für die Symboldefinition verwendet wurde. Beachten Sie, dass es in der Zeichnung nur einen Kontrollpunkt für das neue Bibliothekssymbol gibt.	32
2.27 Das Popup-Menü zum Ändern von Symboleigenschaften in einer Benutzerbibliothek.	33
3.1 Figure 2.20, wie es in FidoCAD für Windows erscheinen würde. .	42
3.2 Beispiel dafür wie Bibliotheken im Ordner “Bibliotheken” angezeigt werden.	45
A.1 Die Einstellung der Dateirechte, auf Ubuntu 8.04.	50
A.2 FidoCadJ für Android, läuft auf einem Samsung S5 Smartphone.	54
B.1 Benutzerbeispiel DanteCpp (Österreich)	56
B.2 Ein Beispiel einer FidoCadJ-Zeichnung mit unterschiedlicher Nutzung der verfügbaren Ebenen (lesen Sie: Farben).	57
B.3 Modulares Effekt Rack. Durch Vergrößern der Seite (>200%) werden Schaltflächen und Texte besser lesbar.	57
B.4 Tuner aus Philips Mischverstärkereinheiten. Durch Vergrößern der Seite (>200%) werden Schaltflächen und Texte besser lesbar.	58
B.5 Beispiel eines Tonstudio-Layouts (der eigentliche Aufbau ist viel komplizierter und passt nicht auf ein A4, die Idee wird klar sein). Durch das Vergrößern der Seite (>200%) werden Texte besser lesbar.	59
B.6 Mein Versuch einer neuen Aufteilung des Wohnzimmers.	60

Tabellenverzeichnis

2.1	Zusammenfassung der in FidoCadJ verfügbaren Zeichenbefehle. Die in der Spalte ganz links angezeigte Taste ermöglicht eine schnelle Auswahl mit der Tastatur. Klicken Sie mit der rechten Maustaste in einen der Grundelement-Platzierungsmodi, um auf dessen Eigenschaften zuzugreifen.	9
2.2	Dieser Code beschreibt die Schaltung aus unserem Beispiel.	14
2.3	Dieser Code beschreibt die Leiterplatte aus unserem Beispiel.	21
2.4	Liste aller in FidoCadJ verfügbaren Exportdateiformate.	25
3.1	Bedeutung des Parameters a für das Vorhandensein einer Pfeilspitze an den Seiten einer Linie oder eines Bézier-Grundelements.	36
3.2	Bedeutung des b -Parameters für den Pfeilspitzenstil.	36
3.3	Funktion der Bits im Textstilbegriff.	37
A.1	Liste der verfügbaren Make-Ziele zum Kompilieren von FidoCadJ aus Quellen.	53

Kapitel 1

Einführung

In diesem Kapitel stellen wir FidoCadJ kurz vor. Insbesondere werden wir die Philosophie hinter dieser Software beschreiben und eine kurze Geschichte ihrer Entwicklung geben. Wenn Sie FidoCadJ installieren und ausführen müssen, gehen Sie zu Anhang A, in dem beschrieben wird, wie Sie FidoCadJ auf den meisten gängigen Betriebssystemen installieren.

1.1 Die Philosophie von FidoCadJ

FidoCAD (ohne das J am Ende) war eine Vektorzeichensoftware, die sich besonders für elektrische Schaltpläne und einfache Leiterplatten eignete. Es wurde Ende der 1990er Jahre von der italienischen Usenet-Community genutzt.

Es kann immer noch kostenlos von Lorenzo Luttis Seite heruntergeladen werden (eine in italienischer Sprache verstaatlichte Windows-Version):

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

Die von FidoCAD generierten Ausgabedateien waren sehr kompakter Textcode. Diese Funktion machte es sehr einfach, Zeichnungen in Textnachrichten einzufügen, wie sie beispielsweise in nicht-binären Usenet-Gruppen verwendet werden.

Leider existierte FidoCAD nur in einer Windows-Version. Linux-Benutzer konnten es mit WInE verwenden, aber diejenigen, die wie ich auf andere Plattformen angewiesen waren (ich verwende macOS), mussten eine andere Lösung finden. Deshalb beschloss ich, einen kleinen Beitrag zur Community zu leisten, indem ich FidoCadJ schrieb (diesmal mit dem letzten J). Dieser Editor ist in reinem Java geschrieben und vollständig plattformübergreifend.

Mit FidoCadJ können Sie eine Zeichnung im Dateiformat FidoCAD anzeigen und ändern. Heute arbeite ich nicht mehr alleine an dem Projekt und FidoCadJ ist ein vollwertiges und ich wage es, ein erfolgreiches Open-Source-Projekt mit Beiträgen aus der ganzen Welt zu schreiben.

Jeder, der FidoCAD in der Vergangenheit verwendet hat, sollte sich sehr schnell mit FidoCadJ vertraut machen, da viele der Befehle und Verfahren der Originalanwendung sehr ähnlich sind. FidoCadJ ist bis auf wenige Details nahezu vollständig mit dem Original-FidoCAD kompatibel. Das Ziel der vollständigen Kompatibilität wurde weitestgehend verfolgt, Anfragen neuer Nutzer führten jedoch zu einigen Erweiterungen des Originalformats.

Einige von FidoCadJ angebotene Funktionen, die im ursprünglichen FidoCAD nicht vorhanden sind, sind die verfügbaren Exportfunktionen. Da ich ein L^AT_EX-Benutzer bin, habe ich beschlossen, eine Exportfunktion für eine Reihe von Vektorformaten einzubinden, darunter Encapsulated PostScript (EPS) und Skripte für das PGF -Paket. Natürlich kann FidoCadJ mit einigen Einschränkungen in das bekannte PDF-Format exportieren.

1.2 Geschichte von FidoCadJ

Ich interessiere mich schon seit langem für elektronische Schaltkreise. Als ich anfing, mehreren speziellen italienischen Usenet-Newsgroups zu folgen, fiel mir auf, dass viele Schaltpläne im FidoCAD-für-Windows-Format bereitgestellt wurden. Das war viel besser als klobige ASCII-Zeichnungen. Da ich kein Windows verwende war es für mich fast unmöglich es anzusehen und ich wollte versuchen etwas zu tun um dieses Problem zu lösen.

Das erste, was ich Ende 2006 tat, war das von FidoCAD verwendete Dateiformat zu studieren und ein Java-Applet namens FidoReadJ zu schreiben. Es konnte die Schaltung analysieren um sie auf einer Webseite anzuzeigen. Das Internet war damals anders! Um das zu erreichen, habe ich praktisch überall gesucht (alte Beiträge, Webseiten) und eine Menge Reverse Engineering aus vorhandenen FidoCAD-Dateien durchgeführt. Ich habe auch die FidoCAD-Ressourcen heruntergeladen und studiert, die von Lorenzo Lutti in einem ziemlich klaren C++ geschrieben wurden.

Ich habe den Kern des Applets im März 2007 geschrieben. Ein paar Monate später war das Applet online und wurde von einem Teil der Community getestet, der sich für it.hobby.elettronica und it.hobby.fai-da-te interessierte.¹

Da ich einen Interpreter des FidoCAD-Formats hatte, war es interessant einen vollständigen Editor zu erstellen . Ein Großteil der anfänglichen Arbeit wurde in mehreren Schritten zwischen Januar und Juli 2008 erledigt: FidoCadJ ist keine Portierung von FidoCAD für Windows, sondern ein neues Programm, das von Grund auf neu geschrieben wurde.

Die Entscheidung Java zu verwenden liegt an der Tatsache dass ich derzeit viele verschiedene Betriebssysteme verwende. Zeit und Energie in etwas zu investieren das nicht vollständig übertragbar ist reizt mich nicht mehr. Der Aufwand das Cocoa-Framework zu erlernen hätte unter macOS wahrscheinlich zu einem besseren Ergebnis geführt, aber es hätte FidoCadJ völlig nicht portierbar gemacht. Ich bin kein Computertyp. Die Zeit die ich mit Programmieren verbringe ist Zeit die ich meinen elektronischen Interessen entziehe. Tatsächlich zeigte die FidoCadJ-Codequelle besonders am Anfang, dass ich kein großer Fan von Java und Objektorientierte Programmierung bin und einige Lösungen waren eher praktisch als elegant. Ich habe jedoch versucht die allgemeine Codequalität von FidoCadJ kontinuierlich zu verbessern. Es ist im Vergleich zu anderen Open-Source-Projekten recht erfolgreich, zumindest gelegentlich². Auf diese Weise

Übrigens denke ich dass es besser ist aktiv zu sein an einer Lösung zu arbeiten statt zu jammern weil einem anderen Betriebssystem als Windows bestimmte Software fehlt.

Um ehrlich zu sein, habe ich etwa 1993 meinen ersten Versuch unternommen, ein 2D-Vektorzeichensystem zu schreiben

¹FidoReadJ ist jetzt noch unter der folgenden Adresse verfügbar, es ist jedoch unwahrscheinlich dass verfügbare Webbrower noch Java-Applets unterstützen:

<http://davbucci.chez-alice.fr/index.php?argument=elettronica/fidoreadj/fidoreadj.inc>.

²Siehe zum Beispiel Rahman, M. M., Roy, C. K. und Keivanloo, I. "Subjektive Bewertung der Softwarequalität mithilfe von Crowdsource-Wissen: Eine explorative Studie". Univ. of Saskatchewan, technischer Bericht des Department of Computer Science 2013-01

wurde eine statische und dynamische Analyse des Quellcodes durchgeführt um zu ermitteln wo die Codeorganisation geändert und verbessert werden kann.

Was zählt ist die Erfahrung derjenigen die das Programm verwenden, viel mehr als die Wahl der Sprache. Aus diesem Grund höre ich mir immer Ihre Vorschläge an, um zu verstehen wie dieses Projekt weiterentwickelt werden kann. Java ist möglicherweise nicht für jedes Problem die perfekte Wahl oder Lösung. Es hat sich jedoch als flexibel genug für ein Tool wie FidoCadJ erwiesen.

Ohne nach Perfektion zu streben und die Grenzen meiner Programmierkenntnisse zu kennen, möchte ich sicherstellen, dass FidoCadJ KEINE Anwendung von schlechter Qualität ist. Aus diesem Grund ist jeder Fehlerbericht oder Kommentar zur Benutzerfreundlichkeit des Programms mehr als willkommen.

Im November 2009 habe ich ein SourceForge-Projekt für FidoCadJ eröffnet. Von hier aus können Sie alle ausführbaren Dateien, Handbücher und den Quellcode herunterladen. Leider hat SourceForge seit 2013 nach und nach einige zwielichtige Praktiken übernommen: Überall tauchten Anzeigen mit gefälschten Download-Buttons auf und fügten den Installationsprogrammen einiger Projekte Mistware hinzu. Im Jahr 2015 war die Website etwa 15 Tage lang offline: Es wurde klar dass es an der Zeit war etwas zu ändern, und wir migrierten das FidoCadJ-Projekt auf GitHub. Seit August 2015 erfolgt die gesamte Entwicklung auf FidoCadJ mit Git: Sie können gerne FidoCadJ forken und sich die Ressourcen ansehen. Allerdings bleibt die SourceForge-Seite bestehen, da jemand immer noch gerne Pakete von SourceForge lädt (aber der Quellcode ist nicht aktuell).

Eine umfassende Überarbeitung des bestehenden Codes wurde zwischen 2013 und 2014 von Kohta Ozaki und mir durchgeführt und dann 2015 schrittweise fortgesetzt. Dies ermöglichte die Portierung des Programms auf Android-Plattformen, was durch den Beitrag anderer Personen, insbesondere Dante Loi, ermöglicht wurde. Viele andere Menschen halfen damals. Leider ist die Arbeit an Android seit 2016 ins Stocken geraten, obwohl die Portierung erfolgreich war und eine Vorab-Beta bereit und verteilt wurde.

Wenn Sie der Community mit FidoCadJ helfen möchten, machen Sie sich keine Sorgen: Sie müssen kein erfahrener Java-Programmierer sein. Sie können beispielsweise die Benutzeroberfläche oder die Handbücher in eine neue Sprache übersetzen und die Dokumentation sorgfältig auf Inkonsistenzen, Tippfehler und Fehler prüfen. FidoCadJ ist derzeit auf Italienisch, Französisch, Englisch, Spanisch, Deutsch, Chinesisch, Japanisch, Griechisch, Tschechisch und Niederländisch verfügbar. Einige Übersetzungen sind möglicherweise veraltet und erfordern etwas Community-Arbeit. Durchsuchen Sie die Ausgaben auf GitHub (mit dem Tag “translation”) um zu sehen was benötigt wird. Es ist ziemlich einfach, Menüs und Befehle zu übersetzen. Wenn Sie also FidoCadJ in Ihrer Muttersprache sehen möchten, erstellen Sie eine neue “issue” auf GitHub!

Es gibt auch noch einiges zu tun um die Bibliotheken (zumindest die Standardbibliotheken) und natürlich dieses Handbuch zu überprüfen. Sie können auch mit der Standardbibliothek... arbeiten. Ich glaube dass nur die Hälfte der Zeit die ich mit FidoCadJ verbringe tatsächlich für die Codierung selbst aufgewendet wird. Der Rest wird für die Beantwortung von Benutzerfragen, das Schreiben und Verbessern der Dokumentation usw. aufgewendet. Auf GitHub können Sie sich an den Diskussionen beteiligen, Verbesserungen vorschlagen oder einen Fehlerbericht abgeben. Zögern Sie nicht mitzumachen:

<https://github.com/DarwinNE/FidoCadJ>

1.3 FidoCadJ und die Zukunft

Im April 2010 stieß ich fast zufällig auf einen Thread auf einer bekannten italienischen Website www.electroyou.it. Piercarlo Boletti schlug dort vor ein Schaltplanaufzeichnungstool direkt in das Forum zu integrieren, da Ähnliches bereits mit L^AT_EX-Gleichungen gemacht wurde. Da FidoCadJ in seinem Beitrag zitiert wurde, habe ich einen Account erstellt um im Forum zu schreiben und angeboten mitzuarbeiten. Ich war ein paar Tage mit dem sehr netten Webmaster in Kontakt und das System war fertig: ein einfaches Kopieren und Einfügen des FidoCadJ-Codes und ein paar Tags. Das Forum führt FidoCadJ automatisch auf seinen Servern aus und erstellt sofort ein Bild aus dem Code. Dieses Bild wird dann im Forumsbeitrag angezeigt, der Quellcode ist jedoch immer verfügbar. Auf diese Weise muss der Diskussionsleser nichts installiert haben, aber wenn Sie etwas ändern möchten, können Sie den Code abrufen mit wenigen Klicks erhalten.

Abbildung 1.1 zeigt einen Teil eines meiner Beiträge auf www.electroyou.it. Dieses System ist so leistungsstark und flexibel, dass ich als erster von der Begeisterung der Benutzer begeistert war!³

The screenshot shows a forum post titled "[12] Re: Orologi e orologiai" by DarwinNE. The post contains a message from TardoFreak and a detailed circuit diagram. The circuit diagram is a complex analog clock design using various transistors (Q1-Q8, Q10-Q12) and components like resistors (R1-R12, R25, R26), capacitors (C1-C4, C10, C11, C12), and diodes (D1-D4). The diagram includes power supplies (+17V, -17V) and output sections labeled "Rampa a gradini" and "Rampa lineare". A note at the bottom says "Sorgente FidoCadJ | Ingrandisci | Cos'è Fidocad".

Abbildung 1.1: Teile einen meiner Beiträge www.electroyou.it. Sie können den Zeitplan mit nur einem Klick vergrößern. Mit einem zweiten Klick erhalten Sie sofort den Quellcode, den Sie in FidoCadJ einfügen können um ihn zu ändern.

³Wenn Sie Italienisch lesen können, finden Sie hier einen Artikel den ich geschrieben habe: <http://www.electroyou.it/darwinne/wiki/fidocad>

FidoCadJ hat sich auch für Zeichnungen als nützlich erwiesen, die nicht direkt mit der Elektronik zu tun haben (siehe Anhänge B und C, um zu sehen was ich meine).

Der Erfolg, den FidoCadJ auf www.electroyou.it hatte, löste einige Anfragen aus, ein ähnliches System auf anderen Plattformen und insbesondere auf www.grix.it, einer weiteren bekannten italienischen Elektronik-Website zu implementieren. FidoCadJ wird auch von der [Matematicamente](#)-Community verwendet. Im Fall von www.grix.it konnte der Server kein Java-Programm ausführen, daher wurde die FidoReadPHP-Klasse geschrieben. Es kann auf einem PHP-Interpreter ausgeführt werden um die Bilder aus dem FidoCadJ-Quellcode abzurufen. Das FidoReadPHP-Projekt ist Open Source und auf SourceForge verfügbar:

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

aber leider wird es im Moment nicht aktiv gepflegt. Das Ergebnis ähnelt in gewisser Weise dem, was es von www.electroyou.it erhalten hat, auch wenn die grafischen Fähigkeiten von PHP im Vergleich zu Java recht eingeschränkt sind⁴. Abbildung 1.2 zeigt ein Beispiel einer Zeichnung, die mit FidoReadPHP erstellt wurde.

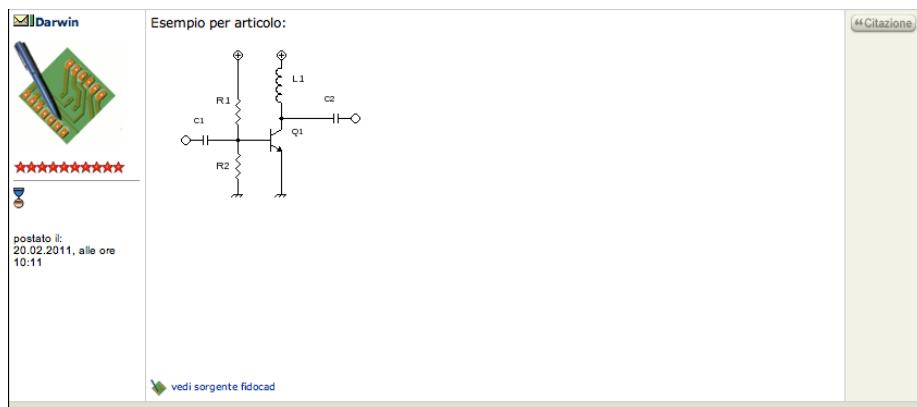


Abbildung 1.2: Ein Beispiel einer FidoCadJ-Zeichnung die in einen www.grix.it-Forumsbeitrag integriert ist. Den Quellcode der Zeichnung erhalten Sie mit nur einem Klick.

Ich denke, dass die Zukunft von FidoCadJ als komplettes CAD-System für die Elektronik nicht komplexer werden wird. Stattdessen sollte die Integration in Diskussionsgruppen und Foren weiterentwickelt werden. Die Erfahrung zeigt, dass dies möglich ist und die Begeisterung der Nutzer weckt.

⁴Auch hier gilt: Wenn Sie Italienisch lesen können, finden Sie hier einen Artikel:
<http://www.grix.it/viewer.php?page=9335>

Kapitel 2

Zeichnen mit FidoCadJ

Erfahrene Mac-Benutzer werden feststellen, dass sich alle Menüs an ihrem eigenen Platz befinden!

Die Verwendung von FidoCadJ sollte für diejenigen, die bereits ein Vektorzeichnungsprogramm verwenden, recht intuitiv sein. Ein Screenshot des Programms das unter macOS läuft ist in Abb. 2.2 dargestellt; Bei der Verwendung auf anderen Betriebssystemen können einige Details abweichen (z.B. Abb. 2.3 zeigt das Ergebnis mit Aussehen und Erfahrung Metal auf einem Raspberry Pi 400), aber die Philosophie bleibt dieselbe. Wir werden die Funktionen des Programms sowie die Grundelemente (die Grundelemente) beschreiben, die es ausmacht Erstellen einer FidoCadJ-Zeichnung. Unten sehen Sie ein Bild, das die drei Symbolleisten und ihre Darstellung in FidoCadJ zeigt.



Abbildung 2.1: Die drei Menüleisten von FidoCadJ 0.24.8.

2.1 Zeichenwerkzeuge

In der Symbolleiste oben im Fenster finden wir die am häufigsten verwendeten Funktionen zum Erstellen und Bearbeiten einer Zeichnung. Table 2.1 zeigt eine kurze Zusammenfassung der Funktionalitäten, der Befehle und der möglichen Aktionen. Sie werden feststellen, dass eine einmal gedrückte Symbolleistenschaltfläche an dieser Position verbleibt, bis eine andere Symbolleistenfunktion ausgewählt wird. In der Symbolleiste können wir das Zeichenelement auswählen.¹ Die Arbeitsbereichsleiste zeigt rechts die aktuelle Arbeitsebene an. (Weitere Informationen finden Sie unter 2.3)

Dieses Verhalten der Benutzeroberfläche wird allgemein als "Drucktasten"-d bezeichnet und ist von den alten Röhrenradios und Schaltern inspiriert, die bis in die 1970er Jahre in Mode waren.

Die Symbolleiste ist teilweise anpassbar. Wir können zwischen nur Symbolen oder Symbolen mit Beschreibung wählen. Die Icons sind in zwei Größen erhältlich. Um diese Einstellungen zu ändern, können Sie das Menü "Datei/Op-

¹Weitere Informationen zu den Zeichnungselementen finden Sie unter 3.3.

