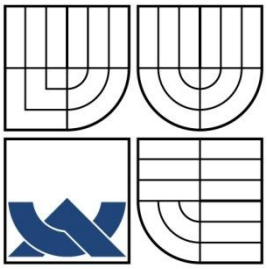
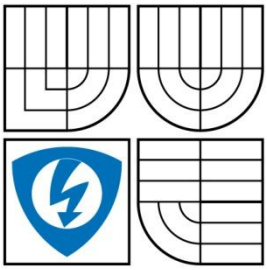


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIOELECTRONIC

ELEKTRONICKÝ KATALOG AKTIVNÍCH SOUČÁSTEK

ELECTRONIC CATALOGUE OF THE ACTIVE DEVICES

DIPLOMOVÁ PRÁCE

MASTER´S THESIS

AUTOR PRÁCE Bc. ALEŠ AMBROŽ

AUTHOR

VEDOUCÍ PRÁCE ING. JIŘÍ PETRŽELA, PH.D.

SUPERVISOR

BRNO 2009

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

lta elektrotechniky
nunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student: Bc. Aleš Ambrož ***ID:*** 89961
Ročník: 2 ***Akademický rok:*** 2008/2009

NÁZEV TÉMATU:

Elektronický katalog aktivních součástek

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s parametry a možnostmi třídění (vyhledávání) aktivních součástek typu bipolární a unipolární tranzistor, operační zesilovač DISO, CFOA, transadmitanční zesilovač atd.

Ve Vámi zvoleném programovém prostředí vytvořte kostru programu umožňujícího multiparametrické vyhledávání součástek. K tomuto účelu vytvořte krátkou knihovnu elektronických součástek s parametry dostupnými v příslušných katalogových listech.

Dopracujte GUI programu a rozšiřte knihovnu prvků o typy součástek dané zadáním. Důraz je kladen zejména na uživatelskou vstřícnost programu a jeho snadné intuitivní ovládání.

DOPORUČENÁ LITERATURA:

[1] Analog Devices [online], dostupné na www.analog.com.
[2] Texas Instruments [online], dostupné na www.ti.com.
[3] Linear Technology [online], dostupné na www.linear.com.

Termín zadání: 9.2.2009 ***Termín odevzdání:*** 29.5.2009

Vedoucí práce: Ing. Jiří Petržela, Ph.D.

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Bc. Aleš Ambrož
Bytem:	Na Skalce 18, Boskovice, 680 01
Narozen/a (datum a místo):	24. srpna 1983 v Boskovicích

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☒ diplomová práce
   ☐ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Elektronický katalog aktivních součástek
Vedoucí/ školitel VŠKP:	Ing. Jiří Petržela, Ph.D.
Ústav:	Ústav radioelektroniky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 29. května 2009

…….. ……

Nabyvatel Autor

ABSTRAKT

Cílem této práce je vytvoření elektronického katalogu vybraných typů aktivních součástek, který umožňuje jejich třídění, editaci, atd. Práce obsahuje stučný popis databází a zvolených programovacích jazyků a dále popis funkce vytvořeného programu.

KLÍČOVÁ SLOVA

Katalog, Součástek, Třídění, Elektronický, Databáze

ABSTRACT

This work deals with creating an catalogue of active devices. The catalogue enables editing and sorting the devices etc. This work describes briefly databases, choosen programming language and deals with the functions of the catalogue.

KEYWORDS

Katalogue, Devices, Sorting, Electronic, Database

AMBROŽ, A. *Elektronický katalog aktivních součástek.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2009. 35 s., 13 s. příloh. Diplomová práce. Vedoucí práce: Ing. Jiří Petržela, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Elektronický katalog aktivních součástek jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 29. května 2009 .....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Jiřímu Petrželovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 29. května 2009 .....

podpis autora

Obsah

1. Úvod

Cílem této práce je vytvoření elektronického katalogu aktivních součástek. Konkrétně bipolárních a unipolárních tranzistorů, operačních zesilovačů DISO, CFOA a transadmitančních zesilovačů.

První část se zabývá obecným popisem databází. V další části jsou stručně popsány použité programovací jazyky a vývojové prostředí, v němž byla aplikace naprogramována. Třetí část se zabývá výběrem nejdůležitějších parametrů jednotlivých typů součástek, podle kterých probíhá třídění. Čtvrtá část popisuje vlastní program, jeho funkce a ovládání.