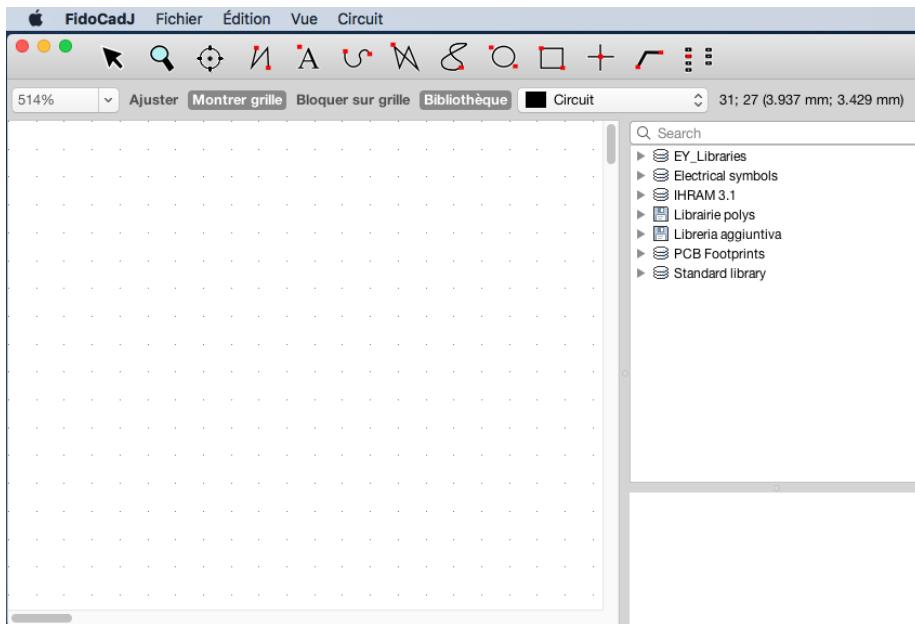


Abbildung 2.2: Eine FidoCadJ 0.24.8-Sitzung, die auf einem macOS 10.13 High Sierra läuft. Anhang A beschreibt die Besonderheiten der Version für Macintosh-Computer.

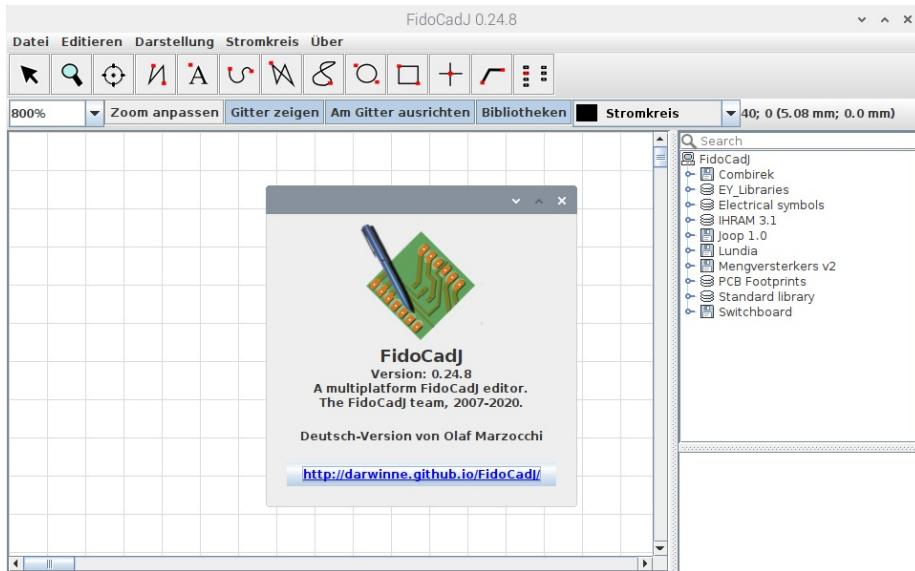


Abbildung 2.3: Eine FidoCadJ 0.24.8-Sitzung, die auf einem Raspberry Pi mit Looks and Feel von Metal läuft.

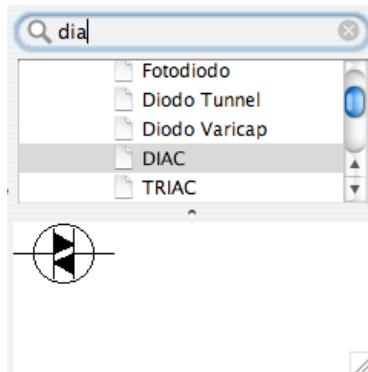


Abbildung 2.4: Die Schnellsuchfunktion für installierte Bibliotheken.

tionen” in der Menüleiste auswählen.² Solche Einstellungsänderungen werden erst nach einem Neustart von FidoCadJ wirksam, da es nicht vorstellbar ist, dass wir diese Einstellungen jeden Tag ändern möchten. Die Abbildungen 2.2 und 2.3 zeigen die Symbolleiste die so konfiguriert ist dass kein Text und Symbole in ihrer normalen Größe angezeigt werden. In der Arbeitsbereichsleiste können wir die Schaltflächen “Zoom anpassen”, “Gitter zeigen”, “Am Gitter ausrichten” und “Bibliotheken” sehen. Die erste ermöglicht die automatische Auswahl der am besten geeigneten Zoomeinstellungen, um die gesamte Zeichnung auf dem Bildschirm anzuzeigen. Die zweite Option schaltet zwischen einem sichtbaren und einem unsichtbaren Raster um. Der dritte Button steuert das Snapping, also die automatische Ausrichtung von Elementen am Raster. Letzteres ermöglicht das Ändern der Sichtbarkeit des Bibliotheksbereichs auf der rechten Seite des Fensters. Wenn Sie die Elemente sorgfältig ausrichten müssen, kann es hilfreich sein, bei Verwendung der Cursortasten **Alt** gedrückt zu halten: Die ausgewählten Objekte werden als einzelne Einheit in die gewünschte Richtung verschoben.

Auf der rechten Seite des Hauptfensters von FidoCadJ werden die Symbole (auch als Makros bezeichnet) in den geladenen Bibliotheken aufgelistet und als Baum dargestellt. Um ein Element aus der Bibliothek in die Zeichnung einzufügen, wählen Sie es aus der Liste aus und klicken Sie auf die Zeichnung. FidoCAD-Bibliotheken enthalten alle Standardsymbole, die in elektrischen Diagrammen verwendet werden, und eine große Auswahl an Footprints zum Zeichnen von Leiterplatten.

Sie können auch eine Schnellsuche in den Bibliotheken durchführen. Geben Sie einfach etwas in das Textfeld ein, das direkt über der Bibliotheksstruktur erscheint (siehe Abbildung 2.4). Sie können durch die Ergebnisse navigieren, indem Sie “return” eingeben.

Abbildung 2.5 zeigt ein Beispiel dafür, was durch einen Doppelklick im Auswahlmodus auf ein Zeichenelement (in diesem Fall ein Transistorsymbol) erhalten werden kann. In diesem Fenster ist es möglich, alle Parameter (Koordinaten, Rotation...) jedes Zeichenelements zu ändern. Das Aussehen dieses Fensters hängt vom ausgewählten Element ab.

² Außer unter macOS, wo dieses Element im FidoCadJ-Menü unter “Einstellungen” zu finden ist.

Tabelle 2.1: Zusammenfassung der in FidoCadJ verfügbaren Zeichenbefehle. Die in der Spalte ganz links angezeigte Taste ermöglicht eine schnelle Auswahl mit der Tastatur. Klicken Sie mit der rechten Maustaste in einen der Grundelement-Platzierungsmodi, um auf dessen Eigenschaften zuzugreifen.

Taste	Befehl	Verwendung
A, ESC oder Leerzeichen	► WÄHLEN	Wählen Sie ein oder mehrere grafische Elemente aus. Verwenden Sie Control (Befehl in macOS), um mehrere Elemente auszuwählen oder nur ein Element abzuwählen. Klicken und ziehen Sie um mehrere Elemente in einem Bereich auszuwählen. Verwenden Sie R um die ausgewählten Elemente zu drehen. Verwenden Sie S um die ausgewählten Elemente zu spiegeln. Doppelklicken Sie auf ein Element um seine Eigenschaften zu ändern.
	❖ ZOOM	Klicken Sie mit der linken Maustaste, um die Zoomstufe zu erhöhen. Klicken Sie mit der rechten Maustaste, um es zu verkleinern.
	❖ BEWEGEN	Klicken Sie auf die Zeichnung und bewegen Sie die Maus um die Zeichnung zu verschieben.
L	↙ GERADE	Fügen Sie eine Linie oder Reihe von Linien ein. Drücken Sie Esc oder doppelklicken Sie, um das Einfügen abzuschließen.
T B P	• A TEXT • B BÉZIER • P POLYGON	Fügen Sie eine Text ein. Zeichnen Sie eine Bézier-Kurve Zeichnen Sie eine geschlossene oder offene Polygonkurve. Doppelklicken oder Esc drücken um das Einfügen neuer Scheitelpunkte abzuschließen.
K	❖ KURVE	Offene oder geschlossene natürliche Kurve. Doppelklicken oder Esc drücken um das Einfügen neuer Scheitelpunkte abzuschließen.
E	○ ELLIPSE	Zeichnen Sie eine geschlossene oder offene Ellipse (halten Sie in macOS Control oder Befehl gedrückt, um einen Kreis zu erstellen).
G	□ RECHTECK	Zeichnen Sie ein geschlossene oder offene Rechteck (halten Sie in macOS Control oder Befehl gedrückt, um ein Quadrat zu erhalten).
C I	⊕ VERBINDUNG ✓ LEITERBAHN	Einen elektrischen Anschluss zeichnen. Eine Leiterbahn zeichnen. Die Standardbreite kann über den Dialog geändert werden, der über das Menü “Datei/Optionen” zugänglich ist.
Z	■ PCB PAD	Ein Pad legen. Die Standardgröße kann über das Menü “Datei/Optionen” geändert werden.



Abbildung 2.5: Dialogfeld für die Parameter eines Symbols in einer FidoCadJ-Zeichnung. Das Betriebssystem ist Raspberry Pi OS, auf Deutsch. Drücken Sie die Taste unten links, um Symbole einzugeben oder Hinweise zu erhalten.

2.2 Ein einfaches Schema

Als Beispiel für die Verwendung dieser Anwendung zeigen wir, wie das einfache elektrische Diagramm der Figur gezeichnet wird [2.6](#).

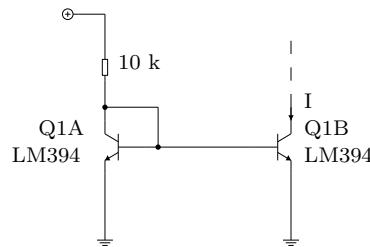


Abbildung 2.6: Das Referenzschema: Stromspiegel mit NPN-Transistoren.

Sobald FidoCadJ ausgeführt wird, erstellen Sie eine neue Zeichnung über die Menüleiste und das Menü "Datei/Neu". Anschließend platzieren wir zunächst die beiden Transistoren im Zeichenbereich, um den herum unser Schaltplan aufgebaut ist. Dazu benötigen wir die Symbole (auch Makros genannt), die sich in der Standardbibliothek befinden und standardmäßig geladen werden. Die Bibliothek befindet sich auf der rechten Seite des Fensters. Das von uns verwendete Symbol heißt "NPN-Transistor" und ist in der Kategorie "Dioden und Transistoren" der "Standardbibliothek" enthalten. Durch Klicken auf den Namen des gewünschten Symbols wird das Symbol ausgewählt und kann dann an einer beliebigen Stelle in der Zeichnung platziert werden (FidoCadJ zeigt ein Beispiel), indem Sie ein zweites Mal auf die gewünschte Stelle klicken. Wir sollten uns jetzt in einem ähnlichen Stadium wie in der Abbildung [2.7](#) befinden.

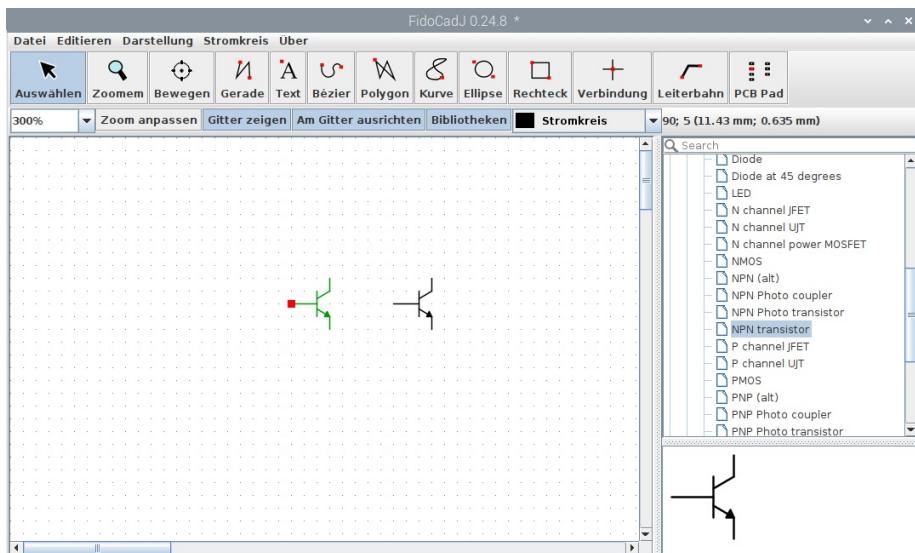


Abbildung 2.7: Wir beginnen mit der Zeichnung einiger Transistoren.

Wir können feststellen, dass der Bipolartransistor auf der linken Seite nicht richtig ausgerichtet ist. Um das Problem zu lösen, klicken Sie einfach auf "Auswahl" in der Symbolleiste, dann auf den Transistor (der grün hervorgehoben wird, wobei ein Prüfpunkt durch ein kleines rotes Quadrat gekennzeichnet ist) und drücken Sie dann **S**, um eine gespiegelte Version zu erhalten. Sie können auch **S** drücken, wenn Sie das Symbol in den Bibliotheken ausgewählt haben und es in der Zeichnung platzieren möchten. Wir erhalten also ein ähnliches Ergebnis wie in der Abbildung 2.8.

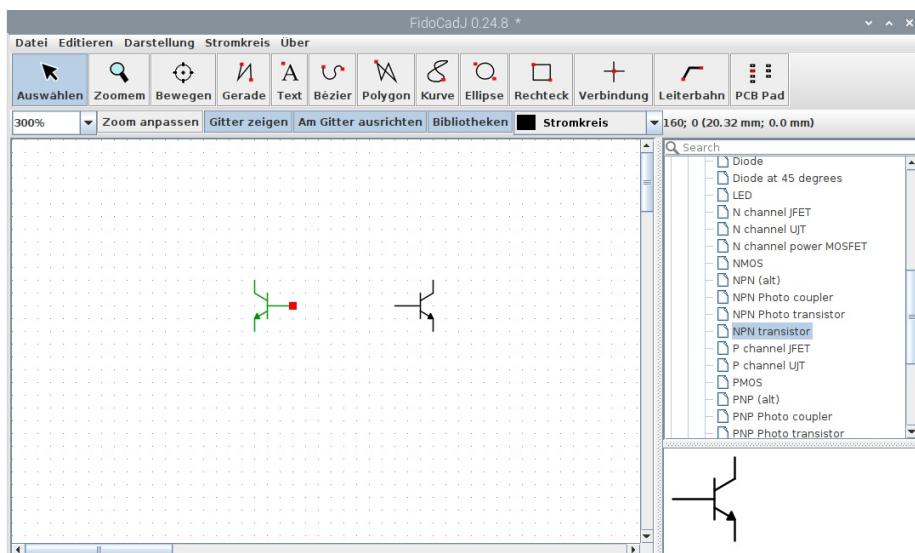


Abbildung 2.8: Wählen Sie den Transistor auf der linken Seite aus und spiegeln Sie ihn, indem Sie **S** drücken.

Mit dem “Gerade”-Werkzeug aus der Symbolleiste können wir ein paar elektrische Verbindungen zeichnen, bis wir feststellen dass wir zu nah an den Rändern der Zeichenfläche zu zeichnen begonnen haben. Dies kann leicht gelöst werden, indem die gesamte Zeichnung ausgewählt wird: Im Modus “Auswählen” können wir auf die obere linke Ecke der Zeichnung klicken und bei gedrückter linker Maustaste den Cursor in die untere rechte Ecke ziehen. Es erscheint ein Rechteck mit einer grünen Umrundung, was darauf hinweist, dass wir versuchen alle darin enthaltenen Elemente auszuwählen. Da wir alles, was wir bisher gezeichnet haben, verschieben möchten, müssen wir sie zuerst alle auswählen (siehe Abbildung 2.9). Jetzt, immer noch im Auswahlmodus, können wir auf jedes ausgewählte Element klicken um die gesamte Auswahl an die richtige Position zu ziehen.

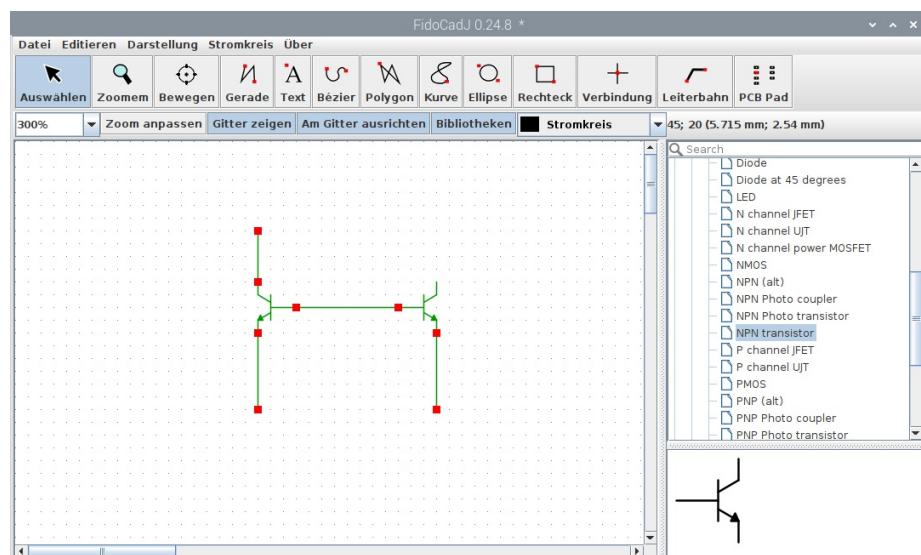


Abbildung 2.9: Wir sind zu nah am oberen Rand des Blattes: Wählen Sie die gesamte Zeichnung aus und verschieben Sie sie in die Mitte.

Anschließend können wir mit der Platzierung der anderen Teile der Schaltung fortfahren, eines Widerstands (Standard Library/Discrete devices/Resistor) und der Beschriftung für die positive Stromversorgung (Standard Library/Basic symbols/ Terminal +). Letzteren müssen wir drehen, um ihn in die gewünschte Position zu bringen. Wir können es auswählen und die Taste **R** drücken, bis wir das gewünschte Ergebnis erhalten. Wir sollten jetzt eine Zeichnung haben, die der in der Abbildung ähnelt 2.10. Um das Schema zu vervollständigen, müssen wir nur noch die Textzeichenfolgen und den Pfeil hinzufügen um die Richtung des Stroms anzusehen. Für Letzteres gibt es ein Symbol namens “Arrow”, das in der “Standard library/Basic symbols” enthalten ist. Um den Text zu platzieren, können wir auf das Symbol doppelklicken (siehe Abbildung 2.5).

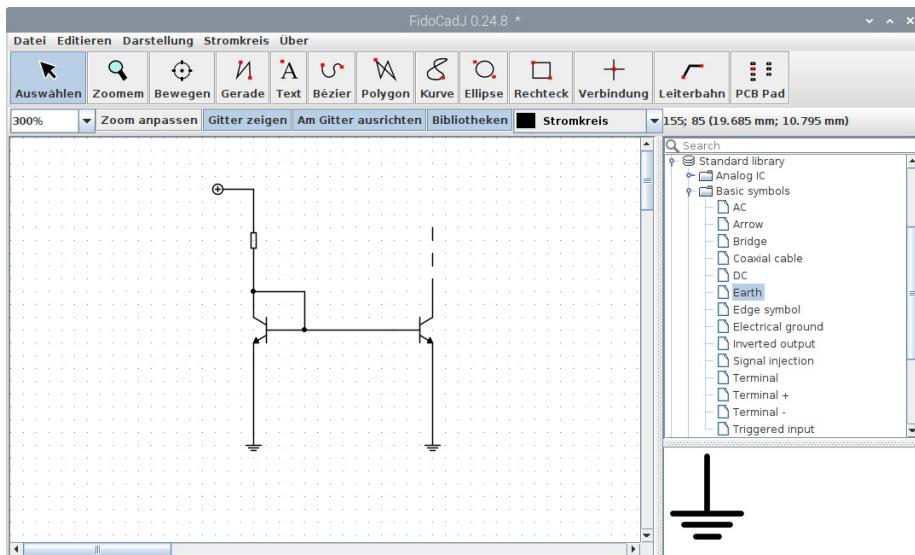


Abbildung 2.10: Die Schaltung ist fast fertig.

Die Teilenummer und der verwendete Name der Transistoren werden in den Feldern "Name" und "Wert" angegeben. Die Funktion, die einem Element einen Namen und einen Wert hinzufügen kann, ist eine Erweiterung die mit Fido-CadJ eingeführt wurde. Weitere Informationen zur Kompatibilität finden Sie im Abschnitt 3.4. Die Textgröße beträgt vertikal 4 Einheiten und horizontal 3 Einheiten. Die vollständige Schaltung ist in Abbildung 2.11 dargestellt.

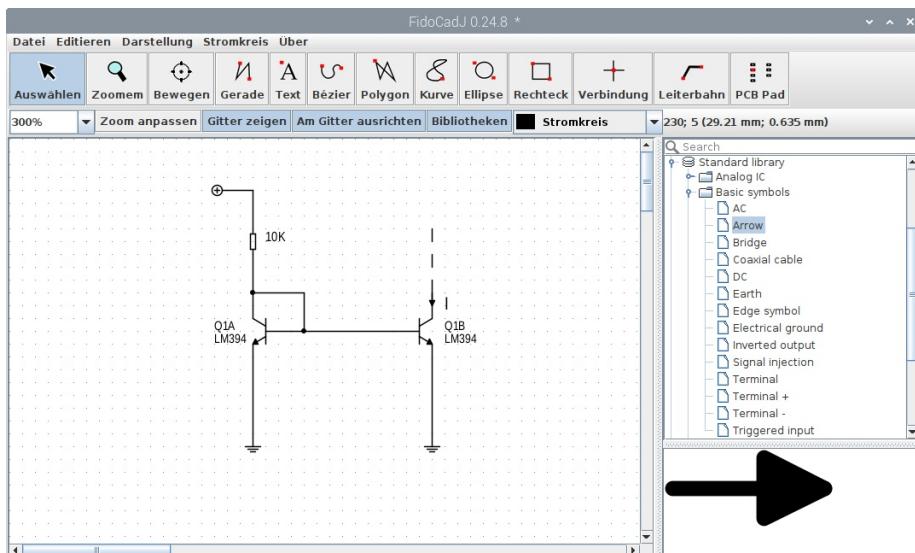


Abbildung 2.11: Die komplette Schaltung.

Tabelle 2.2: Dieser Code beschreibt die Schaltung aus unserem Beispiel.

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

Aus Neugier ist oben der Code zu finden, der die Schaltung unseres Beispiels beschreibt. Um auf diesen Code zuzugreifen, wählen Sie einfach “Quelle anschauen” aus dem Menü “Stromkreis”. Jetzt können wir unsere Schaltung kopieren und in eine E-Mail-Nachricht, in eine Newsgroup oder in einen Forumbeitrag einfügen.

Wenn Sie sich für das von FidoCadJ verwendete Exportformat interessieren, finden Sie eine ausführliche Beschreibung im Kapitel 3. Es ist jedoch nicht notwendig, den “Quelle anschauen” aus dem Dialogfeld zu verwenden; Sie können einfach die gesamte Zeichnung auswählen, sie kopieren (indem Sie “Editieren/-Kopieren” auswählen oder **Ctrl+C** drücken) und sie in den Beitrag einfügen die wir schreiben, und der Code wird automatisch hinzugefügt. FidoCadJ passt normalerweise eine Diagonalverschiebung auf die kopierten Elemente an; Die Breite und Höhe von x und y entsprechen dem Rasterabstand. Das Standardverhalten besteht darin um so vorzugehen die eingefügten Elemente von den Originalelementen zu unterscheiden. Da dies in manchen Situationen problematisch sein kann, kann diese Einstellung in den Programmoptionen geändert werden.

2.3 Die Schichten

Eine Möglichkeit sich Schichten vorzustellen, entsteht indem man an eine Zeichnung übereinander liegender Acetatfolienblätter denkt. Jede Ebene zeichnet sich durch eine andere Farbe aus und kann sichtbar oder verborgen sein. Dieser Ansatz ist in vielen CAD-Paketen üblich, da er die einfache Anzeige und Verwaltung verschiedener übereinanderliegender Teile der Zeichnung ermöglicht, beispielsweise in einem Leiterplattenentwurf. FidoCadJ erlaubt bis zu 16 Ebenen, nummeriert von 0 bis 15. Herkömmlicherweise haben einige Ebenen einen bestimmten Zweck. Schicht Null wird üblicherweise für elektrische Schaltpläne verwendet, Schicht 1 für die Kupferlötseite, Schicht 2 für die Seite der Kupferkomponenten und Schicht 3 für den Siebdruck. Die übrigen Schichten haben keinen vorgegebenen Zweck und können frei verwendet werden. Name und Farbe jeder Ebene können über die Menüleiste und das Menü “Darstellung/Ebenen” angegeben werden. Im selben Menü können wir auch die Ebenen auswählen, die wir auf dem Bildschirm sehen oder ausdrucken möchten. Die Reihenfolge der Ebenen ist wichtig, da die Wahrscheinlichkeit höher ist, dass Ebenen mit niedrigeren Nummern gezeichnet werden: Elemente auf aufeinanderfolgenden Ebenen überdecken die bereits gezeichneten.

2.4 Das Gitter

Die logische Einheit in FidoCadJ beträgt 5 Mil (127 Mikrometer) und “halbe Einheiten” sind nicht zulässig, was bedeutet dass die Koordinaten jedes Grafiikelements ganze Zahlen sein müssen. Dadurch ist es möglich eine Auflösung zu erhalten die fein genug ist um einen Schaltplan und die meisten Leiterplatten zu zeichnen. Um das Zeichnen zu erleichtern, ermöglicht der Anwendung jedoch ein größeres Raster festzulegen und die Maus dazu zu zwingen sich am nächstgelegenen Punkt des Rasters auszurichten. Um diese Funktionalität zu aktivieren, gibt es zwei Schaltflächen in der Arbeitsbereichsleiste , “Gitter zeigen” und “Am Gitter ausrichten” die Folgendes ermöglichen, zwischen sichtbarem/verborginem Raster zu wechseln und den Mauszeiger auf dem Raster zu erzwingen oder frei zu bewegen. Der Rasterschritt kann über ein Dialogfeld ausgewählt werden, das über die Menüleiste und das Menü “Datei/Optionen” Tab “Zeichnung” zugänglich ist.

2.5 Eine einfache Leiterplatte

Um das bisher Gelernte in die Praxis umzusetzen, sehen wir uns an wie man eine einfache Leiterplatte entwirft. Im Gegensatz zu anderer elektronischer CAD-Software, die sehr leistungsfähig, aber manchmal recht schwierig zu verwalten ist, bietet FidoCadJ eine elektronische Version der guten alten R41-Transfers oder des Bandes auf Mylar-Blättern. Bei der Arbeit am Computer ist es selbstverständlich möglich, die gesamte Flexibilität der Maschine zu nutzen.

Es ist zu beachten, dass der Entwurf einer Leiterplatte, insbesondere wenn es sich um eine komplexe handelt, keine leichte Aufgabe ist. Die Funktionen autoplacer und autorouter versprechen Wunder in den Werbebroschüren großer

Ein paar Kritzeleien mit Papier und Bleistift (und vielen Radiergummis) sparen Zeit, da man mit Hilfe des Computers eine klare Idee entwickelt.

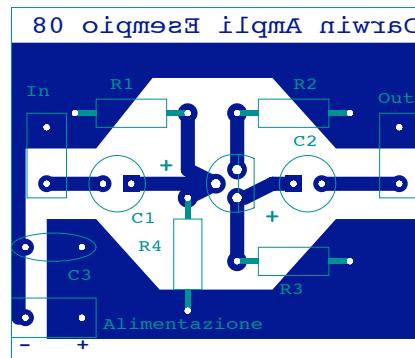


Abbildung 2.12: Eine sehr einfache Verstärkerstufe mit einem NPN-Transistor in Emitterschaltung.

CAD-Unternehmen, aber es besteht kein Zweifel, dass dies immer noch Aufgaben sind bei denen das Benutzererlebnis immer noch eine wichtige Rolle spielt. FidoCadJ ist eine sehr direkte und schnelle Möglichkeit um kleine Leiterplatten im DIY-Maßstab zu zeichnen. Dieser Ansatz ist sehr einfach, aber wenn Sie eine sehr komplexe Leiterplatte entwerfen müssen ist FidoCadJ NICHT das richtige Werkzeug für Sie: Verwenden Sie zum Beispiel das ausgezeichnete KiCad: Die zusätzliche Komplexität ist gerechtfertigt. Wir werden hier sehen, wie man eine sehr einfache, aber vollständige Zeichnung erstellt.

Ich würde vorschlagen, mit einer klaren Vorstellung davon zu beginnen wo jedes Teil platziert werden soll und wie alle Spuren so gezeichnet werden, dass sie sich so wenig wie möglich kreuzen.

Hier schummeln wir ein wenig und gehen von dem Ergebnis aus das wir erreichen wollen, wie in Abbildung 2.12 gezeigt. Es handelt sich um einen einfachen Common-Emitter-Verstärker, der um einen NPN-Transistor vom Typ BC547 oder ähnlichem herum aufgebaut ist. Es ist hilfreich, sich die Platine von der Komponentenseite aus so vorzustellen, als wäre sie transparent. Der Siebdruck mit der Teilezeichnung ist hierfür sehr nützlich, obwohl wir ihn für ein Do-it-yourself-Projekt wahrscheinlich nicht auf die eigentliche Platine drucken möchten.

Als Erstes würde ich vorschlagen, alle Komponenten so gut wie möglich zu platzieren. In unserem Beispiel sind dies der Transistor (aus der Bibliothek "PCB Footprints/3 terminals semiconductors/TO92"), die Widerstände ("PCB Footprints/Resistor/1/4 W 0,4i resistor") und der Elektrolytkondensatoren ("PCB Footprints/Electrolytic capacitors/Vert. diam. 5 mm, 2.5 mm pitch"). Um die Platine abzugrenzen, kann es hilfreich sein, ein leeres Rechteck auf der Siebdruckebene (Ebene Nr. 3) zu platzieren. Dazu können wir das Grundelement *Rechteck* verwenden um sicherzustellen dass wir zuerst die richtige Ebene ausgewählt haben. Wir sollten ein ähnliches Ergebnis wie im Bild erhalten 2.13.

Anschließend können wir die Kupferbereiche einführen, die für die positive und negative Stromversorgung sorgen. Diese werden durch Zeichnen eines Polygon (unter Verwendung des Grundelements *Polygon*) und durch Doppelklicken in der Nähe der Grenze im Auswahlmodus angegeben, um FidoCadJ (über das Dialogfeld) mitzuteilen, dass wir ein gefülltes Polygon haben möchten. Stellen

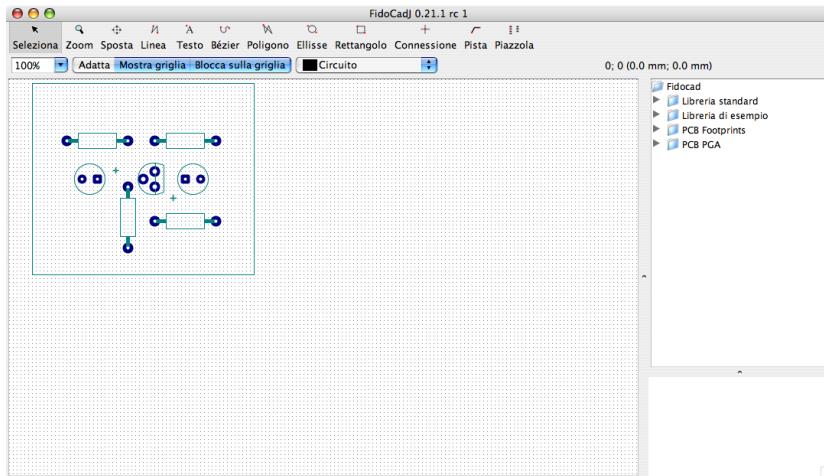


Abbildung 2.13: Die wichtigsten Teile sind auf der Leiterplatte untergebracht.

Sie vor dem Platzieren des Polygons sicher, dass die aktuelle Ebene diejenige ist, auf der wir den Kupferbereich platzieren möchten (Ebene Nr. 1 oder Kupferlotseite). Die Verwendung eines gefüllten Bereichs für die Stromversorgungsleitungen kann sinnvoll sein, um sicherzustellen dass diese Verbindungen geringe Streuinduktivitäten aufweisen. Wir sollten in der Lage sein, das Ergebnis wie in der Abbildung gezeigt zu reproduzieren 2.14. Um die elektrischen Verbindun-

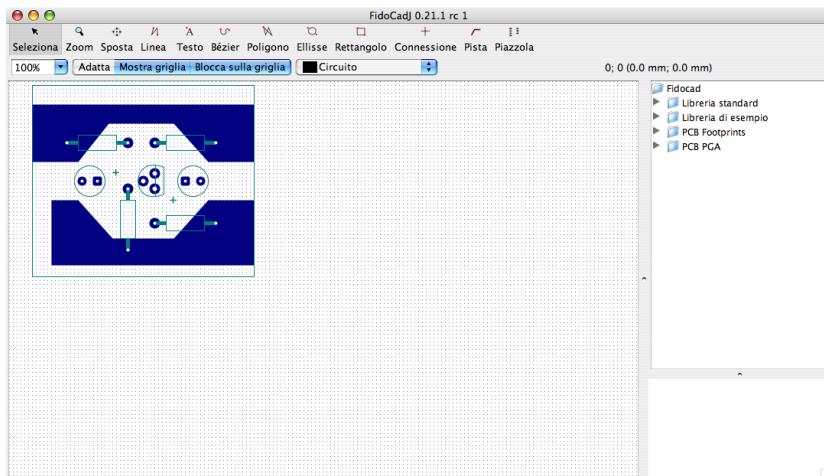


Abbildung 2.14: Stromanschlüsse mit Polygonlinien hinzugefügt.

gen zu vervollständigen, können wir das Grundelement *Leiterbahn* verwenden. Ich habe eine Leiterbahndicke von 10 Einheiten (1,27 mm) gewählt was den Lötprozess erleichtert, siehe Abbildung 2.15. Uns ist klar, dass wir ein paar Anschlüsse benötigen: einen für den Eingang, einen für den Ausgang und einen für die Stromversorgung. Wir können die für Polyesterkondensatoren vorgesehene Grundfläche verwenden. Es hat wahrscheinlich die richtigen Abmessungen. Vergessen wir nicht dass FidoCadJ als Ersatz für die Transfers gedacht ist...

Seien Sie vorsichtig mit den Leiterbahnbreiten: Alles was auf dem Bildschirm wie eine Autobahn aussieht, ist wahrscheinlich eine so dünne Leiterbahn, dass es sich beim Löten von der Platine löst.

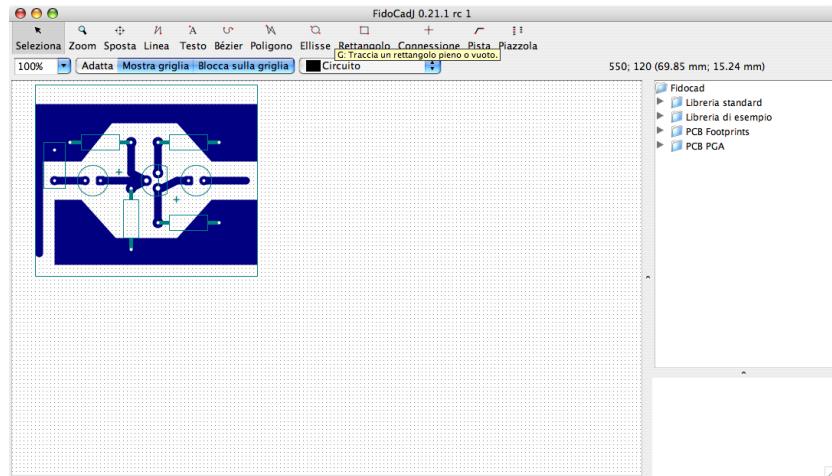


Abbildung 2.15: Die verbleibenden Leiterplattenanschlüsse wurden hinzugefügt.

Wir können auch ein “+” und ein “-“ auf die Kupferschicht setzen und einen Keramikkondensator parallel zur Stromversorgung schalten. Lassen Sie uns auch Text darüber platzieren. Um auf der Kupferschicht zu schreiben, ist es nach Auswahl der richtigen Schicht auch notwendig, alle Texte zu spiegeln. Dies lässt sich ganz einfach im üblichen Eigenschaften dialog erreichen, indem Sie im Auswahlmodus auf die Zeichenfolge doppelklicken, die Sie ändern möchten. Möglicherweise müssen Sie ein wenig experimentieren, um die richtige Größe für die Zeichen zu erhalten. Um Ihnen eine Vorstellung zu geben, ist es nützlich, ein Verhältnis von 3/4 zwischen den horizontalen und vertikalen Abmessungen der Zeichen einzuhalten. Figure 2.16 zeigt das Ergebnis, das mit 11 Einheiten für die horizontale und 18 Einheiten für die vertikale Dimension des Textes erzielt wurde.

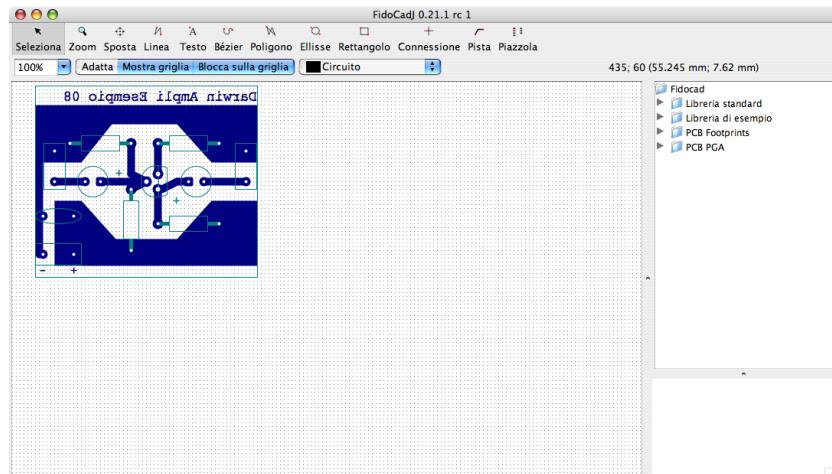


Abbildung 2.16: Die Platine ist fast fertig.

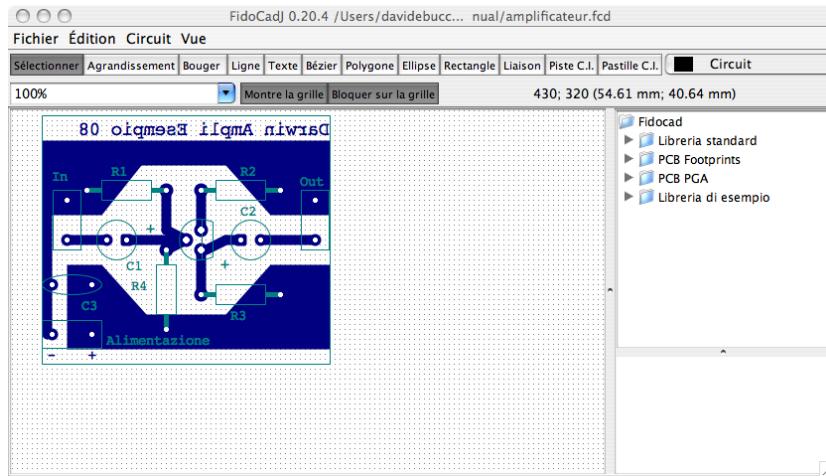


Abbildung 2.17: Mit dem Siebdruck ist die Arbeit abgeschlossen.

Zu diesem Zeitpunkt fehlt nur noch der Text mit dem Namen jeder Komponente, der auf Ebene 3 (Komponentenseite) platziert werden kann. Das Programm mit der fertigen Platine ist in Abbildung 2.17 dargestellt. Sobald die Zeichnung fertig ist, müssen wir sie wahrscheinlich auf einer Folie ausdrucken, um sie mit einer Entwicklungsmaschine zu verwenden, oder um sie mit anderen Methoden wie “Press&Peel” Transfersysteme zu verwenden. Dazu müssen wir alle Ebenen, die wir nicht drucken möchten, unsichtbar machen. Dies geschieht über das Dialogfenster, das über die Menüleiste und das Menü “Darstellung/Ebenen” aufgerufen werden kann. In unserem Fall reicht es aus, Layer 3 mit der Komponentenseite auszublenden. Das Programm zeigt dann nur noch die Kupferschicht an.

Anschließend drucken wir alles aus, was auf dem Bildschirm angezeigt wird (natürlich NICHT an die Seitengröße angepasst, da wir die Abmessungen in der Zeichnung festlegen möchten) und achten dabei auf die Auswahl des Schwarz-weißdrucks, um den maximalen Kontrast zu gewährleisten. Abhängig von der für die Herstellung der Leiterplatte gewählten Technik kann es sinnvoll sein, die gesamte Zeichnung zu spiegeln. Da unsere Leiterplatte recht klein ist, nimmt sie in ihren tatsächlichen Abmessungen nur eine kleine Ecke des Blattes ein (unter der Annahme eines Standard-ISO-UNI-A4), wie wir in Abbildung 2.18 sehen können.



Abbildung 2.18: Die Leiterplatte, wie sie aussieht wenn sie auf ein ISO-UNI-A4-Blatt gedruckt (gespiegelt) wird.

Tabelle 2.3: Dieser Code beschreibt die Leiterplatte aus unserem Beispiel.

Zu Ihrer Information finden Sie unten den Code, der für die Platine des obigen Beispiels generiert wurde (Vorsicht bei langen Zeilen!):

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

Beim Zeichnen einer Leiterplatte ist es oft hilfreich, Abstände im Arbeitsbereich zu messen. Sie können beispielsweise die Leiterbahnbreite, den Abstand zwischen zwei Leiterbahnen oder die Gesamtgröße einer Leiterplatte überprüfen. FidoCadJ bietet (seit Version 0.23.2) eine Linealfunktion mit der Sie diese Aufgaben einfach ausführen können. Klicken Sie einfach mit der rechten Maustaste und ziehen Sie. Sie sollten ein grünes Lineal erhalten, wie in Abbildung 2.19 gezeigt. Wenn das Rechtsklicken und Ziehen in Ihrem System nicht einfach funktioniert, können Sie auch mit der linken Maustaste klicken und ziehen, während Sie die **Shift**-Taste gedrückt halten. Die gemessene Gesamtlänge wird

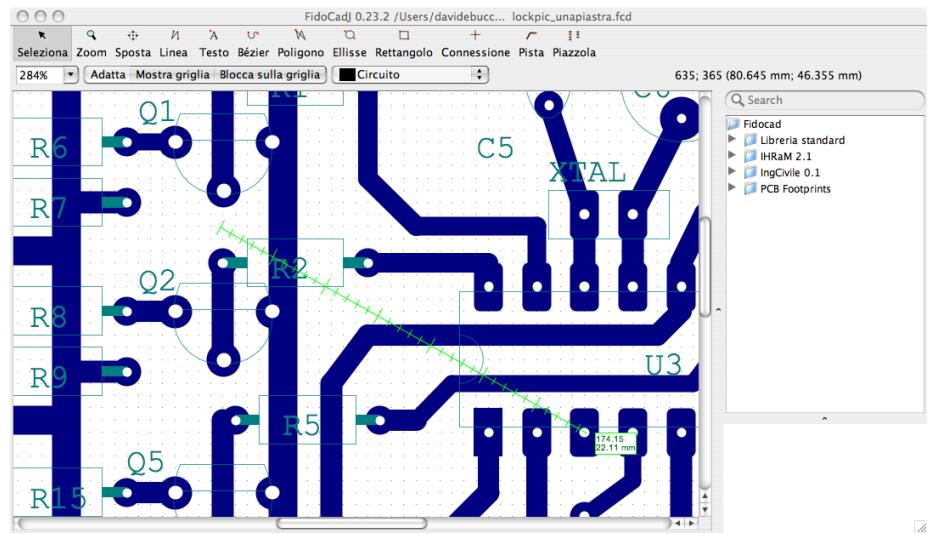


Abbildung 2.19: Klicken Sie mit der rechten Maustaste und ziehen Sie, um das FidoCadJ-Lineal zu aktivieren.

sowohl in logischen FidoCadJ-Einheiten als auch in Millimetern angezeigt. Dies ist nützlich, wenn die Zeichnung im 1:1-Modus gedruckt wird (z. B. für Leiterplatten).