2. Databáze

2.1 Vývoj databází

V 60. letech tohoto století vzniká současný pojem databáze, entita, atribut entity a vazba mezi entitami. To jsou základní pojmy z oblasti databází. Databázi si lze představit jako soubor dat, který slouží pro popis reálného světa (např. evidence školní knihovny, sklad chemikálií, evidence studentů). Entita je prvek reálného světa (např. člověk, stroj, vyučovaný předmět, město), který je popsán svými charakteristikami (vlastnostmi). Ty se většinou považují za atribut (např. jméno, příjmení, stav, plat, hmotnost).

Dalším důležitým pojmem je vazba mezi entitami**.** Jednotlivé entity odpovídající prvkům z reálného světa, mají mezi sebou určitý vztah. Např. každý člověk má právě jedny osobní údaje. To hovoříme o vazbě typu 1:1. Dalším typem je vazba 1:N, jíž bude odpovídat např. skutečnost, že jeden člověk může vlastnit více kreditních karet (ale jedna kreditní karta může být vlastněna pouze jedním člověkem). Posledním typem vazby je M:N. Zde není žádné omezení. V této době vzniká ještě jeden pojem, a to databázový model**.** Ten byl zaveden zejména matematiky jako prostředek pro popis databáze. Zpočátku se používaly modely dvojího typu: hierarchický (založen na modelování hierarchie mezi entitami se vztahy podřízenosti a nadřízenosti) a dále síťový (vychází z teorie grafů, uzly v grafu odpovídají entitám a orientované hrany definují vztahy mezi entitami). V 70. letech se uvedené databázové modely ukázaly být nedostatečné (objevily se problémy s realizací a implementací vazby M:N), a proto vznikl relační model, který se stal standardem a používá se dodnes. [3]

2.2 Relační databáze

Základním pojmem relačních databází je relace**.** Relaci si lze představit jako tabulku, která se skládá ze sloupců a řádků. Sloupce odpovídají jednotlivým vlastnostem (atributům) entity. Údaje v jednom řádku tabulky zobrazují aktuální stav. Například tabulka `ZAMĚSTNANEC` má sloupce `ČÍSLO`, `JMÉNO`, `PŘÍJMENÍ`, `DAT_NAR`, `PLAT` a `SMLOUVA_OD`. Tabulka je základním stavebním kamenem pro budování celé databáze. Relace tedy odpovídá celé tabulce a prvku relace odpovídá jeden konkrétní řádek. Jeden řádek bývá často nazýván databázovým záznamem. Soubor tabulek (relací) pak tvoří celou databázi (relační schéma)**.** [3]

2.3 Základní pojmy

Hodnotou většinou rozumíme uživatelská data. Každý sloupec v tabulce má svůj datový typ (např. celé číslo, řetězec, datum, logická hodnota, atd.).

Pro práci s databázovými tabulkami je nutné mít alespoň jednu položku (sloupec), jejíž hodnota bude jednoznačně identifikovat záznam v tabulce. Takováto položka se nazývá primární klíč. V případě tabulky `ZAMĚSTNANEC` lze za primární klíč zvolit položku `ČÍSLO`. Primární klíč má tu vlastnost, že jeho hodnota je jedinečná, tj. pro žádné dva řádky v tabulce nemůže nastat situace, že by hodnota primárního klíče byla totožná. Databázové systémy většinou umožňují definovat jako primární klíč n-tici položek, např. dvojici nebo trojici položek. V takovém případě se mohou některé položky v klíčích opakovat, ale nesmí být shodné všechny položky dvou primárních klíčů najednou.

Neméně důležitá je i funkční závislost. Například plat zaměstnance závisí na tom, jakou vykonává funkci, tj. plat závisí na funkci, tedy `FUNKCE->PLAT`. [3]

2.4 Normální formy

Pojem normálních forem se používá ve spojitosti s dobře navrženými tabulkami. Správně vytvořené tabulky splňují 4 základní normální formy. [3]

2.4.1 První normální forma (1NF)

První, nejjednodušší, normální forma (značíme **1NF**) říká, že všechny atributy jsou atomické, tj. dále již nedělitelné (jinými slovy, hodnotou nesmí být relace). To znamená, že jedna databázová položka obsahuje pouze jedinou hodnotu. [3]