2.7 Pfeil- und Strichstile

Mit FidoCadJ können Pfeilspitzen am Anfang und am Ende von Linien, Bézier-Kurven und natürlichen kubischen Splines gezeichnet werden. Außerdem können Sie festlegen, ob sich der Pfeil am Anfang oder am Ende eines Elements (oder an beiden) befinden soll, und bietet die Wahl zwischen verschiedenen Pfeilstilen. Die Pfeillänge kann negativ sein: In diesem Fall erstreckt sich der Pfeil über die Linie oder Kurve hinaus. Experimentieren Sie ruhig! Für technische oder mechanische Zeichnungen stehen auch einige Strichstile zur Verfügung. Figure 2.20 zeigt ein Beispiel, in dem ein elektrischer Schaltkreis (ein GIC) in einem gestrichelten Rechteck eingeschlossen ist. Es wird auch ein Pfeil am Ende einer Bézier-Kurve verwendet. Wenn Sie auf dieses Element doppelklicken, zeigt FidoCadJ einen Dialog wie in Abbildung 2.21. Sie bemerken, dass das Optionskästchen “Pfeil

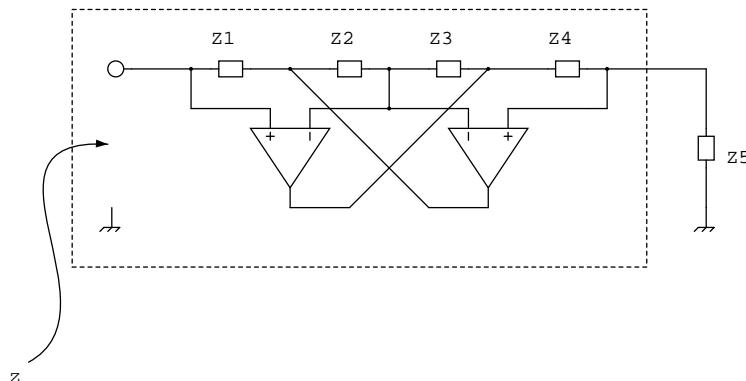


Abbildung 2.20: Eine elektrische Zeichnung (ein GIC von Antoniou) unter Verwendung einiger FidoCadJ-Erweiterungen.



Abbildung 2.21: Das in den Schemata der Abbildung verwendete Parameterfenster der Bézier-Kurve 2.20.

am Anfang” aktiviert ist. FidoCadJ zeichnet eine Pfeilspitze nach, indem es auf seine Ausrichtung achtet. Es gibt verschiedene Zeichenstile für den Pfeil sowie für den Strich. Versuchen Sie vielleicht, ein wenig mit ihnen zu spielen, um ihre Unterschiede zu erkennen. Wenn die Pfeillänge negativ ist, erstreckt sich der Pfeil über die Länge des Elements hinaus und zeigt auf dieses.

Die Möglichkeit, einen Strichstil auszuwählen und Pfeilspitzen zu setzen, war im ursprünglichen FidoCAD-Format nicht vorhanden. Leider bedeutet dies, dass FidoCadJ-Zeichnungen, die diese Funktionen verwenden, nicht vollständig abwärtskompatibel mit FidoCAD für Windows sind. Wenn Sie vollständige FidoCAD-Kompatibilität benötigen, können Sie die Option “Strikte FidoCad Kompatibilität” auf der Registerkarte “FidoCadJ Erweiterungen” des Fensters “FidoCadJ Einstellungen” aktivieren. Auf diese Weise erlaubt Ihnen das Programm nicht, grafische Elemente einzugeben, die zu Kompatibilitätsproblemen mit FidoCAD führen würden. Wenn Sie weitere Informationen zur Kompatibilität der neuen Grafiken mit FidoCAD benötigen, lesen Sie den Abschnitt 3.4.

2.8 Exportieren

Eines der wichtigsten Dinge an FidoCadJ ist für mich die Möglichkeit, einfache Schaltpläne für typografische Zwecke zu erstellen. Aus diesem Grund habe ich eine Funktion eingeführt, die den Export von Zeichnungen in verschiedene Dateiformate ermöglicht.

Ein Vektorformat speichert die Elemente, aus denen die Zeichnung besteht. Ein Bitmap-Format arbeitet mit einer Pixelmatrix.

Um die aktuelle Zeichnung zu exportieren, wählen Sie im Menüleiste “Datei” und den Befehl “Exportieren”. Die Tabelle 2.4 zeigt eine Liste der derzeit verfügbaren Grafikdateiformate. Für jedes Dateiformat gibt die Tabelle (aber auch der Exportdialog in FidoCadJ) an, ob es sich um ein Vektor- oder ein Bitmap-Format handelt³.

Bei Bitmap-Dateiformaten kann es sinnvoll sein, die Option “Anti-Aliasing” zu aktivieren, um den störenden Effekt der Quantisierung, der insbesondere bei diagonalen Linien sichtbar ist, zu reduzieren. Die Optionen Auflösung und “Anti-Aliasing” werden beim Exportieren in ein Vektordateiformat nicht verwendet. In diesem Fall können Sie stattdessen einen Skalierungsfaktor angeben.

Mit der Option “Schwarz&Weiß” kann jede sichtbare Ebene in sattem Schwarz gedruckt werden. Dies ist wichtig für die Vorbereitung von Filmen für Typografiezwecke oder eine Entwicklungsmaschine.

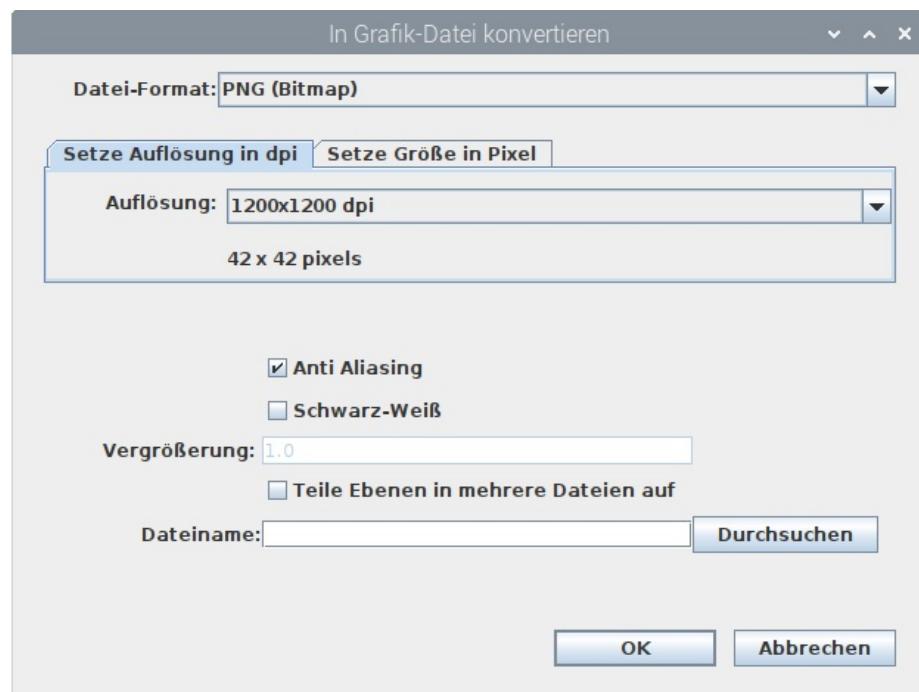


Abbildung 2.22: Das Setup-Menü für den Export in ein Bild

³Die Codestruktur von FidoCadJ ermöglicht das problemlose Hinzufügen eines anderen Dateiformats. Bitte kontaktieren Sie mich, wenn Sie an dem Projekt teilnehmen möchten.

Tabelle 2.4: Liste aller in FidoCadJ verfügbaren Exportdateiformate.

Format	Beschreibung
JPG	Allgegenwärtiges Bitmap-Format. Die verwendete Komprimierung ist verlustbehaftet und eignet sich daher nicht perfekt für den Export von FidoCadJ-Schaltplänen, da es zu sichtbaren Artefakten kommen kann.
PNG	Komprimiertes Bitmap-Format, geeignet zum Exportieren von Schaltplänen. Dies ist wahrscheinlich die beste Möglichkeit, eine FidoCadJ-Zeichnung zu exportieren, wenn ein Vektorformat nicht verwendet werden kann.
SVG	W3C-Standard-Vektorformat. Internetbrowser können es innerhalb einer Webseite anzeigen. Sehr gutes Format für Bilder und Schaltpläne. Es kann mit Anwendungen wie Inkscape verwendet werden, um die mit FidoCadJ erstellten Zeichnungen zu ändern. Beim Export von gedrehtem und gespiegeltem Text bestehen einige Einschränkungen.
EPS	Gekapseltes Postscript-Vektorformat. Wird in professionellen Grafikanwendungen verwendet und eignet sich zum Integrieren einer FidoCadJ-Zeichnung in ein L ^A T _E X-Dokument. Dies wurde verwendet, um Figur 2.12, Seite 16 in diesem Handbuch zu erhalten (durch eine PDF-Konvertierung, da ich PDFL ^A T _E X verwende).
PDF	Die sehr bekannte tragbare Dokumentendatei. Exportierte Dateien enthalten keine Schriftarten, daher kann die Textdarstellung auf einigen Plattformen leicht verändert sein.
PGF	Vektorformat zur direkten Verwendung in einem L ^A T _E X-Dokument bei Verwendung des <i>pgf</i> -Pakets, verfügbar im CTAN-Archiv. Diese Exportoption sollte Schaltpläne in ein bearbeitbares Skript exportieren. Textattribute werden nicht übersetzt. Dies ermöglicht die direkte Einführung von L ^A T _E X-Code in die Zeichnung und ist die in diesem Handbuch verwendete Technik, um Figur 2.6, Seite 10 zu erstellen.
SCR	Mit FidoCadJ kann eine Zeichnung in ein Skript exportiert werden, das in CadSoft Eagle importiert werden kann. Um diese Funktion nutzen zu können, ist die Installation der Bibliothek FidoCadJLIB.1br im Ordner 1br der aktuellen Installation von Eagle erforderlich. Die Bibliothek kann von der FidoCadJ-Website heruntergeladen werden. Zum Zeitpunkt des Verfassens dieses Artikels funktioniert diese Option nur mit Schaltplänen, die nur die gängigsten Symbole enthalten. Einige Zeichnungselemente wie Pads und Spuren sind nicht verfügbar und werden nicht exportiert.

2.9 Befehlszeilenoptionen

Die Anwendung wird als .jar-Datei verteilt, bei der es sich um ein Java-Archiv handelt.⁴ In vielen Betriebssystemen sollte ein Doppelklick auf die Datei ausreichen um die Anwendung auszuführen, vorausgesetzt, dass eine aktuelle Version von Java auf dem Computer installiert ist. In Oracles-Terminologie ist die sogenannte JRE oder Java Runtime Environment alles, was erforderlich ist, um ein in Java geschriebenes Programm auszuführen (aber nicht, um es zu schreiben, in diesem Fall wäre das SDK erforderlich...). Die zum Ausführen von FidoCadJ mindestens erforderliche Java-Version ist 9, die es bereits seit einigen Jahren gibt.

In manchen Fällen kann es nützlich sein, FidoCadJ über eine Befehlszeile auszuführen (das Terminal in den Unix-Systemen oder die MS-DOS in Windows). Dazu reicht es aus, den Befehl `java` mit der Option `-jar` auszuführen:

```
java -jar fidocadj-0.24.8.jar
```

Wenn in der Befehlszeile eine Datei angegeben ist, versucht FidoCadJ, diese zu öffnen. Zum Beispiel (auf einer Unix-Maschine):

```
java -jar fidocadj-0.24.8.jar ~/FidoCadJ/test.fcd
```

FidoCadJ läuft und versucht die Datei `~/FidoCadJ/test.fcd` (falls vorhanden). Es gibt noch einige interessante Dinge, die FidoCadJ tun kann. Option `-h` zeigt eine Auflistung der FidoCadJ-Optionen auf Englisch:

```
[davidebucci@davide-bucci-portable]$ java -jar fidocadj-0.24.8.jar -h
Dies ist FidoCadJ, Version 0.24.8.
Vom FidoCadJ-Team, 2007-2023.

Verwendung: java -jar fidocadj-0.24.8.jar [-options] [Datei]
wobei die Optionen Folgendes umfassen (übersetzt):

-n      Startet die grafische Benutzeroberfläche nicht (Headless-Modus)

-d      Legen Sie das externe Bibliotheksverzeichnis fest.
        Verwendung: -d dir
        wobei 'dir' der Pfad des Verzeichnisses ist, das Sie verwenden möchten.

-c      Konvertieren Sie die angegebene Datei in ein grafisches Format.
        Verwendung: -c sx sy eps|pdf|svg|png|jpg|fcd|sch ausgehende_Datei
        Wenn Sie diese Befehlszeilenoption verwenden, *müssen* Sie eine FidoCadJ-Datei zum Konvertieren angeben.
        Eine Alternative besteht darin, die Auflösung in Pixel pro logischer Einheit anzugeben, indem Sie ihr den Buchstaben 'r' (ohne Leerzeichen) voranstellen, anstatt sx und sy anzugeben.
        HINWEIS: Die Richtigkeit der Dateierweiterung wird überprüft, sofern nicht die Option -f angegeben ist.

-m      Wenn eine Datei in ein Vektorbildformat exportiert wird, teilen Sie die Ebenen auf und schreiben Sie eine Datei für jede Ebene. Der Dateiname wird durch Hinzufügen von _ gefolgt von der Layer-Nummer zum angegebenen Dateinamen ermittelt. Der folgende Befehl erstellt beispielsweise die Dateien test_0.svg, test_1.svg ... aus der Zeichnung in test.fcd:

        java -jar fidocadj.jar -n -m -c r2 svg test.svg test.fcd
```

⁴Mit Ausnahme der Macintosh-Version, bei der es sich um eine eigenständige Anwendung handelt.

```

Fortsetzung der Liste der FidoCadJ-Optionen

-s      Gibt die Größe der angegebenen Datei in logischen Einheiten aus
-h      Drucken Sie diese Hilfe aus und beenden Sie den Vorgang.
-t      Gibt die Zeit aus, die FidoCadJ für den angegebenen Vorgang verwendet hat.
-p      Aktivieren Sie einige plattformabhängige Optimierungen nicht. Sie können diese Option ausprobieren, wenn FidoCadJ hängen bleibt oder sehr langsam ist.
-l      Erzwingen Sie, dass FidoCadJ ein bestimmtes Gebietsschema verwendet (der Code kann direkt folgen oder durch ein optionales Leerzeichen getrennt werden).
-k      Zeigt das aktuelle Gebietsschema an.
-f      Erzwingen Sie, dass FidoCadJ einige Gesundheitstests für die Eingabedaten überspringt.

[Datei] Die optionale FidoCadJ-Datei (außer wenn Sie die Optionen -d oder -s verwenden), die beim Start geladen werden soll.

Beispiel: Laden und konvertieren Sie eine FidoCadJ-Zeichnung in eine PNG-Datei mit 800 x 600 Pixeln, ohne die GUI zu verwenden.
          java -jar fidocadj.jar -n -c 800 600 png out1.png test1.fcd

Beispiel: Laden und Konvertieren einer FidoCadJ-Zeichnung in eine PNG-Datei, ohne die grafische Benutzeroberfläche zu verwenden (der sogenannte Headless-Modus). Jede logische FidoCadJ-Einheit wird im Bild in 2 Pixel umgewandelt.
          java -jar fidocadj.jar -n -c r2 png out2.png test2.fcd

Beispiel: Laden Sie FidoCadJ und erzwingen Sie das Gebietsschema auf vereinfachtes Chinesisch (zh).
          java -jar fidocadj.jar -l zh

[davidebucci@davide-bucci-portable]$
```

Die einfachste Option ist **-n**, die keine Auswirkungen auf die Software... hat, d. h. die GUI nicht aktiviert und einfach beendet wird. In diesem Fall ist die Java-Umgebungsvariable `java.awt.headless` auf "true" gesetzt. Offensichtlich ist diese Option allein nicht so nützlich, aber sie ist wertvoll wenn sie in Kombination mit anderen Funktionen verwendet wird, die wir gleich beschreiben werden. Option **-d** ermöglicht die Angabe des Ordners, in dem FidoCadJ nach Bibliotheken sucht, die beim Start geladen werden sollen. Option **-c** ermöglicht es FidoCadJ, eine FidoCAD-Datei (die bereitgestellt werden muss) in ein Bild im Vektor- oder Rasterformat zu konvertieren. Dies kann sehr nützlich sein, da FidoCadJ auf nicht interaktive Weise als Konverter verwendet werden kann (zusammen mit der Option **-n**).

Wir können uns also das erste Beispiel in der Hilfe ansehen:

```
java -jar fidocadj.jar -n -c 800 600 png out1.png test1.fcd
```

FidoCadJ läuft ohne Aktivierung der GUI und exportiert im PNG-Format die Datei `test1.fcd`. Die Ausgabedatei erhält den Namen `out1.png` und hat eine Pixelgröße von 800x600. Es gibt eine alternative Version von **-c** Option, mit der angegeben werden kann, wie viele Pixel zur Konvertierung einer logischen Einheit verwendet werden sollen (wir nennen diesen Faktor r_p). FidoCadJ funktioniert nicht mit halben logischen Einheiten (es sind immer ganze Zahlen). Wenn Sie beispielsweise r_p wählen, was zwei Pixeln pro logischer Einheit ent-

spricht, ist sichergestellt dass Schaltpläne immer verständlich sind (auch wenn sie wahrscheinlich etwas klein sind). Der Faktor r_p kann nicht ganzzahlig sein, wie im folgenden Beispiel:

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

Um die Gesamtgröße (in logischen Einheiten) eines Schaltplans zu ermitteln, können Sie `-s` verwenden. Denken Sie auf jeden Fall daran, dass FidoCadJ beim Export immer einen Rand von $t_{margin} = 3$ logischen Einheiten für jede Seite der Zeichnung hinzufügt. Wenn also t_w die Breite der Zeichnung in logischen Einheiten ist, die durch `-s` gegeben ist, können Sie davon ausgehen dass p_w die Breite Ihrer Zeichnung in Pixeln wie folgt berechnet :

$$p_w = r_p(t_w + 2t_{margin}) \quad (2.1)$$

Eine weitere interessante Option, obwohl eine Funktion von Java mehr als FidoCadJ, ist die Möglichkeit, das Erscheinungsbild der Anwendung zu ändern (genannt Look & Feel). Sie können das gewünschte Erscheinungsbild auswählen, ohne eine einzige Codezeile zu ändern. Folgendes werden Linux-Benutzer zu schätzen wissen: das GTK+-Aussehen:

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

Es gibt auch das klassische Erscheinungsbild von Motif, wie in Abbildung gezeigt [2.23](#)⁵:

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

Offensichtlich sind die oben genannten Befehle dazu gedacht von einem Terminal gesendet zu werden. Dabei muss sichergestellt werden, dass das aktuelle Verzeichnis die Datei `fidocad.jar` enthält und alles steht auf derselben Zeile.

⁵Dieser Stil könnte Leute, die mit hochentwickelten grafischen Oberflächen wie Aqua unter macOS vertraut sind, irgendwie schockieren. Allerdings habe ich vor vielen Jahren ein Synchrotron-Betriebssystem mit einer grafischen Oberfläche auf Basis von Motif gesehen und Respekt einflößte.

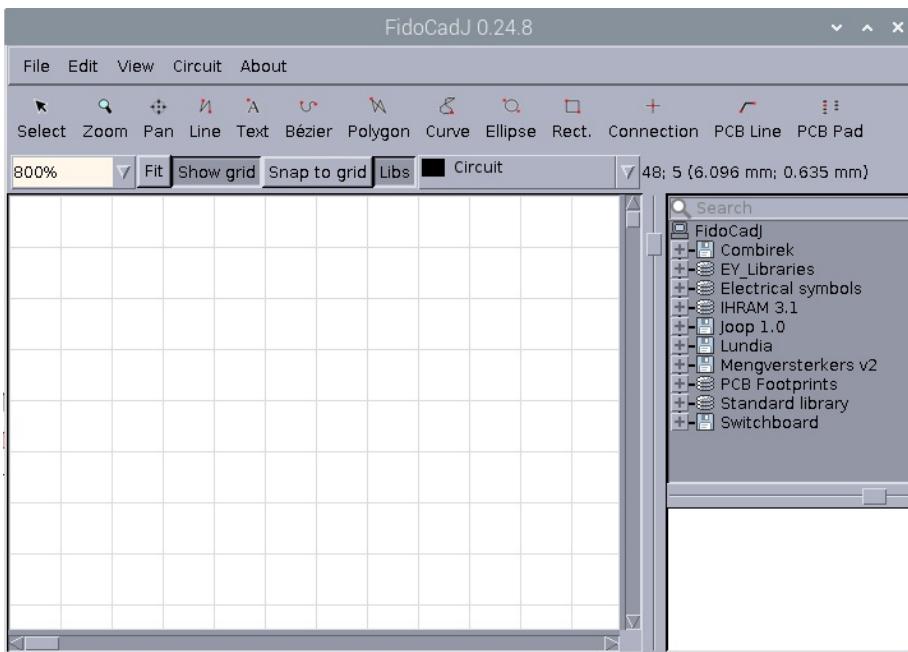


Abbildung 2.23: Das Aussehen von FidoCadJ unter Verwendung des Motif-Look & Feel.

2.10 Bibliotheksverwaltung

2.10.1 Verwendung von Bibliotheksdateien

Eine Bibliothek ist eine Sammlung von Symbolen, die der Benutzer in seine Zeichnungen integrieren kann. Das FidoCadJ-Paket enthält eine Sammlung von Bibliotheken sowie die Möglichkeit, neue Symbole und Bibliotheken zu definieren. Mit FidoCadJ können Sie sogar ein Ordner angeben, in dem alle Benutzerbibliothek-Dateien abgelegt werden (mit der Dateierweiterung `.fcl`). Dies kann über die Menüleiste und das Menü “Datei/Optionen” erfolgen.

In einigen ganz besonderen Fällen (oder zum Testen) können die Bibliotheken im FidoCadJ-Paket durch externe ersetzt werden. Wenn eine Datei mit dem Namen `FCDstdlib.fcl` vorhanden ist, ersetzt der Inhalt die Standardbibliothek die in der Anwendung sofort verfügbar ist. Analog gilt, wenn eine Datei mit dem Namen `PCB.fcl` vorhanden ist, ersetzt sein Inhalt die PCB-Bibliothek⁶.

In Version 0.23 konnte ich dank Roby IZ1CYN die IHRAM 3.1-Bibliothek direkt in die FidoCadJ-Distribution einbinden. Ich habe dies getan, weil diese von allen Bibliotheken die ich gesehen habe, eine der vollständigsten und rationalsten aufgebauten war. Genauso wie es bei den anderen in FidoCadJ enthaltenen Bibliotheken der Fall ist, wenn eine Datei mit dem Namen `IHRAM.FCL` vorhanden ist. Im aktuellen Bibliothekssuchpfad wird sie anstelle der im Programm eingebetteten Version geladen. Sie verfügen außerdem über eine Bi-

⁶Beachten Sie die Verwendung von Großbuchstaben, wenn Ihr Betriebssystem im Dateimanager zwischen Groß- und Kleinschreibung unterscheidet.

bliothek elektrischer Symbole, deren Datei `elettrotecnica.fc1` heißt.

Andere Dateien mit `.fc1` Erweiterung im Bibliothekssuchpfad gelten als Bibliotheken und FidoCadJ versucht sie beim Start zu laden. Dies geschieht wenn die Anwendung gestartet wird, wenn der Benutzer den Bibliothekssuchpfad ändert oder wenn in der Menüleiste und im Menü “Stromkreis” “Bibliothek aktualisieren” wird ausgewählt.

FidoCadJ ermöglicht das Aufteilen von nicht standardmäßigen Symbolen (wie es das ursprüngliche FidoCAD tut). Dies kann sehr nützlich sein, wenn Sie eine Zeichnung in einer Newsgruppe veröffentlichen, da dadurch alle Symbole, die nicht zu den Standardbibliotheken von FidoCAD gehört, in ihre grafischen Grundelemente konvertiert werden. Wer Ihre Nachricht liest, muss also nicht über dieselben Bibliotheken verfügen, die Sie in Ihrem System installiert haben. Für das Kopieren/Einfügen steht in der Menüleiste im Menü “Editieren” ein Befehl und eine Verknüpfung namens “Kopieren, nicht standardisierte Makros trennen” zur Verfügung (oder `Ctrl+I`⁷), um sicherzustellen dass die kopierte Zeichnung alle nicht standardmäßigen Makros aufgeteilt werden. Sie können eine Datei auch mit aufgeteilten Nicht-Standard-Makros speichern, indem Sie in der Menüleiste im Menü “Datei” auf “Speichern, nicht standardisierte Makros trennen” klicken. Es erscheint ein Dialog in dem Sie einen neuen Dateinamen wählen können, da Sie die Datei mit dem Sie gerade arbeiten normalerweise nicht überschreiben möchten.

2.10.2 Neue Symbole definieren

Mit FidoCadJ können Sie neue Bibliotheken und neue Symbole auf eine Art und Weise erstellen, die einigermaßen intuitiv sein sollte. Hier sind die Schritte, die Sie befolgen müssen:

- FidoCadJ benötigt einen Ort zum Speichern der erzeugten Dateien. Stellen Sie sicher, dass ein Ordner mit den Benutzbibliotheken definiert ist. Wenn nicht, geben Sie es in den FidoCadJ-Benutzereinstellungen an.
- Zeichnen Sie Ihr neues Symbol.
- Wählen Sie aus, was Sie gerade gezeichnet haben, und klicken Sie mit der rechten Maustaste darauf. Es erscheint ein Popup-Menü, wie in Abbildung 2.24 gezeigt.
- Wählen Sie “Symbol-o-matic”. Es erscheint ein Dialog zum Definieren der Details des neu erstellten Symbols (siehe Abbildung 2.25).
- Im ersten Feld “Dateiname der Bibliothek:” finden Sie den Dateinamen der Bibliothek, in die das neue Symbol eingefügt werden soll. Wenn Sie dort einen neuen Namen eingeben, erstellt FidoCadJ eine neue Datei.
- Das zweite Feld “Bibliothekname:” ist der vollständige Name der Bibliothek, der im Bibliotheksbaum angezeigt wird. Sie könnten dort schreiben was Sie wollen, aber es ist besser einen kurzen und aussagekräftigen Namen zu wählen.

⁷ Auf macOS-Systemen lautet die Tastenkombination `Command+I`

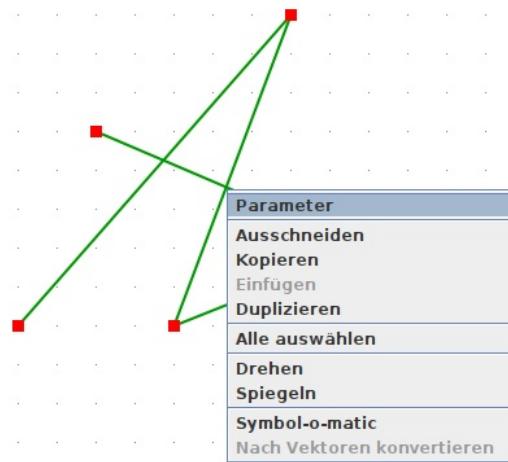


Abbildung 2.24: Das per Rechtsklick erscheinende Popup-Menü ermöglicht die Umwandlung von Zeichenelementen in ein Symbol

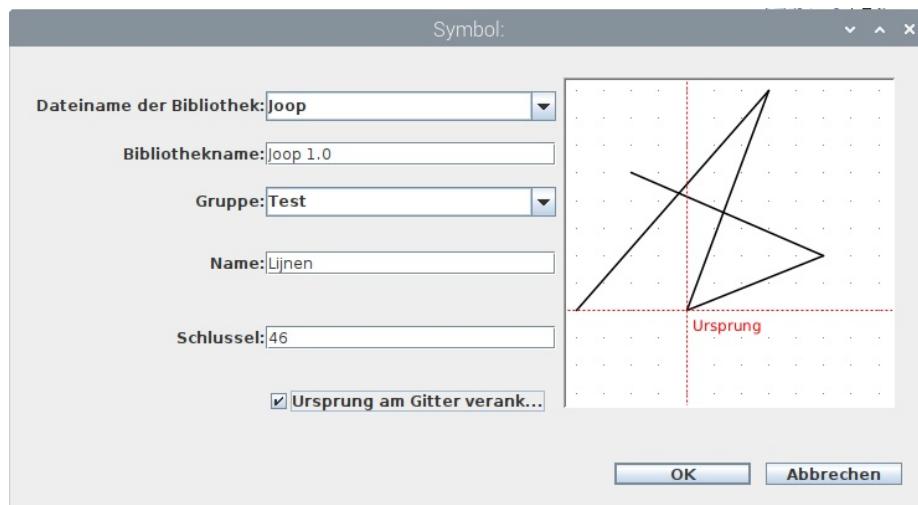


Abbildung 2.25: Der neue Symboldefinitionsdialog. Hier können Sie alle wichtigen Eigenschaften des Symbols einstellen. Beachten Sie den durch die beiden roten Achsen definierten Ursprung.

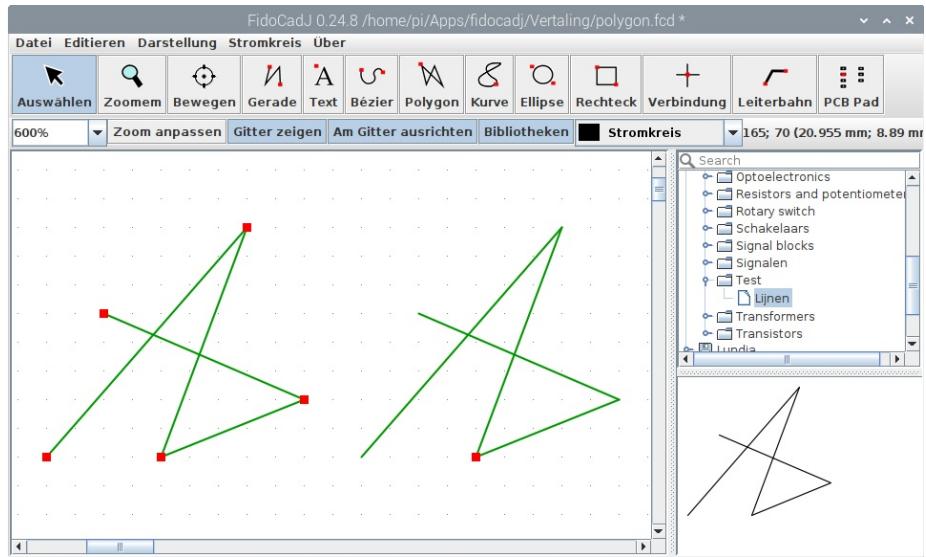


Abbildung 2.26: Das neu erstellte Symbol, angezeigt in der Symboliste und in der Zeichnung. Auf der linken Seite befindet sich noch die Zeichnung, die für die Symboldefinition verwendet wurde. Beachten Sie, dass es in der Zeichnung nur einen Kontrollpunkt für das neue Bibliothekssymbol gibt.

- Im dritten Feld “Gruppe:” können Sie auswählen in welcher Gruppe Sie das neue Symbol innerhalb der Bibliothek einfügen möchten. Auch hier wird durch die Eingabe eines neuen Namens eine neue Gruppe erstellt.
- Das vierte Feld “Name:” ist der Name des Symbols, wie es im Bibliotheksbaum erscheint. Wählen Sie erneut einen kurzen, aber aussagekräftigen Namen.
- Das fünfte Feld “Schlüssel:” ist ein sehr kurzes Tag, das das Symbol im Code identifiziert. Es muss innerhalb jeder Bibliothek eindeutig sein und darf keine Leerzeichen oder Zeichen wie Klammern, Punkte usw. enthalten. FidoCadJ schlägt kurzen numerischen Code vor, Sie können aber auch Mnemoniken verwenden.
- Durch Klicken in das Feld auf der rechten Seite müssen Sie den Ursprung des Symbols auswählen, den Punkt der als Referenz für die Platzierung des Symbols in der Zeichnung verwendet werden soll. Sie können die gewünschte Position im Raster ausrichten, indem Sie auf “Ursprung am Gitter verankern” klicken.
- Sobald Sie auf “OK” klicken, sollten Sie Ihr neu erstelltes Symbol in der Benutzerbibliothek im Bibliotheksbaum im Hauptfenster von FidoCadJ finden.

Ein Beispiel für ein in einer Zeichnung verwendetes Benutzersymbol ist in Abbildung 2.26 dargestellt. Beachten Sie den Unterschied zwischen dem ausgewählten Teil der Zeichnung auf der linken Seite (der tatsächlich zum Definieren eines neuen Symbols verwendet wurde) und dem neuen Symbol auf der



Abbildung 2.27: Das Popup-Menü zum Ändern von Symboleigenschaften in einer Benutzerbibliothek.

rechten Seite. Tatsächlich steht für die Symbolplatzierung nur ein Kontrollpunkt zur Verfügung, und dieser Kontrollpunkt entspricht der Wahl des Ursprungs bei der Symboldefinition. Daher ist es sinnvoll dafür einen Punkt auszuwählen, der in Ihrem Symbol eine gewisse Relevanz hat. Sie können ein Symbol wieder in seine Grundelemente aufteilen, indem Sie es auswählen und auf “Nach Vektoren konvertieren” klicken.

2.10.3 Vorhandene Symbole ändern

Für Symbole im Bibliotheksbaum sind einige Menüaktionen verfügbar. Sobald Sie ein Symbol in einer Benutzerbibliothek auswählen und mit der rechten Maustaste darauf klicken, erscheint ein Popup-Menü, wie in Abbildung 2.27 gezeigt. Hier haben Sie mehrere Möglichkeiten:

- “Kopieren” kopiert eine gesamte Bibliothek oder ein Bibliothekssymbol.
- “Einfügen” fügt die gesamte Bibliothek oder ein Bibliothekssymbol an der gewünschten Stelle ein.
- “Löschen” wirft Bibliothek oder Symbol weg. Seien Sie dabei vorsichtig: Wenn eine Zeichnung etwas aus der Bibliothek enthält oder enthält ein Symbol wird es ungültig und verschwindet.
- “Umbenennen” ändert den angezeigten Bibliotheks- oder Symbolnamen.
- “Schlüssel umbenennen” ist nützlich um den mit dem Symbol verknüpften Hotkey zu ändern. Auch dies muss dann erfolgen wenn das Symbol in der Zeichnung nicht verwendet wurde.

Durch die Verwendung von Drag-and-Drop können Sie die Bibliothek oder die Kategorie ändern, in die ein Symbol kategorisiert wird. Beachten Sie, dass ein Tag zur vollständigen Identifizierung eines Symbols aus dem Namen der Bibliothek gefolgt vom Schlüssel des Symbols besteht. Wenn Sie also ein Symbol von einer Bibliothek in eine andere verschieben, werden Zeichnungen die es enthalten ungültig.

Alle Bibliotheksänderungen können rückgängig gemacht werden, genau wie Zeichnungsänderungen. Beachten Sie, dass leere Bibliotheken oder Gruppen nicht angezeigt werden.

Kapitel 3

Zeichnungsformat, Makros & Bibliothek

Dieses Kapitel enthält eine detaillierte Beschreibung des Formats, das von FidoCAD und damit auch von FidoCadJ zum Speichern einer Zeichnung verwendet wird. Es handelt sich um ein einfaches Textformat, das den Vorteil hat, sehr kompakt und effizient zu sein. Da das Format noch nie in einem Dokument ausführlich beschrieben wurde, versuche ich, alles zusammenzufassen, was ich darüber gelernt habe. Da ich einige Erweiterungen hinzugefügt habe, die nur in FidoCadJ verfügbar sind, beschreibe ich diese auch hier. Beachten Sie, dass FidoCadJ seit Version 0.23.4 auf allen Plattformen nur noch die UTF-8-Kodierung verwendet.

3.1 Header-Beschreibung

Alle Zeichnungsdateien im FidoCAD-Format müssen mit dem Tag *[FIDOCAD]*. beginnen. Daher kann ein Programm das Vorhandensein von FidoCAD-Befehlen erkennen indem Sie dieses Tag lesen. In dieser Hinsicht ist FidoCadJ toleranter als das ursprüngliche FidoCAD und erkennt und interpretiert eine Datei korrekt die nicht den Standard-Header enthält. Daher können auch Befehle mit Text korrekt interpretiert werden, solange die Anzahl der Fehlerzeilen einen im Programm intern festgelegten Wert (ca. 100) nicht überschreitet. Dadurch wird sichergestellt dass FidoCadJ keine Zeit damit verschwendet einige Minuten lang zu arbeiten, beispielsweise mit dem Versuch eine sehr große Binärdatei zu öffnen.

3.2 Koordinatensystem

FidoCadJ arbeitet mit einem sehr einfachen Koordinatensystem. In der Praxis steht ihm ein sehr großes Gebiet zur Verfügung, das nur durch ganze und positive Koordinaten identifiziert wird. Die Länge jeder Einheit in x und in y ist auf $127\mu\text{m}$ festgelegt, ein Wert, der es ermöglicht, selbst für das kleinste SMD-Gehäuse eine gute Auflösung zu erzielen, ohne für den täglichen Gebrauch zu fein zu sein. Typografisch gesehen beträgt die FidoCadJ-Auflösung 200 Punkte pro inch.

Das ursprüngliche FidoCAD hatte zwei verschiedene Verarbeitungsmodi: PCB und elektrisches Diagramm. In FidoCadJ ist dieser Unterschied unscharf und erscheint erst beim Drucken der Zeichnung. Es empfiehlt sich daher, das Programm so einzustellen, dass die Größe eines Schaltplans angepasst wird, um die Seitengröße besser auszunutzen andernfalls hat das Druckergebnis die Größe einer Briefmarke.

3.3 Elemente zeichnen

FidoCadJ kann 12 Zeichnungselemente wie folgt verwalten

- Gerade
- Gefülltes oder leeres Rechteck
- Einfacher Text (veraltet)
- Erweiterter Text
- Gefülltes oder leeres Polygon
- Gefüllte oder leere Ellipse
- Bézier-Kurve
- Natürliche kubische Kurve
- Elektrischer Verbindung
- PCB-Pad
- Leiterbahn
- Makroaufruf

Wir werden sie alle beschreiben. Im Allgemeinen wird jedes Element durch einen Befehl und eine Reihe von Parametern (normalerweise ganze Zahlen oder Textzeichenfolgen) identifiziert, die in derselben Zeile platziert und durch ein Leerzeichen getrennt sind.

Gerade

Das Geradeprimitiv wird durch den Befehl `LI` identifiziert und seine Definition erfordert nur die Anfangs- und Endkoordinaten und die Ebene:

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

Mathematiker würden wahrscheinlich den Begriff "Segment" passender finden.

Die Punkte (x_1, y_1) und (x_2, y_2) stellen jeweils die Anfangs- und Endkoordinaten dar, während l die Ebene ist, die durch eine ganze Zahl gekennzeichnet ist zwischen 0 und 15.

FidoCadJ-Erweiterung: Ab FidoCadJ-Version 0.23 kann auf `LI` in der nächsten Zeile eine Erweiterung folgen:

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

Tabelle 3.1: Bedeutung des Parameters a für das Vorhandensein einer Pfeilspitze an den Seiten einer Linie oder eines Bézier-Grundelements.

a	Pfeilspitze
0	keiner
1	am Anfang
2	am Ende
3	beide Seiten

Tabelle 3.2: Bedeutung des b -Parameters für den Pfeilspitzenstil.

b	Pfeilspitzenstil
0	gefüllte Standardpfeilspitze
1	Gefüllte Standardpfeilspitze mit Maßlinie
2	leere Standardpfeilspitze
3	leere Standardpfeilspitze mit Maßlinie

wobei a eine Ganzzahl ist, die das Vorhandensein oder Nichtvorhandensein der Pfeilspitzen an den Enden des Segments darstellt (siehe Tabelle 3.1), b den Stil der Pfeilspitze darstellt (wie Tabelle 3.2). Die Parameter c und d geben die Gesamtlänge bzw. die halbe Breite der Pfeilspitze an, während e eine Ganzzahl ist die den Strichstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Gefülltes oder leeres Rechteck

Ein gefülltes oder leeres Rechteck wird durch die Befehle **RP** bzw. **RV** identifiziert, gefolgt von den Koordinaten der beiden Scheitelpunkte auf einem der zwei Diagonalen und die Ebene.

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

die Punkte (x_1, y_1) und (x_2, y_2) stellen die beiden Scheitelpunkte dar und l ist die Schicht, gekennzeichnet durch eine ganze Zahl zwischen 0 und 15 .

FidoCadJ-Erweiterung: Ab FidoCadJ-Version 0.23 kann auf **RP** und **RV** in der nächsten Zeile eine Erweiterung folgen:

```
FCJ e nv
```

Dabei ist e eine Ganzzahl, die den Streifenstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Einfacher Text (veraltet)

Der einfache Text war das erste Grundelement, das mit den frühen Versionen von FidoCAD ausgeliefert wurde. FidoCadJ erkennt es und schreibt einfach Text der Größe 12, unabhängig vom aktuellen Zoom.

Da dieses Grundelement von Lorenzo Lutti, dem Schöpfer von FidoCAD, als veraltet angesehen wurde, macht FidoCadJ genau das Gleiche, und obwohl

Tabelle 3.3: Funktion der Bits im Textstilbegriff.

Bit	Wert	Verhalten
0	1	Text fett gedruckt
2	4	gespiegelter Text

dies korrekt interpretiert wird, ist es nicht verfügbar auf Symbolleiste. FidoCadJ speichert dieses Element genau so, als wäre es erweiterter Text und verwendet beim Speichern der Datei den Befehl **TY**.

Der Befehl lautet **TE** und das Format ist wie folgt:

```
TE x1 y1 te schrijven tekst
```

Der Punkt (x_1, y_1) ist der Ort, an dem die Zeichenfolge “zu schreibender Text” positioniert werden soll. Beachten Sie, dass die Ebeneninformationen fehlen. FidoCadJ behandelt dieses Objekt so, wie es auf Ebene Null (Stromkreis) platziert wurde.

Erweiterter Text

Der primitive erweiterte Text bietet viel mehr Flexibilität im Vergleich zum oben eingeführten einfachen Text.

Es wird durch den Befehl **TY** identifiziert, gefolgt von einer Reihe von Parametern zur Bestimmung der Textausrichtung (gedreht oder gespiegelt) und für Abmessungen x und y und die verwendete Schriftart. Aufgrund der Menge an Informationen die bereitgestellt werden müssen, ist die resultierende Befehlszeichenfolge recht komplex:

```
TY x1 y1 sy sx a s l f zu schreibender Text
```

Der Punkt (x_1, y_1) ist der Ort, an dem die Zeichenfolge “zu schreibender Text” positioniert werden soll. Der Wert von s_y und s_x gibt die horizontalen und vertikalen Abmessungen des Textes in logischen Einheiten an. FidoCadJ respektiert die vertikale Dimension, geht von der horizontalen Dimension aus und ändert dieses Seitenverhältnis nur, wenn dies unbedingt erforderlich ist. Die Textdrehung wird durch den Begriff a beschrieben, ausgedrückt in Sexagesimalgraden, während der Wert von s den Textstil gemäß Tabelle 3.3 bestimmt. Die Ebene wird durch den allgemeinen Begriff l angegeben, und f gibt die zu verwendende Schriftart an, oder es kann ein Sternchen sein, um die Verwendung der Standardschriftart Courier New. Wenn der Schriftartname Leerzeichen enthält, müssen diese durch “++” ersetzt werden.