2.4.2 Druhá normální forma (2NF)

Tabulka splňuje 2NF, právě když splňuje 1NF a navíc každý atribut, který není primárním klíčem je na primárním klíči úplně závislý. To znamená, že se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. Z definice vyplývá, že problém 2NF se týká jenom tabulek, kde volíme za primární klíč více položek než jednu. Jinými slovy, pokud má tabulka jako primární klíč jenom jeden sloupec, pak 2NF je splněna triviálně. Obecně převedení do tabulky, která již bude splňovat 2NF, znamená rozpad na dvě a více tabulek, kde každá už bude splňovat 2NF. To se nazývá dekompozice relačního schématu. Pokud tabulka nesplňuje 2NF, dochází často k redundanci. [3]

2.4.3 Třetí normální forma (3NF)

Relační tabulky splňují třetí normální formu (3NF), jestliže splňují 2NF a žádný atribut, který není primárním klíčem, není tranzitivně závislý na žádném klíči. Postup, jak dostat tabulky do 3NF, je podobný jako v případě 2NF, tj. opět se provede dekompozice. [3]

2.4.4 Boyce - Coddova normální forma (BCNF)

Poslední prakticky užívanou formou je tzv. Boyce-Coddova normální forma **(**BCNF**).** Tabulka splňuje BCNF, právě když pro dvě množiny atributů `A` a `B` platí: `A->B` a současně `B` není podmnožinou `A`, pak množina `A` obsahuje primární klíč tabulky. Tato forma zjednodušuje práci s tabulkami, ve většině případů, pokud dobře postupujeme při tvorbě tabulek, aby splňovaly postupně 1NF, 2NF a 3NF, forma BCNF je splněna. [3]

2.5 Systém řízení báze dat

Důležitým rysem databází je práce se soubory. Před započetím "éry databází" se k souborům přistupovalo skoro výhradně tzv. souborově orientovaným přístupem. Ten spočívá v tom, že každý program zpracovává svá vlastní data. Tento přístup dodnes přetrvává, ale spíše jen v oblastech, kde hlavním účelem aplikace nejsou přímo data, ale výpočty na jejich základě. Nevýhody tohoto systému jsou zřejmé:

Izolace dat - nelze podchytit vazby mezi daty už na úrovni dat, ale až na úrovni aplikace.
Duplicita dat - přestože několik aplikací pracuje se stejnými daty, každá z nich má svou vlastní kopii těchto dat.
Závislost dat na programu - data a programy jsou vzájemně závislé.

Druhý přístup k datům se nazývá databázový přístup. Ten odstraňuje nevýhody souborově orientovaného přístupu a lze jej charakterizovat dvěma základními body:

- Definice dat je provedena mimo aplikační programy (data mohou být uložena nezávisle na programech).
- V aplikačních programech není zabudován mechanismus přístupu k datům.

Systém řízení báze dat je programový systém, který zajišťuje následující akce a činnosti:

- *definování struktury dat*
- *ukládání dat*
- *výběr dat*
- *opravu dat*
- *komunikaci mezi uživatelem a systémem*

Systém řízení báze dat jako celek poskytuje například prostředky pro definování dat, pro ukládání, změnu, vymazání a vyhledání dat, dále zajišťuje bezpečnost systému a správu přístupových práv. Umožňuje také sdílený přístup více uživatelů a obnovu systému po chybě. [2]

2.6 Dotazy

Chceme-li získat z databáze data splňující nějakou podmínku, použijeme *dotaz.* Možnost klást dotazy na informace uchovávané v databázi je jedním z nejsilnějších motivů uživatele pro pořízení databázového systému.

Databázový dotaz by měl být nezávislý na jazykovém vyjádření. Tentýž dotaz je možné realizovat různými prostředky v různých systémech řízení báze dat. Dotaz by neměl být závislý na tom, jak je databáze fyzicky uspořádána.

Dotazovací jazyk umožňuje uživateli klást databázové dotazy. Zapíšeme-li v dotazovacím jazyku nějaký výraz, dotazovací jazyk převede tento výraz na databázový dotaz. [2]

2.7 Lokální a vzdálené databáze

Umístění databází můžeme dělit podle dvou hledisek - podle místa uložení a podle způsobu pracování.

Z hlediska místa uložení mohou být data uložena přímo na uživatelově počítači (lokální uložení, slouží k výhradnímu používání jedním uživatelem), nebo na jiném počítači (na tzv. serveru, který umožňuje sdílený přístup více uživatelů zároveň).