Textbefehle unterstützen Indizes: Um sie zu verwenden, fügen Sie Text zwischen dem Unterstrich $_$ und dem Caretzeichen $^$ ein. Beispielsweise schreibt $_12^$ 12 als Index. Sie unterstützen auch hochgestellte Zeichen: Fügen Sie Text zwischen dem Caretzeichen $^$ und dem Unterstrich $_$ ein; Beispielsweise schreibt 56_ 56 hochgestellt. In beiden Fällen sind die letzten $^$ oder $_$ optional. Um das Zeichen $_$ oder $^$ zu drucken, verwenden Sie eine Rücktaste, d. h. $\backslash_$, um $_$ zu erhalten, und $\backslash^$, um $^$ zu erhalten. Verwenden Sie zwei Rücktasten, um einen einzelnen \ zu erhalten. Beispiel für Text mit hochgestellten Zeichen: $3^2_+4^2_=_5^2$ oder C_12

Die maximale Textlänge beträgt etwa 80 Wörter. Die Zählung erfolgt in Wörtern und nicht in Zeichen, da in der internen Struktur des Programms die Wörter (und Befehlsbegiffe) bei der Interpretation einer Zeile getrennt werden.

Gefülltes oder leeres Polygon

Ein gefülltes oder leeres Polygon wird durch die Befehle **PP** bzw. **PV** angezeigt, gefolgt von den Koordinaten der Scheitelpunkte, die das Polygon definieren und die Schicht. Die Befehle sind

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

wobei die Punkte $(x_1, y_1), (x_2, y_2)\dots$ die Scheitelpunkte sind, die das Polygon definieren und l ist die Schicht, gekennzeichnet durch eine ganze Zahl zwischen 0 und 15. Daher kann die Länge der Befehlszeile je nach Anzahl der verwendeten Scheitelpunkte variieren. Bei Polygonen ist die maximale Anzahl der verfügbaren Scheitelpunkte intern auf knapp 5000 festgelegt.

FidoCadJ-Erweiterung: Ab FidoCadJ-Version 0.23 kann auf **PP** und **PV** eine Erweiterung in der folgenden Zeile folgen:

```
FCJ e nv
```

wobei e eine Ganzzahl ist, die den Streifenstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Gefüllte oder leere Ellipse

Eine gefüllte oder leere Ellipse wird durch die Befehle **EP** bzw. **EV** identifiziert, gefolgt von den Koordinaten der zwei Scheitelpunkte auf der Diagonale und die Schichtnummer.

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

Der Punkt (x_1, y_1) stellt den ersten Scheitelpunkt auf der Diagonale dar, (x_2, y_2) ist der zweite Scheitelpunkt und l ist der identifizierte Schicht mit einer Ganzzahl zwischen 0 und 15.

FidoCadJ-Erweiterung: Ab FidoCadJ-Version 0.23 kann auf **EP** und **EV** eine Erweiterung in der folgenden Zeile folgen:

```
FCJ e nv
```

wobei e eine Ganzzahl ist, die den Streifenstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Bézier-Kurve

Ein Bézier-Kurvensegment wird in seiner kubischen Variante durch vier Scheitelpunkte identifiziert, die daher im entsprechenden Befehl **BE** erforderlich sind:

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

wobei $P_1 \equiv (x_1, y_1)$, $P_2 \equiv (x_2, y_2)$, $P_3 \equiv (x_3, y_3)$ und $P_4 \equiv (x_4, y_4)$ die vier Kontrollpunkte des Bézier-Kurvensegments, während l die Schicht ist, identifiziert durch eine ganze Zahl zwischen 0 und 15. Angesichts der vier oben definierten Punkte wird das Kurvensegment mit dem Ausdruck

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

berechnet, wobei $t \in [0, 1]$ ein Parameter ist.

FidoCadJ-Erweiterung: Ab FidoCadJ-Version 0.23 kann auf **BE** eine Erweiterung in der folgenden Zeile folgen:

FCJ a b c d e nv

wobei a eine ganze Zahl ist, die das Vorhandensein oder Nichtvorhandensein von Pfeilspitzen an den Enden der Kurve darstellt (siehe Tabelle 3.1), b den Stil der Pfeilspitze darstellt (wie Tabelle 3.2). Die Parameter c und d geben die Gesamtlänge bzw. die halbe Breite der Pfeilspitze an, während e eine Ganzzahl ist, die den Strichstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Natürliche kubische Kurve

Eine natürliche kubische Kurve wird durch eine bestimmte Anzahl von Scheitelpunkten definiert. Die Kurve kreuzt jeden Scheitelpunkt und wird so berechnet, dass sie sehr glatt ist. Im FidoCadJ-Format wird eine Kurve durch die Befehle **CV** und **CP** identifiziert:

CV aa x1 y1 x2 y2 ... 1
CP aa x1 y1 x2 y2 ... 1

Der Parameter aa ist gleich 1, wenn die Kurve geschlossen ist, andernfalls ist er 0. Die Scheitelpunkte $(x_1, y_1), (x_2, y_2)\dots$ definieren die Kurve und l ist die Ebene, eine Ganzzahl aus 0 bis 15. Daher kann die Länge der Befehlszeile variieren, je nachdem wie viele Scheitelpunkte enthalten sind. Was Polygone betrifft, ist die maximale Anzahl der verfügbaren Scheitelpunkte intern auf knapp 5000 festgelegt. Dieses Grundelement wurde in Version 0.24 eingeführt und ist im ursprünglichen FidoCAD nicht vorhanden.

FidoCadJ-Erweiterung: Auf **CV** und **CP** kann eine Erweiterung in der folgenden Zeile folgen:

FCJ a b c d e nv

wobei a eine ganze Zahl ist, die das Vorhandensein oder Nichtvorhandensein der Pfeilspitzen an den Enden der Kurve darstellt (siehe Tabelle 3.1), b den Stil der Pfeilspitze darstellt (wie Tabelle 3.2). Die Parameter c und d geben die Gesamtlänge bzw. die halbe Breite der Pfeilspitze an, während e eine Ganzzahl ist, die den Strichstil angibt. Wenn nv gleich 1 ist sollten der Befehl **FCJ** gefolgt werden von zwei **TY** Befehle, die den mit diesem Element verknüpften Namen und Wert angeben.

Elektrische Verbindung

Der primitive Elektrische Verbindung ist einfach ein gefüllter Kreis mit konstanten Abmessungen und wird zur Darstellung einer Verbindung in einem elektrischen Diagramm verwendet. Es wird durch den Befehl **SA** identifiziert und benötigt lediglich die Koordinaten und die Schicht:

```
SA x1 y1 1
```

Bei FidoCadJ wird der Durchmesser des Kreises programmintern auf zwei logische Einheiten festgelegt.

Wenn auf **SA** ein **FCJ** folgt, müssen auf den **FCJ**-Befehl zwei **TY**-Befehle folgen, die den mit diesem Element verknüpften Namen und Wert angeben.

PCB-Pad

Ein PCB-Pad wird durch den Befehl **PA** identifiziert und ist durch seinen Stil (rund, rechteckig, rechteckig mit abgeschrägten Ecken) und den Durchmesser des Innenlochs gekennzeichnet:

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

wobei der Punkt (x_1, y_1) die Position des Pads darstellt, d_x die Breite des Pads (entlang der x -Achse) ist, d_y die Höhe (entlang der y -Achse). Der Wert s_i ist der Durchmesser des Lochs im Pad, während s_t der Stil des Pads ist:

0 ovales Pad

1 rechteckiges Pad

2 rechteckiges Pad mit abgerundeten Ecken

Der Wert von l muss eine ganze Zahl sein, um die Ebene anzugeben, auf der das Pad platziert werden soll. Wenn auf **PA** ein **FCJ** folgt, müssen auf den **FCJ**-Befehl zwei **TY**-Befehle folgen, die den mit diesem Element verknüpften Namen und Wert angeben.

Leiterbahn

Die Leiterbahn ist im Wesentlichen ein Segment, dessen Breite angegeben werden kann. Die Ecken an den Enden des Segments sind immer abgerundet, um die Verbindung mit anderen Leiterplattenbahnen oder -pads zu erleichtern. Der zu verwendende Befehl lautet **PL** und hat das folgende Format:

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

Die Spur wird zwischen den Punkten (x_1, y_1) und (x_2, y_2) gezeichnet, mit der Gesamtbreite d_i , und die verwendete Ebene ist l . Die Breite d_i kann eine Bruchzahl sein. Wenn **PL** von **FCJ** gefolgt wird, müssen auf den **FCJ**-Befehl zwei **TY**-Befehle folgen, die den mit diesem Element verknüpften Namen und Wert angeben.

Makroaufruf

Ein Makro ist eine Zeichnung oder ein Symbol in einer Bibliothek. Im Allgemeinen werden häufig verwendete elektrische Symbole auf diese Weise dargestellt. Der zum Aufruf eines Makros verwendete Befehl lautet **MC** und der Aufruf erfolgt wie folgt:

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

Das Symbol wird mit der Position (x_1, y_1) als Referenz gezeichnet und die Ausrichtung wird durch den Wert von o (multipliziert mit 90° im Uhrzeigersinn) bestimmt und wenn m gleich 1 ist, wird das Makro gespiegelt. Der letzte Parameter n ist der Name des Makros in der Bibliothek, angegeben als *library.code*.

Wenn auf **MC FCJ** folgt, müssen auf den **FCJ**-Befehl zwei **TY**-Befehle folgen, die den mit diesem Element verknüpften Namen und Wert angeben.

3.4 FidoCadJ-Erweiterungen

Seit Version 0.21 hat FidoCadJ damit begonnen, einige Verbesserungen gegenüber dem ursprünglichen FidoCAD-Format einzuführen. Die FidoCadJ Erweiterungen werden im Code durch den Befehl **FCJ** dargestellt. Dieser Befehl wird nicht alleine verwendet, sondern bedeutet, dass FidoCadJ zusätzliche Informationen zu den Angaben in der vorherigen Zeile angeben muss. Hier ist ein Beispiel:

```
[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
```

Die Anwesenheit des **FCJ** Der Befehl gibt an, dass der Name und der Wert des in der ersten Zeile angegebenen Makros durch die beiden **TY**-Befehle angegeben werden. Befehle, die folgen. Für die Textdarstellung werden lediglich die Koordinaten und die Schriftarten berücksichtigt.

FidoCadJ ermöglicht die Aktivierung eines “strikten FidoCAD Kompatibilitätsmodus”, in dem alle Erweiterungen deaktiviert sind. Auf diese Weise ist FidoCadJ perfekt kompatibel mit dem ursprünglichen FidoCAD, mit der Ausnahme, dass FidoCadJ weiterhin die UTF-8-Kodierung anstelle der alten CP-1252 wird von FidoCAD verwendet. FidoCAD für Windows kann die vom Befehl **FCJ** übertragenen zusätzlichen Informationen nicht verstehen. Wenn Sie mit dem Original-FidoCAD eine FidoCadJ-Zeichnung lesen die eine Erweiterung enthält, meldet das Programm einige Fehler und liefert ein Ergebnis, das sich von der Original-FidoCadJ-Zeichnung unterscheidet. Die Ergebnisse hängen davon ab, welche Erweiterung verwendet wurde: Eine gestrichelte Linie wird als durchgehend dargestellt und die Pfeilspitzen werden nicht gezeichnet. Andererseits wird der mit dem Namen und dem Wert eines Makros verknüpfte Text perfekt wiedergegeben.

Figure 3.1 zeigt, was mit FidoCAD erreicht werden kann, um die FidoCadJ-Datei zu lesen, die Figure 2.20 beschreibt. Nachdem man die FidoCAD-Fehler ignoriert hat, fehlen einige Details (die Strichelung und der Pfeil), aber die Gesamtzeichnung ist immer noch verständlich.

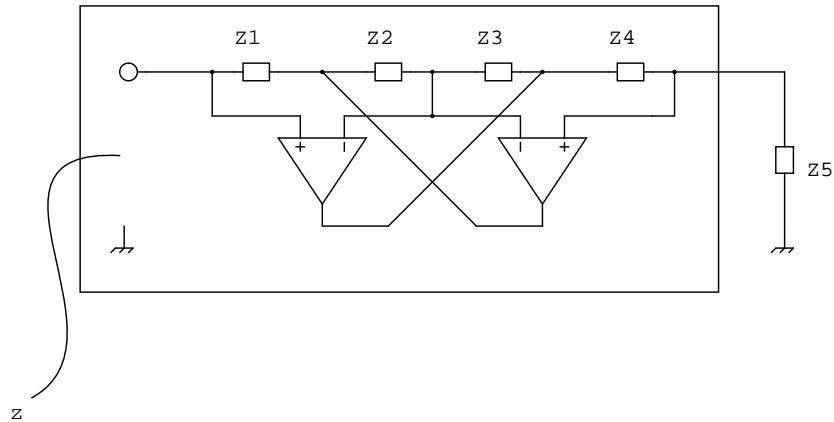


Abbildung 3.1: Figure 2.20, wie es in FidoCAD für Windows erscheinen würde.

Ein wichtiger Unterschied zum ursprünglichen FidoCAD besteht darin, dass FidoCadJ die Speicherung einer bestimmten Anzahl von Konfigurationshinweisen in den Ausgabedateien ermöglicht. Der dafür verwendete Befehl lautet **FJC** und sollte normalerweise ganz am Anfang der Datei stehen. In den nächsten Absätzen werden die behandelten Fälle beschrieben.

3.4.1 Ebenenaufbau

Die Layerstruktur wird mit dem Befehl **FJC L** angegeben. FidoCadJ speichert Daten nur, wenn die entsprechende Ebene gegenüber der Standardebene geändert wurde. Die Syntax lautet wie folgt:

```
FJC L n xxxx yy
```

Dabei ist *n* die Nummer der Ebene (im Bereich von 0 bis 15) und *xxxx* eine 32-Bit-Ganzzahl mit Einstellungen zur zu verwendenden RGB-Farbe. Die rote Komponente befindet sich in den Bits 16-23, die grüne in Bits 8-15 und die blaue in den Bits 0-7. Der Wert *yy* ist eine Gleitkommakonstante mit einfacher Genauigkeit, die der Ebene Transparenz zwischen 0,0 (vollständig transparent) und 1,0 (vollständig undurchsichtig) ändert.

Eine zweite wichtige Information ist der Name der Ebene, der (sofern der Benutzer ihn geändert hat) wie folgt angegeben wird:

```
FJC N n aaaa
```

Dabei ist *n* die Nummer der Ebene (Ganzzahl von 0 bis 15), während *aaaa* der Name der zu berücksichtigenden Ebene ist. Wenn dieser Befehl nicht vorhanden ist, wird der Name der Ebene standardmäßig von FidoCadJ verwendet, abhängig von der Sprache und der lokalen Konfiguration des Betriebssystems.

3.4.2 Elektrischer Anschlussaufbau

Die Größe des schwarzen Kreises, der zur Anzeige einer elektrischen Verbindung verwendet wird, kann vom Benutzer geändert werden. Wenn die FidoCadJ-Erweiterungen aktiv sind, wird der ausgewählte Wert mit dem Befehl **FJC C** wie folgt in der Datei gespeichert:

```
FJC C aaaa
```

Dabei ist *aaaa* ein Gleitkommawert mit doppelter Genauigkeit (positiv), der den Durchmesser der elektrischen Verbindung in logischen FidoCadJ-Einheiten angibt.

3.4.3 Linienbreite

Die Linienstärke für den “Stift”, der beim Zeichnen elektrischer Diagramme verwendet wird, kann geändert werden. FidoCadJ verwendet den Befehl **FJC A**, um die Linienbreite der Linien, Ovale, Bézier, komplexen Kurven(Splines), Rechtecke und Polygone anzugeben. Die Breite kann eine Konstante mit doppelter Genauigkeit sein.

```
FJC A aaaa
```

Wobei *aaaa* die Linienbreite ist (in logischen Einheiten). Die Standardbreite ist intern auf 0,5 logische Einheiten definiert. In älteren Versionen von FidoCadJ wurde ein **FJC B bbbb**-Befehl verwendet, um die Linienbreite von gekrümmten Linien (Ovale, Bézier) anzugeben. Seit Version 0.23.6 entfällt diese Unterscheidung und dieser Befehl wird nicht mehr übernommen.

3.5 Toleranz für Syntaxfehler

FidoCadJ ist darauf ausgelegt, Fehler und/oder bestimmte Syntaxfehler in den an das Programm übergebenen Befehlen zu tolerieren. Sofern Sie keine Kristallkugel als USB-Gerät angeschlossen haben, kann das Programm natürlich keine Fehler korrigieren, sondern überspringt (und löscht) einfach alle betroffenen Zeilen.

Eine Ausnahme von diesem Verhalten besteht darin, dass eine Reihe von Elementen ohne den Layer angegeben werden können, der als Teil des Layer 0 (Stromkreis) betrachtet wird. Dies dient der Abwärtskompatibilität mit FidoCAD.

3.6 Bibliotheksformat

Der Dateiaufbau einer Bibliotheksdatei ist recht einfach:

```
[FIDOLIB Librairie de base]
{Sybôles de base}
[000 Terminal]
LI 100 100 102 100
EV 102 98 106 102
[010 Terminal +]
LI 100 100 102 100
EV 102 98 106 102
LI 103 100 105 100
LI 104 99 104 101
[020 Terminal -]
LI 100 100 102 100
EV 102 98 106 102
LI 103 100 105 100
...
...
```

Die erste Zeile enthält in eckigen Klammern den Namen der Bibliothek (vorangestellt durch “FIDOLIB”). Die zweite Zeile muss in geschweiften Klammern die Bibliothekskategorie enthalten, unter der die später in der Datei angegebenen Makros gespeichert werden sollen.

Jedes Makro besteht aus einem Header (in eckigen Klammern) und einer Befehlsfolge. Der Header besteht aus einem *Elementnamen* (der innerhalb der Bibliothek eindeutig sein muss) und seiner Beschreibung. Der Elementnamen wird in einem FidoCadJ-Skript verwendet, während die Beschreibung dem Benutzer beim Durchsuchen aller in der Datei enthaltenen Makros hilft. Die Befehle sind nichts anderes als FidoCadJ-Zeichnungen, bei denen der Koordinatenpunkt (100,100) als Ursprung verwendet wird. Dies ist der Punkt, der beim Aufruf des Makros als Referenz verwendet werden soll. In einem FidoCadJ-Skript wird ein durch “library.macro” identifiziertes Makro mit dem Befehl MC verwendet.

Nichts verhindert den Aufruf eines Makros innerhalb eines anderen Makros. Allerdings muss recursion (also ein Makro, das sich selbst aufruft) vermieden werden.

Eine Bibliothek *darf* unter den Makrodefinitionen *keine* Informationen über die FidoCadJ-Konfiguration enthalten. Mit anderen Worten: Befehle **FJC** sollten *nie* in einer Bibliotheksdatei erscheinen.

3.7 Standardbibliotheken

FidoCadJ enthält zwei Bibliotheken, die traditionell mit FidoCAD geliefert werden. Dies sind die Standardbibliothek und die Bibliothek mit den PCB-Symbolen. Eine weitere Bibliothek, die für FidoCAD entwickelt wurde und so nützlich ist, dass sie Teil von FidoCadJ ist, ist die IHRAM-Bibliothek. IHRAM steht für **it.hobby.radioamatori.moderato** und es ist eine italienische Usenet-Diskussionsgruppe.

Es ist jedoch möglich, den Inhalt dieser internen Bibliotheken zu überschreiben, indem Sie (Menüleiste, dann Menü “Datei/Optionen/Bibliothekenpfad:”) einen Ordner angeben, der die zu ladenden Bibliotheken enthält. Wenn eine Datei mit

dem Namen `FCDstdlib.fcl` in diesem Ordner vorhanden ist, wird sein Inhalt anstelle der Standardbibliothek verwendet. Wenn eine Datei namens `PCB.fcl` vorhanden ist, wird dessen Inhalt anstelle der internen PCB-Bibliothek verwendet. Andere Bibliotheken mit anderen Dateinamen (aber immer noch mit der Erweiterung `fcl`) werden zusammen mit den Standardbibliotheken geladen. Wenn eine Datei mit dem Namen `IHRAM.FCL` im aktuellen Bibliothekssuchpfad wird sie anstelle der im Programm eingebetteten Version geladen. Sie verfügen außerdem über eine elektrische Symbolbibliothek, deren Datei `elettrotecnica.fcl` heißt.

Die Standardbibliotheken in FidoCadJ sind außerhalb des Namens an dem Symbol mit drei übereinander liegenden Scheiben (für “Festplatte”) zu erkennen. Eigene und andere Bibliotheken, die sich im definierten Ordner “Datei-/Optionen/Bibliothekenpfad:” befinden, sind am Diskettensymbol zu erkennen.



Abbildung 3.2: Beispiel dafür wie Bibliotheken im Ordner “Bibliotheken” angezeigt werden.

FidoCadJ betrachtet die oben genannten Bibliotheken als Teil des Standardsatzes. Mit anderen Worten: Die in Zeichnungen enthaltenen Symbole werden nicht geteilt, wenn sie zu diesen Bibliotheken gehören, es sei denn, die Option “Strikte Kompatibilität mit FidoCAD” ist aktiv. In diesem Fall gilt nur die ursprüngliche FidoCAD-Bibliothek als Standard, genau so wie sie früher in der Originalsoftware ausgeführt wurde.

Kapitel 4

Conclusie

In diesem Handbuch haben wir gesehen, wie man mit FidoCadJ einen elektrischen Schaltplan oder eine einfache Leiterplatte zeichnet. Zu diesem Zeitpunkt sollte der Leser über alle notwendigen Elemente verfügen, um FidoCadJ effektiv für seine Bedürfnisse zu nutzen.

FidoCadJ sollte nicht nur für das Elektronikdesign gelten. Es kann für jede Art von 2D-Zeichnung und in vielen Situationen verwendet werden, sofern spezielle Bibliotheken verfügbar sind.

Die Vorteile eines kostenlosen Programms bestehen darin, dass es der Benutzergemeinschaft vollständig offen steht. Aus diesem Grund ist Ihr Feedback sehr wichtig (auf jeden Fall, um zu verstehen, ob und in welche Richtung das Projekt eine Weiterentwicklung wert ist). Um mich zu kontaktieren, öffnen Sie ein “Issue” im GitHub FidoCadJ-Projekt¹.

¹<https://github.com/DarwinNE/FidoCadJ/issues>

Anhang A

Plattformspezifische Informationen

A.1 macOS

Einer der häufigsten Kritikpunkte von Macintosh-Benutzern (wie übrigens auch mir) an den frühen Versionen von FidoCadJ war die schlechte Integration des Programms unter macOS. Ab Version 0.21.1 unternimmt FidoCadJ besondere Anstrengungen, um sich besser an das Aussehen und die Philosophie der nativen Mac-Anwendungen anzupassen. Aus diesem Grund unterscheiden sich einige Details im Programm geringfügig, wenn FidoCadJ auf einer Apple-Plattform ausgeführt wird:

- Standardmäßig verwendet FidoCadJ den Quaqua¹-Aussehen, wenn die vollständige Anwendung (FidoCadJ.app anstelle von fidocadj.jar) ausgeführt wird. Da Quaqua die Leistung auf nicht ganz so aktuellen Rechnern verlangsamen kann, ist es möglich, es über die Einstellungen im Menü “Einstellungen” zu deaktivieren.
- Die Menüleiste wird an ihrer Stelle am oberen Bildschirmrand angezeigt.
- Die Menüpunkte “Einstellungen” und “Über FidoCadJ” befinden sich an ihrer Stelle im FidoCadJ-Menü.
- Das Programm teilt dem Betriebssystem mit, dass es .fcd Dateien öffnen kann; Diese sind einem bestimmten Symbol zugeordnet, das ausreichend suggestiv sein muss.

A.1.1 So laden Sie FidoCadJ unter macOS herunter und führen es aus

FidoCadJ kann mit einer macOS-Version höher als 10.6 (Snow Leopard) arbeiten. Unter der Haube benötigt FidoCadJ mindestens Version 9 von Java. Aus Marketinggründen installiert Apple Java nicht mit den neuesten Versionen von macOS. Übrigens erlaubt Apple nicht, seine App Store-Software zu vertreiben,

¹<http://www.randelshofer.ch/quaqua/>

die auf Java basiert oder unter der GPL vertrieben wird. Dies ist einer der wichtigsten Gründe, warum es ziemlich unwahrscheinlich ist, dass FidoCadJ auf iPad und iPhone portiert wird.

Auch wenn Sie das Java-Archiv `fidocadj.jar` verwenden Wie auf anderen Betriebssystemen können Sie auch auf macOS die speziell zugeschnittene Anwendung verwenden. Alles funktioniert wie bei einer nativen Anwendung: Sie können das Disk-Image über den folgenden Link herunterladen:

https://github.com/DarwinNE/FidoCadJ/releases/download/v0.24.8/FidoCadJ_MacOSX.dmg

Anschließend können Sie die Kopie öffnen und `FidoCadJ.app` verschieben im `Applications` Ordner, in dem Sie es genau wie jede andere Macintosh-Anwendung verwenden können. Um FidoCadJ zu deinstallieren, ziehen Sie einfach `FidoCadJ.app` in den Mülleimer.

A.2 Linux

von Roby IZ1CYN

Voraussetzung: JRE 9 von Oracle und/oder OpenJDK 9 JRE (oder nachfolgende Versionen, die mit den Spezifikationen des Programms kompatibel sind) müssen installiert sein. Im Abschnitt A.2.1 wird die Installation des Programms ausschließlich mit Befehlen von einem Terminal beschrieben. Im Absatz ?? wird stattdessen die Interaktion mit einer grafischen Umgebung verwendet. Eine schlechte Konfiguration Ihrer Java-Laufzeitumgebung führt zu einer schlechten Leistung von FidoCadJ².

A.2.1 Auf jeder Plattform, vom Terminal aus

Laden Sie das Programm mit Befehl `wget` herunter.:

```
$ wget https://github.com/DarwinNE/FidoCadJ/releases/
      download/v0.24.8/fidocadj-0.24.8.jar
--00:48:18-- https://github.com/DarwinNE/FidoCadJ/releases/
      download/v0.24.8/fidocadj-0.24.8.jar
=> 'fidocadj.jar'
Resolution of github.com is being done... xxx.xxx.xxx.xxx
Connection to github.com is being done... xxx.xxx.xxx.xxx
connected.
HTTP request sent, waiting for answer... 200 OK
Length: 343,207 (335K) [application/java-archive]

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

00:48:30 (420.55 KB/s) - ''fidocadj-0.24.8.jar'' saved [xxx/
yyy]
$
```

Alternativ oder wenn Sie auf Probleme stoßen, können Sie die Datei in jedem Browser unter der folgenden URL herunterladen:

²Anmerkung des Herausgebers: Ich schwöre, es ist wahr! Bitte beleidigen Sie mich nicht, wenn Ihre Lieblings-Linux-Distribution eine sehr unzuverlässige Version von Java enthält.

<https://github.com/DarwinNE/FidoCadJ/releases/download/v0.24.8/fidocadj-0.24.8.jar>

und speichern Sie die Datei beispielsweise in Ihrem `/home/[Benutzername]/Download` Ordner (viele moderne Browser tun das standardmäßig).

Erstellen Sie ein neues Verzeichnis (aber zunächst werden wir durch `su` oder `sudo -s` zum Superuser):

```
$ su
-> enter password
# mkdir /usr/local/bin/fidocadj
```

... und wir können die heruntergeladene Datei dorthin verschieben (ersetzen Sie `<user>` durch den Benutzernamen des Kontos, von dem die Datei heruntergeladen wurde):

```
# mv /home/[user_name]/Download/fidocadj-0.24.8.jar /usr/
    local/bin/fidocadj
```

Machen Sie die Datei zu einer ausführbaren Datei

```
# chmod +x /usr/local/bin/fidocadj/fidocadj-0.24.8.jar
```

Und wir dürfen nicht vergessen, zu normalen Benutzern zurückzukehren:

```
# exit
```

Und jetzt können wir das Programm ausführen:

```
$ /usr/local/bin/fidocadj/fidocadj-0.24.8.jar
```

A.2.2 Auf einem grafischen System

Im Beispiel verwenden wir Ubuntu 8.04, aber für ältere oder neuere Versionen ändert sich nicht viel:

- Laden Sie die Datei über den Browser herunter oder verwenden Sie Gwget oder ähnliche Tools.
- Starten Sie unseren Dateimanager (Nautilus, Konqueror...) als Root (wenn wir keinen bestimmten Befehl im Menü haben, müssen wir ihn einfach über die Konsole mit `sudo nautilus` ausführen). Wir können das folgende Verzeichnis erstellen:

```
/usr/local/bin/fidocadj
```

und verschieben Sie dann die gerade heruntergeladene Datei dorthin, die in vielen Fällen in einem Verzeichnis wie `/home/[Benutzername]/Download/` abgelegt wird.

- Klicken Sie mit der rechten Maustaste auf die Datei, wählen Sie im Fenster die Registerkarte "Rechte" aus und fügen Sie das Häkchen vor "Ausführung der Datei als Programm zulassen" hinzu, wie in der Abbildung gezeigt A.1 (nur auf Italienisch...)
- Wählen Sie die Registerkarte "Öffnen als" und wählen Sie "OpenJDK Java 9 Runtime" oder "Oracle Java 9 Runtime" (oder eine nachfolgende Version).

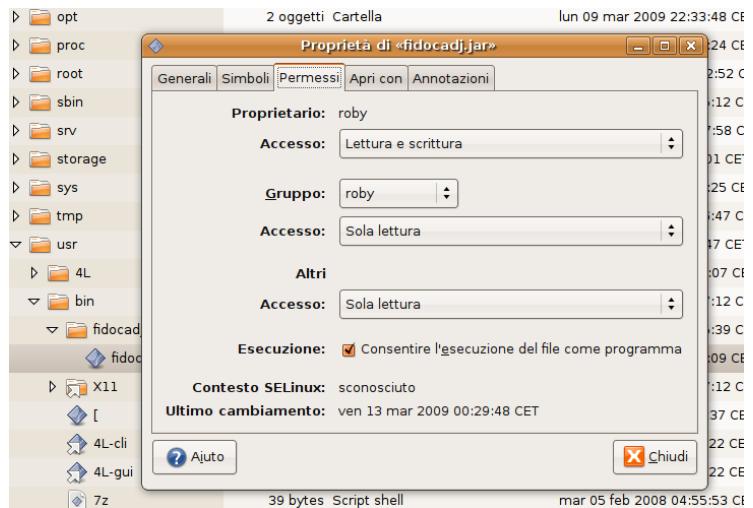


Abbildung A.1: Die Einstellung der Dateirechte, auf Ubuntu 8.04.

- Klicken Sie auf “Schließen” und wir können jetzt FidoCadJ ausführen: Doppelklicken Sie auf die ausführbare Datei, oder fügen Sie sie dem Hauptmenü hinzu. Der Befehl zum Hinzufügen lautet einfach `/usr/local/bin/fidocadj/fidocadj-0.24.8.jar`.

A.2.3 “Portable” Installation

von Max2433BO

Wenn Sie nichts auf dem System installieren können, kann FidoCadJ dennoch als “portable” Anwendung verwendet werden, die überall ausgeführt werden kann, ohne dass Superuser-Rechte erforderlich sind.

Erstellen Sie dazu einen Ordner `/home/[Benutzername]/fidocadj/` wo die folgenden Dateien abgelegt werden:

- `fidocadj-0.24.8.jar`
- `jdk-9.0.4` (javaSE 9, oder höher)
- ein Skript `fidocadj-0.24.8.sh` um FidoCadJ mit der entsprechenden Java-Version zu starten.

```
#!/bin/bash
export PATH=....../fidocadj/jdk-9.0.4/bin
export CLASSPATH=....../fidocadj/jdk-9.0.4/lib:../fidocadj
./fidocadj/jdk-9.0.4/bin/java -jar ../fidocadj/fidocadj_0
    .24.8.jar
```

Das Skript muss mit `chmod +x fidocadj-0.24.8.sh` ausführbar gemacht werden! in dem Verzeichnis, in dem sich das Skript befindet. Natürlich sollten Sie die Java-Version und ihr Verzeichnis entsprechend Ihren Bedürfnissen ändern.

A.2.4 Raspberry Pi 400

von Joop Nijenhuis

Der Raspberry Pi 400 ist ein sehr günstiger Computer. Billig zu sein bedeutet hier nicht, dass das System schlecht ausgeführt ist. So könnte die Zukunft aussehen. Im Inneren befindet sich ein 64-Bit-Quad-Core-ARM-Prozessor mit 4 GB RAM, Gigabit-Ethernet, WLAN und Bluetooth. Es verfügt über zwei HDMI-Videoanschlüsse, sodass es an fast alles angeschlossen werden kann, und verwendet eine microSD-Karte als Hauptlaufwerk. Es gibt 1 USB 2.0-Anschluss und 2 USB 3.0-Anschlüsse. Stromversorgung über einen USB-Typ-C-Anschluss. Für Experimente gibt es einen Erweiterungsport. Es verbraucht nur 15 Watt Energie, wahrscheinlich viel weniger, weil man nicht alles verbraucht. Diese Art von Computern wird bei jungen Menschen gefördert, um ihnen das Programmieren beizubringen und ihnen einen besseren Einblick in die Funktionsweise und Funktionsweise eines Computers zu vermitteln. Raspberry Pi wird mit einer eigenen Debian-basierten Betriebssystemdistribution namens NOOBS geliefert. Die Karte enthält außerdem OpenJDK Java Version 11.x.x. Alles ist bereits installiert! Sie könnten die microSD-Karte in die Rückseite des Kartenlesers stecken und das System laufen lassen, ob das sinnvoll ist, ist eine andere Frage. Wenn Sie einen anderen Hintergrund haben, wird Ihnen Linux vielleicht nicht leicht fallen. Seien Sie sich bewusst, dass es auch anders gemacht werden kann! Sie können den oben beschriebenen Weg gehen oder Sie können es auf meine Weise machen. Aus irgendeinem Grund mag die Raspberry Pi-Community Java nicht, ich weiß nicht warum, also keine Antworten dort. Ich fand “native”

Lösungen nicht besser und manchmal schlechter als die ich bereits in Java ausgeführt hatte. FidoCadJ ist eines davon und funktioniert gut mit/auf dem Pi. Was kommt als Nächstes? Erstellen Sie in /home/pi einen neuen Ordner und nennen Sie ihn “Apps” oder etwas Ähnliches. Öffnen Sie diesen Ordner und erstellen Sie erneut einen neuen Ordner mit dem Namen “fidocadj”. Kopieren Sie den Download “fidocadj-0.24.8.jar” in diesen Ordner.

Starten Sie den Texteditor und kopieren Sie die folgenden Zeilen hinein:

```
#!/bin/sh

#go to script's directory
progdir=${0%/*}
cd $progdir

#launch fidocadj
java -Duser.home=$progdir -jar fidocadj-0.24.8.jar 2>
fidocadj-bugs.txt
```

Speichern Sie es als “fidocadj.sh” im selben Ordner wie FidoCadJ. Sie können diese Datei zum Programm “Main Menu Editor” in der NOOBS-Distribution hinzufügen. Mit einigen Modifikationen kann dieses Setup für jedes andere Java-Programm verwendet werden. Parameter “-Duser.home=\$progdir” stellt sicher dass immer “\$progdir” verwendet wird. Die Datei “2>fidocadj-bugs.txt” sendet jedes Problem an die Datei “fidocadj-bugs.txt”. Keine Sorge, die Datei bleibt auf meinem System leer, aber für den Fall dass etwas schief geht, habe ich wahrscheinlich einige Hinweise wo ich nach dem Problem suchen kann.

Das Schöne an diesem System ist, dass es (bisher) immer funktioniert und Sie ALLE Informationen zum Programm in EINEM Ordner finden, falls Probleme auftreten oder Sie einfach alles löschen möchten. Da OpenJDK Java, Version 11, teil der NOOBS-Distribution ist, erhält es jedes Mal Updates wenn Sie das Pi-System aktualisieren und wenn es ein Update auf Java gibt. Noch etwas: Ich arbeite mit einem Wacom-Tablet und einem Stift als Maus. Funktioniert sehr gut mit FidoCadJ. Wenn Sie mehr lesen möchten;

<https://forums.raspberrypi.com/viewtopic.php?t=320892>

A.3 Windows

Wenn FidoCadJ eine Windows-Plattform erkennt, versucht es, das native Look and Feel zu verwenden.

A.3.1 So laden Sie FidoCadJ herunter und führen es aus

Wenn Sie Java installiert haben, müssen Sie oft nur die Datei `fidocadj-0.24.8.jar` herunterladen und mit einem Doppelklick ausführen. Wenn Ihr Betriebssystem nach dem Herunterladen als `ziparchive` erkennt, ist Java wahrscheinlich nicht auf Ihrem Computer verfügbar. Oracle ermöglicht Ihnen die aktuelle Version der Java-Laufzeitumgebung kostenlos herunterzuladen und zu installieren (ersetzen Sie “it” durch “de” oder die Bezeichnung für Ihr Land):

<http://www.java.com/it/download/>

OpenJDK kann heruntergeladen werden von:

<https://bell-sw.com/pages/downloads/>

A.4 Compileren van sources

Wenn Sie aus FidoCadJ-Quellcode kompilieren möchten, ist dies glücklicherweise eine einfache Aufgabe, da Sie lediglich ein funktionierendes Java SDK und GNU-Makeup benötigen. Weitere Einzelheiten finden Sie in der Datei README.md. Das FidoCadJ-Projekt verfügt über ein Build-Automatisierungssystem, das auf `make` basiert. Sie müssen zunächst das vollständige Quell-Repository von GitHub herunterladen. Die Make-Regeln in table A.1 sind implementiert und können über die Befehlszeile verwendet werden. Um den Quellcode von FidoCadJ zu studieren führen Sie `makecreatedoc` aus, um die Javadoc-Beschreibung zu erhalten die in der `doc`-Unterordner platziert wird.

Tabelle A.1: Liste der verfügbaren Make-Ziele zum Kompilieren von FidoCadJ aus Quellen.

Zeile	Beschreibung
<code>make</code>	Compile FidoCadJ (implicit)
<code>make clean</code>	Erase all the compiled classes
<code>make cleanall</code>	Do a clean, erase fidocadj.jar, Javadocs
<code>make compile</code>	Compile FidoCadJ (explicit)
<code>make createdoc</code>	Run Javadoc on all source files
<code>make createjar</code>	Prepare jar/fidocadj.jar
<code>make rebuild</code>	Do a clean and then run FidoCadJ
<code>make run</code>	Run FidoCadJ

Wenn Sie Windows verwenden, kann es hilfreich sein, die Datei `winbuild.bat` script zu installieren anstelle von `make`. Das Skript wird im Verzeichnis `dev_tools` abgelegt und muss mit einer aus `{run|clean|compile|force|rebuild}` ausgewählten Aktion verwendet werden.

A.5 Android

Im Jahr 2015 wurde eine spezielle Version von FidoCadJ für Android entwickelt. Ein Großteil des Quellcodes ist mit der PC-Version identisch, der gesamte Code der grafischen Benutzeroberfläche musste jedoch neu geschrieben werden. Es sollte auf einer Vielzahl von Geräten mit mindestens Android 4.0 funktionieren:

sowohl Smartphones als auch Tablets. Abbildung A.2 zeigt beispielsweise das Erscheinungsbild von FidoCadJ für Android mit einem Samsung S5-Smartphone.

Es gibt einige geringfügige Unterschiede zwischen der Android-Anwendung und der im Handbuch beschriebenen PC-Anwendung. Sie sind selbstverständlich perfekt mit dem Dateiformat kompatibel und können Zeichnungen austauschen.

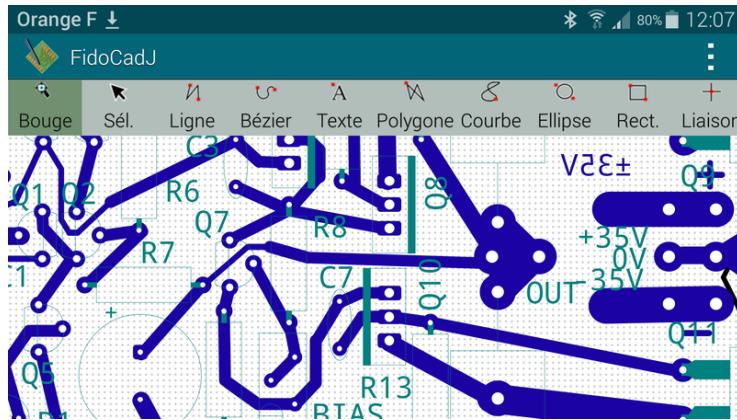


Abbildung A.2: FidoCadJ für Android, läuft auf einem Samsung S5 Smartphone.

- Sie können keine Benutzerbibliotheken erstellen und bearbeiten, aber Sie können die auf einem PC erstellten Bibliotheken verwenden.
- Sie speichern die Benutzerbibliotheken im externen Dateisystem (der “SD-Karte”), im **FidoCadJ/Libs** Ordner.
- Die Symbole werden aus Platzgründen nicht immer angezeigt: Sie müssen mit dem Finger über die rechte Seite des Bildschirms streichen, um sie anzuzeigen.
- In der Symbolleiste gibt es eine zusätzliche Schaltfläche, mit der Sie den angezeigten Teil der Zeichnung verschieben und zoomen können. Berühren Sie in diesem “Verschieben”-Zustand den Bildschirm und bewegen Sie Ihren Finger, um die Zeichnung zu verschieben (schwenken). Verwenden Sie zwei Finger und “ziehen” Sie sie zusammen, um den Zoom zu vergrößern oder zu verkleinern.

Im Jahr 2020 sieht die Zukunft der Android-Anwendung jedoch eher düster aus. Zunächst scheint es, dass Google Play die einzige wirksame Möglichkeit ist die Anwendung zu verbreiten. Ein Entwicklerkonto zum Veröffentlichen einer Anwendung kostet Geld. Es gibt Alternativen, die jedoch nicht weit verbreitet sind. Darüber hinaus wurde die Anwendung im Jahr 2014 mit Techniken entwickelt die Google nun offenbar auslaufen lässt. Noch wichtiger ist, dass ich angesichts der Zeit die ich FidoCadJ gewidmet habe, nicht allein mit der Pflege einer modernen App beschäftigt sein möchte. Wenn sich jemand darum kümmern möchte, treten Sie vor.

Anhang B

FidoCadJ-Beispiele

und wie es anders sein kann...

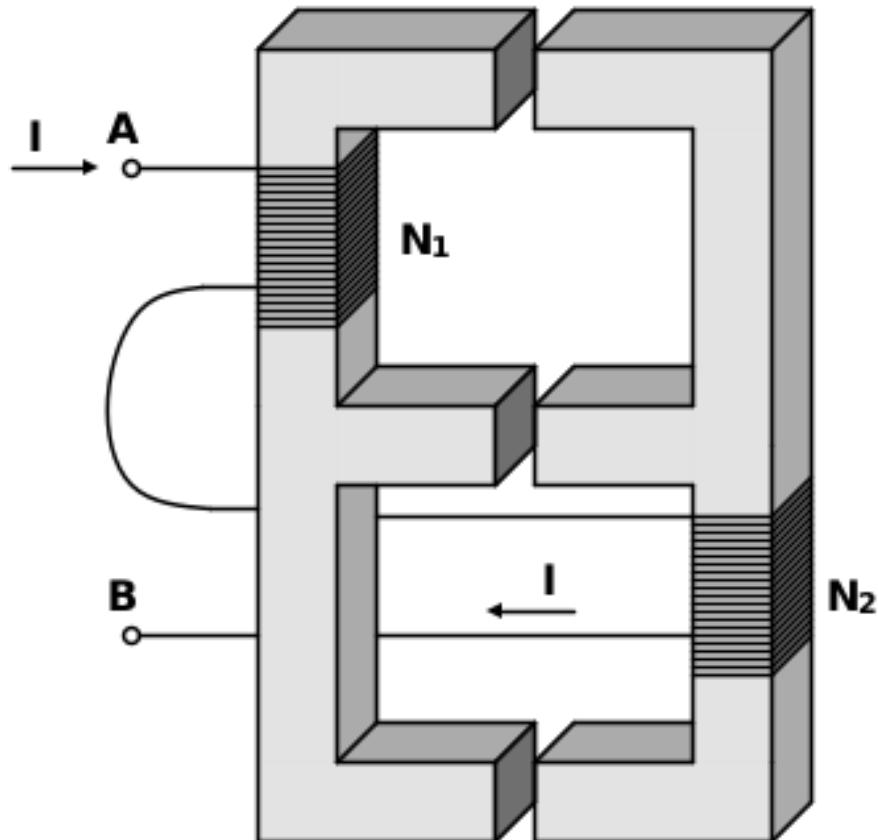


Abbildung B.1: Benutzerbeispiel DanteCpp (Österreich)

Dieses Beispiel stammt von <https://github.com/DarwinNE/FidoCadJ/issues/16> und wurde ursprünglich in einem Ubuntu-Linux-System erstellt. Ich habe den Code aus dem Forumsbeitrag kopiert, ihn in einen ASCII-Texteditor eingefügt und ihn mit einem Dateinamen und der Erweiterung ".fcd" gespeichert. Dann lesen Sie die Datei in FidoCadJ auf einem Raspberry Pi 400 (Niederlande) ein. Das Einzige, was ich korrigieren musste, waren die Überschriften "N1" und "N2", bei denen die Zahlen auf dem Buchstaben "N" standen. Der Rest der Zeichnung bleibt unverändert. Dann in Fido in ein PNG-Bild exportiert. Und das ist das Ergebnis.