Z hlediska způsobu zpracování dělíme databáze na *Desktop* a *klient/server*. Práce s databázemi *Desktop* spočívá v tom, že všechny operace s daty provádí lokální počítač. Na tomto lokálním počítači nemusí být data uložena (mohou být uložena na serveru, z něho přečtena, v lokálním počítači zpracována a vypsána). U *klient/server* databází naproti tomu lokální počítač zpracuje pouze požadavek, ten odešle na server, přímo v něm proběhne zpracování požadavku a zpět je poslán pouze výsledek výpočtu. [2]

3. Použité programovací jazyky a vývojové prostředí

Pro vytvoření daného programu byl zvolen programovací jazyk C++ a SQL. Jazyk C++ byl vybrán, protože jde v současnosti o nejrozšířenější programovací jazyk, je pro něj k dispozici nepřeberné množství knihoven a vývojových prostředí. Jazyk SQL v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje pro manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací).
Program byl realizován v Borland C++ Builder 6.0, a to z několika důvodů:

- jde o rozšířené a propracované vývojové prostředí
- umožňuje jednoduchý návrh vizuálního prostředí aplikace
- velké množství volně dostupných komponent
- podpora databází

Pro práci s databázemi v Borland C++ Builderu je určen BDE (Borland Database Engine) a klientský program Database Desktop. BDE umožňuje přístup k mnoha typům databází. Pomocí systému Database Desktop je možné vytvářet, ukládat a znovu otvírat tabulky, pracovat s daty v tabulkách apod.

3.1 jazyk SQL

Jazyk SQL (Structured Query Language) patří mezi tzv. deklaratitivní programovací jazyky, což v praxi znamená, že kód jazyka SQL nepíšeme v žádném samostatném programu (jako by tomu bylo např. u jazyka C nebo Pascal), ale vkládáme jej do jiného programovacího jazyka, který je již procedurální. Se samotným jazykem SQL můžeme pracovat pouze v případě, že se terminálem připojíme na SQL server a na příkazový řádek bychom zadávali přímo příkazy jazyka SQL. Skládá se z několika částí. Některé části jsou určeny pro administrátory a návrháře databázových systémů, jiné pak pro koncové uživatele a programátory. První částí jazyka SQL je jazyk *DDL - Data Definition Language*. Jedná se o jazyk pro vytváření databázových schémat a katalogů. Způsob ukládání tabulek definuje jazyk *SDL - Storage Definition Language*. Třetí částí pro návrháře a správce je jazyk *VDL - View Definition Language*, určující vytváření pohledů (pohled si lze představit jako virtuální tabulku složenou z různých jiných tabulek).

Poslední částí je jazyk *DML - Data Manipulation Language*, který obsahuje základní příkazy. Mezi ně patří příkazy SELECT, DELETE, INSERT a UPDATE, použité v navrhované aplikaci. [3]

Příkaz SELECT

Tento příkaz slouží k získání dat z tabulky. Syntaxe je následující:

SELECT *Název_sloupce* FROM *Název_tabulky* WHERE *Podmínka*

Povinné jsou pouze slova SELECT a FROM. V tomto případě získáme všechny položky ze sloupce *Nazev_sloupce*. Pokud požadujeme pouze data, která vyhovují nějakým kritériím, přidáme do příkazu část WHERE *Podmínka,* popřípadě WHERE *Podmínka1* AND *Podmínka2,* atd. Potom získáme pouze data, která odpovídají zadaným podmínkám.

Příkaz DELETE

Příkazem DELETE odstraníme data z tabulky. Syntaxe:

DELETE FROM *Název_tabulky* WHERE *Podmínka*

Dojde k vymazání těch dat, která opět vyhovují zadaným podmínkám.

Příkaz INSERT

Slouží k vložení nových dat do tabulky. Syntaxe:

INSERT INTO *Název_tabulky* (*Názvy_sloupců*) VALUES *Hodnota1, Hodnota2,…*

Názvy sloupců jsou nepovinné, pokud nejsou zadány, ukládají se hodnoty postupně zleva doprava.

Příkaz UPDATE

Tímto příkazem se aktualizují již existující data v tabulce. Syntaxe:

UPDATE *Název_tabulky* SET *Sloupec1 = Hodnota1, Sloupec2 = Hodnota2,…*

3.2 Jazyk C++

Jazyk C++ je rozšířením jazyka C. Navíc obsahuje především podporu objektově orientovaného programování (OOP), možnost psát šablony, zpracování chybových stavů pomocí výjimek a přetěžování funkcí a operátorů. Je ve velké míře zpětně kompatibilní. C++ má vlastní standardní knihovnu, ale používá se také standardní knihovna jazyka C.