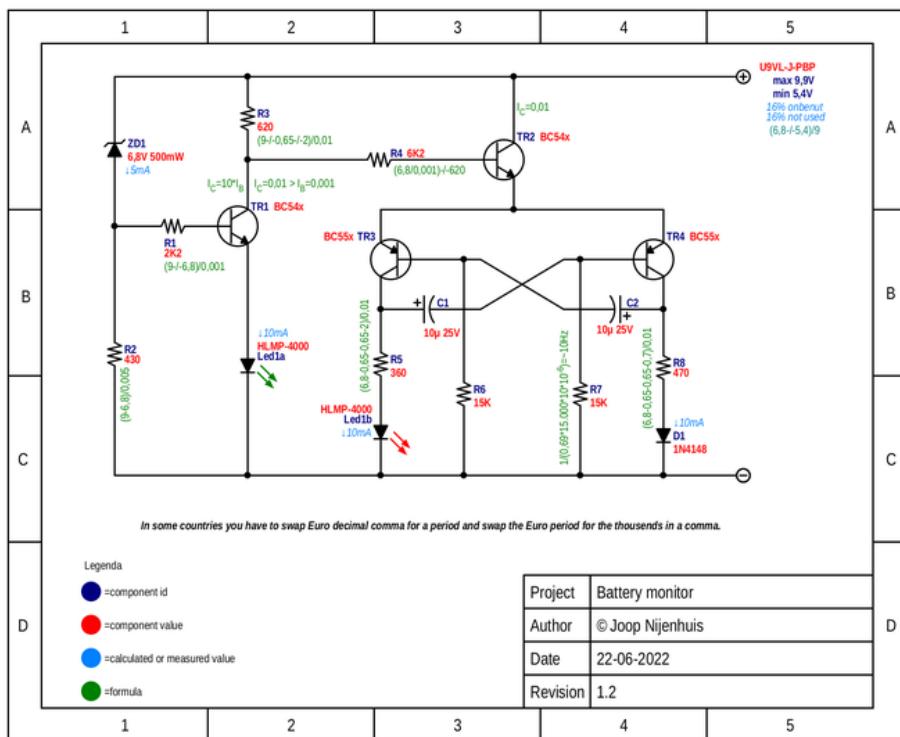


Abbildung B.2: Ein Beispiel einer FidoCadJ-Zeichnung mit unterschiedlicher Nutzung der verfügbaren Ebenen (lesen Sie: Farben).

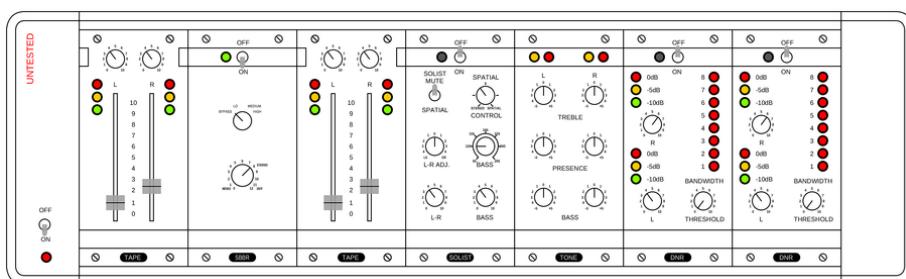


Abbildung B.3: Modulares Effekt Rack. Durch Vergrößern der Seite (>200%) werden Schaltflächen und Texte besser lesbar.

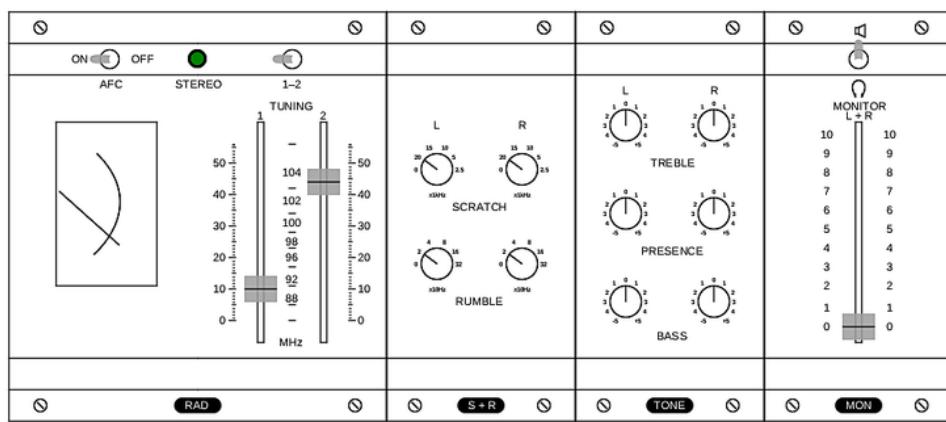


Abbildung B.4: Tuner aus Philips Mischverstärkereinheiten. Durch Vergrößern der Seite (>200%) werden Schaltflächen und Texte besser lesbar.

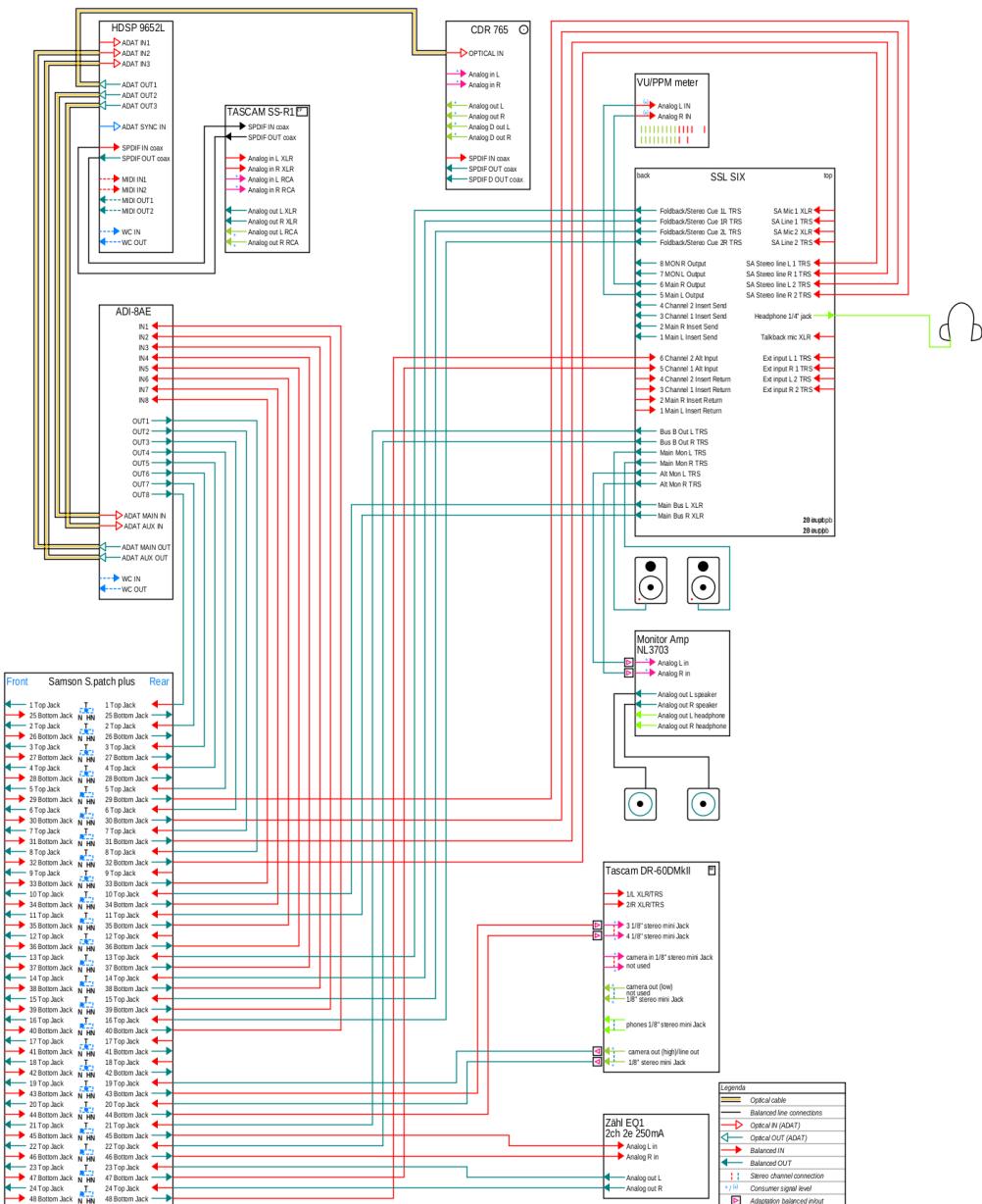


Abbildung B.5: Beispiel eines Tonstudio-Layouts (der eigentliche Aufbau ist viel komplizierter und passt nicht auf ein A4, die Idee wird klar sein). Durch das Vergrößern der Seite (>200%) werden Texte besser lesbar.

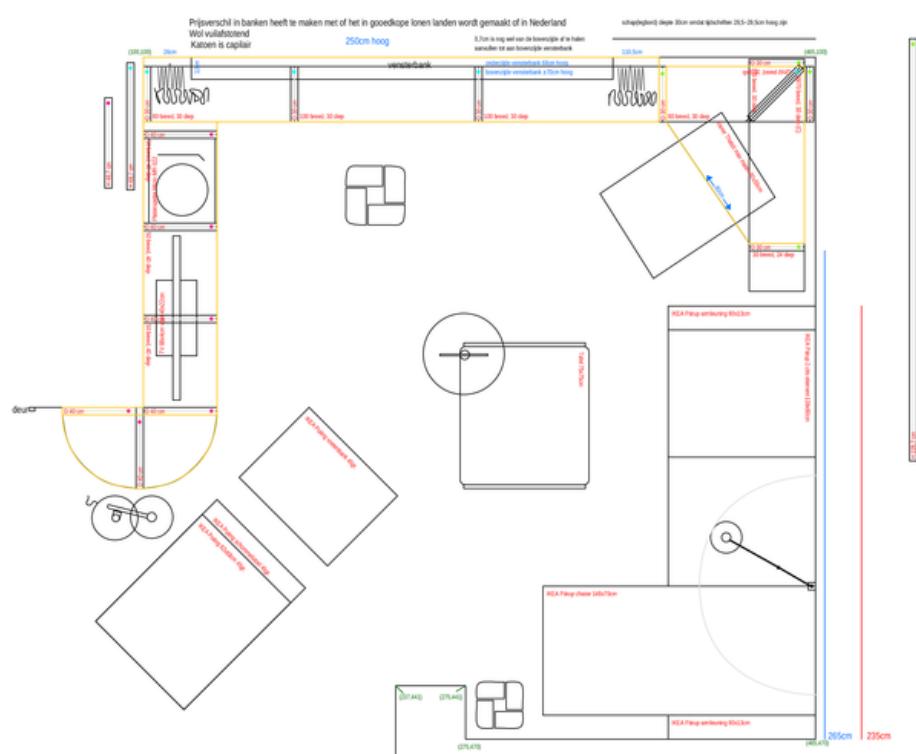


Abbildung B.6: Mein Versuch einer neuen Aufteilung des Wohnzimmers.

Anhang C

FidoCadJ art





Tiger, tiger, burning bright... By Igor Arbanas "elettrodomus".

Index

- ++ , 37
- LATEX , 25
- A4 , 20
- Abbildungsverzeichnis , x
- Abstrakt , iii
- Am Gitter ausrichten , 15
- Android , 53
- App Store , 47
- Arbeitsbereichsleiste , 6 , 8 , 15
- Auf einem grafischen System , 49
- Auf jeder Plattform, vom Terminal aus, 48
- Auflösung , 15
- ausdrucken , 19
- Auswahlmodus , 8 , 16
- autoplacer , 15
- autorouter , 15
- Bézier , 35 , 39
- Bézier-Kurve , 38
- Bézier , 9
- BE , 38 , 39
- Befehlszeile , 26
- Befehlszeilenoptionen , 26
- Beispiel eines Tonstudio-Layouts , 59
- Beispiel FidoCadJ-Zeichnung mit unterschiedlicher Nutzung Ebenen , 57
- Benutzerbeispiel DanteCpp (Österreich) , 56
- Bewegen , 9
- Bibliothek , 29
- Bibliothek aktualisieren , 30
- Bibliothekname , 30
- Bibliotheksformat , 44
- Bitmap-Format , 24
- CAD , 15
- CadSoft , 25
- Chinesisch , 3
- Code , 14
- Compileren van sources , 53
- Conclusie , 46
- Courier New anzuzeigen , 37
- CP , 39
- CP verwendet -1252 , 41
- CV , 39
- Danksagungen , iv
- Darstellung/Ebenen , 15
- Das Gitter , 15
- Datei/Optionen , 29
- Dateiname der Bibliothek , 30
- Deutsch , 3
- Diagonalverschiebung , 14
- die Grundelement , 6
- Die Philosophie von FidoCadJ , 1
- Die Schichten , 15
- Drag-and-Drop , 33
- E-Mail , 14
- Eagle , iv , 25
- Ebene , 17
- Ebenenaufbau , 42
- eckigen Klammern , 44
- Ein einfaches Schema , 10
- Ein Symbol aufteilen , 33
- Eine einfache Leiterplatte , 15
- Einer der Begründer der Schaltkreisanalyse , 61
- Einfügen Bibliothek(symbol) , 33
- Einführung , 1
- einfacher Text , 37
- Einfacher Text (veraltet) , 36
- elektrische Schaltpläne , 1
- Elektrische Verbindung , 40
- elektrischem Diagramm , 8
- Elektrischer Anschlussaufbau , 43
- elektrisches Diagramm , 15 , 35
- Element , 35
- Elemente zeichnen , 35

- Ellipse, 9, 35
- Englisch, 3
- Entwicklungsmaschine, 19
- EP, 38
- EPS, 2, 25
- Erweiterter Text, 37
- EV, 38
- Export, 24
- Exporteren, 24
- Farbe, 15
- FCJ, 41
- Fehlertoleranz, 43
- FidoCAD, 1, 2, 8, 23, 30, 34–36, 39, 41–45
- FidoCadJ und die Zukunft, 4
- FidoCadJ-Beispiele, 55
- FidoCadJ-Erweiterung, 13, 41
- FidoCadJ-Erweiterungen, 41
- FidoCadJ-Lizenz, vi
- FidoCadJ-Optionen, 26
- FidoCadJ-Quellcode, 53
- FidoCadJ.app, 47
- fidocadj.jar, 47
- FIDOLIB, 44
- FidoReadJ, 2
- FJC, 42, 44
- FJC A, 43
- FJC B, 43
- FJC C, 43
- FJC L, 42
- Forum, 14
- Französisch, 3
- Gefüllte oder leere Ellipse, 38
- Gefülltes oder leeres Polygon, 38
- Gefülltes oder leeres Rechteck, 36
- Gerade, 9, 35
- Geschichte von FidoCadJ, 2
- GIG, 22
- GitHub, 3
- Gitter zeigen, 15
- Griechisch, 3
- Gruppe, 32
- GTK+, 28
- header, 34, 44
- Header-Beschreibung, 34
- <http://www.matematicamente.it>, v
- IHRaM-Bibliothek, 29
- Inhaltsverzeichnis, viii
- Inkscape, 25
- Interpreter, 2
- iPad, 48
- iPhone, 48
- it.hobby.elettronica, iv, 2
- it.hobby.fai-da-te, 2
- Italienisch, 3
- Japanisch, 3
- jar, 26
- Java, 1–3, 26, 28
- JPG, 25
- JRE, 26, 48
- KiCad, 16
- Kodierung, 34, 41
- Kompatibilität mit FidoCAD, 45
- kompilieren, 53
- Koordinatensystem, 34
- Kopieren Bibliothek(symbol), 33
- Kopieren, nicht standardisierte Makros trennen, 30
- Kreuzung von Spuren, 16
- Kristallkugel, 43
- Kurve, 9, 35
- Löschen Bibliothek(symbol), 33
- Löten, 15
- LaTeX, 4
- Layer, 43
- Leiterbahn, 9, 17, 35, 40
- Leiterplatte, 1, 15, 20
- Leiterplattenschiene, 18
- LI, 35
- library
 - Standardbibliothek, 29
- Linienbreite, 43
- Linux, iv, 28, 48
- Liste verfügbaren Exportdateiformate, 25
- Lizenz, ii
- logische Einheit, 15, 22, 37
- Look & Feel, 28
- Lorenzo Lutti, 1, 36
- Macintosh, 7, 26
- macOS, 6, 7, 47
- macro, 8
- Makro, 44

- Makroaufruf, 35, 41
Matematicamente, 5
maximale Anzahl der Scheitelpunkte, 39
maximale Scheitelpunktezahl, 38
maximale Textlänge, 38
MC, 41, 44
Menüleiste, 8, 10, 15, 19, 24, 29, 30
Menüleisten, 6
Modulares Effekt Rack, 57
Motif, 28
MS-DOS-Eingabeaufforderung, 26

Name, 32
Natürliche kubische Kurve, 39
Neue Symbole definieren, 30
neuen Aufteilung des Wohnzimmers, 60
Newsgroup, 14
newsgroup, iv
Newsgroups, 2
Niederländisch, 3
NOOBS, 51

Objektorientierte Programmierung, 2
OpenJDK, 48
Oracle, 26, 48

PA, 40
Parametern, 35
PCB, 1, 35
 footprint, 8
PCB Pad, 9
PCB-Bibliothek, 44
PCB-Pad, 35, 40
PDF, 2, 25
PDF^LA_TE_X, 25
Pfeil- und Strichstile, 22
Pfeile, 22
Pfeilstil, 23
PGF, 2, 25
PL, 40
Platine, 18
Plattformspezifische Informationen, 47
PNG, 25, 27
Polygon, 9, 16, 35
polygoon, 17
Portable Installation, 51
Postscript, 25
PP, 38
Prüfpunkt, 11

Press&Peel, 19
PV, 38

Quaqua, iv, 47

R41-Transfers, 15, 17
Raspberry Pi 400, 6, 7, 51
Raster, 15
Rechteck, 9, 16, 35
recursion, 44
repository, 53
resistor, 12
RP, 36
RV, 36

SA, 40
Schicht, 15
Schlüssel umbenennn, 33
Schlussel, 32
Schnellsuche, 8
SCR, 25
Segment, 35
Siebdruck, 15, 16
Snow Leopard, 47
So laden Sie FidoCadJ herunter und führen es aus, 53
So laden Sie FidoCadJ unter macOS herunter und führen es aus, 47
SourceForge, 3
Spanisch, 3
Speichern, nicht standardisierte Makros trennen, 30
Spline, 39
Standardbibliothek, 10, 44
Standardbibliotheken, 44
Sternchen, 37
Strichstil, 23
Strichstile, 22
Stromkreis, 30, 37
Superuser, 49
SVG, 25
Symbol, 10
Symbole, 8
Symbolleiste, 6, 8, 12

Tabellenliste, xi
TE, 37
Terminal, 26
Text, 9, 35

text
 gedreht, 37
 gespiegelt, 37
 stil, 37
Texte spiegeln, 18
Tiger, tiger, burning bright..., 62
Toleranz für Syntaxfehler, 43
Transistor, 10
Tschechisch, 3
Tuner aus Philips Mischverstärkereinheiten,
 58
TY, 37

Ubuntu, 49, 50
Umbenennen Bibliothek(symbol), 33
Unix, 26
Ursprung, 32
Usenet-Gruppen, 1
UTF-8, 34, 41

Vektorformat, 24
Vektorzeichnung, 6
veraltet, 36
Verbindung, 9, 35
verlustbehaftete Komprimierung, 25
Verwendung des Lineals, 22
Verwendung von Bibliotheksdateien, 29
Vorhandene Symbole ändern, 33

Wählen, 9
Wacom-Tablet, 52
Windows, 1, 26, 52
WInE, 1
www.electroyou.it, 4, 5
www.grix.it, v, 5

Zeichenabmessungen, 18
Zeichenelement, 6
Zeichenwerkzeuge, 6
Zeichnen mit FidoCadJ, 6
Zeichnungsformat, Makros & Bibliothek,
 34
Zoom, 36
zoom, 9
Zusammenfassung FidoCadJ Zeichen-
 befehle, 9

Dieses Handbuch wurde mit PDFLATEX und TeXstudio auf einem Rasperry Pi 400 geschrieben. Listen wurden mit `listings`-Paket zusammengestellt. Das `pgf`-Paket wird für die Abbildung 2.6 und auch für das Bild auf Seite 61 verwendet. Siehe die Präambel der tex-Datei für die benötigten Pakete (mit einigen Kommentaren). Alle diese Pakete sind im CTAN-Repository verfügbar.