Jazyk C++ se používá zejména, pokud:

- je potřeba objektově orientované programování, např. GUI, simulace, datové struktury, atd.
- je potřeba využívat další vlastnosti C++, např. šablony, vyjímky, prostory jmen, atd.
- je třeba využít standardní C++ knihovnu [8]

4. Výběr parametrů součástek

Jednou z funkcí programu je možnost vyhledávat součástky podle hodnot jejich parametrů. Vzhledem ke značnému počtu těchto parametrů bylo nutné zvolit pro každý typ součástky několik nejdůležitějších, podle kterých je možné součástky třídit.

4.1 Bipolární tranzistory

1. Maximální napětí kolektor-emitor U_{CEmax}

Je omezeno průrazem kolektoru.

2. Tranzitní kmitočet f_T

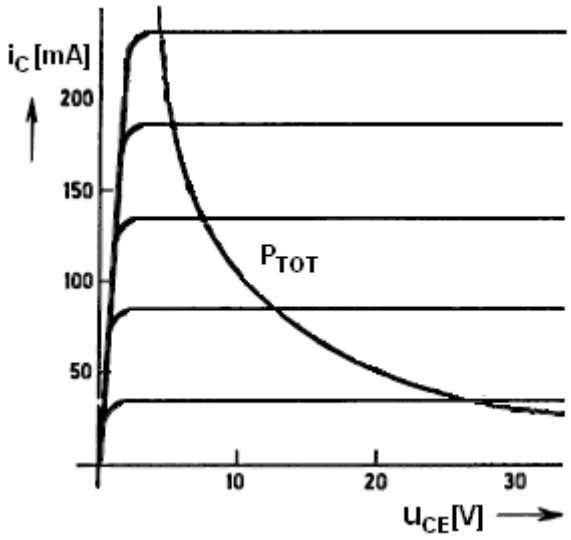
Je to kmitočet, při kterém má zesílení hodnotu 1, tj. OdB.

3. Maximální proud kolektorem Ic_{max}

Je určen konstrukcí tranzistoru a dovoleným oteplením.

4. Maximální výkonová ztráta P_{tot}

Souvisí s U_{CEmax} a Ic_{max}. Je dána maximální dovolenou teplotou přechodu a chlazením tranzistoru.



Obr. 4.1: Maximální výkonová ztráta

5. Proudový zesilovací činitel β

Je to podíl změny proudu kolektorem a změny proudu bází.

$$\beta = \frac{\Delta i_C}{\Delta i_B} \tag{1}$$

4.2 Unipolární tranzistory

1. Napětí U_{DS}

Obdobné jako U_{CE}, rovněž omezeno průrazem drainu.

2. Proud I_d

Opět dán konstrukcí tranzistoru.

3. Maximální výkonová ztráta P_{TOT}

4.3 Operační zesilovače

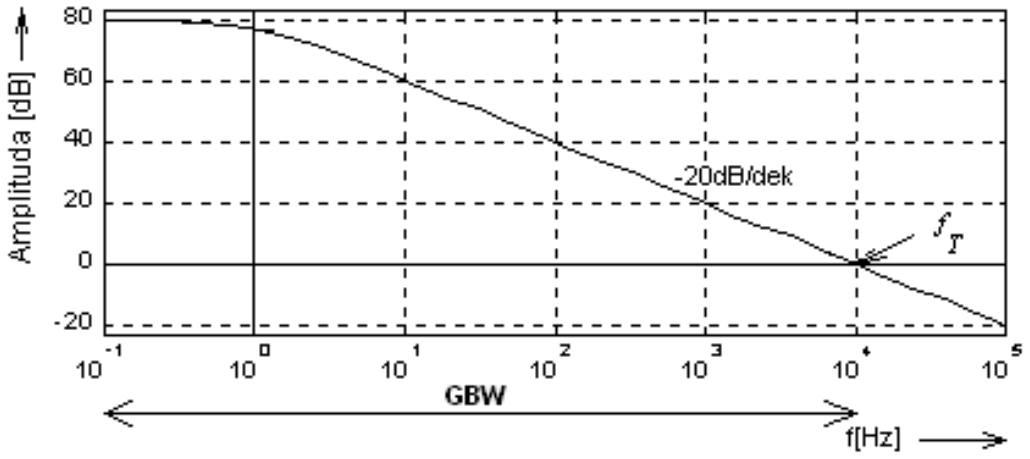
1. Zesílení Au [dB]

Je definováno jako poměr výstupního a vstupního napětí.

$$A_u = \frac{A_{uOUT}}{A_{uIN}}, \quad A_{u\,dB} = 20log\frac{A_{uOUT}}{A_{uIN}} \tag{2,3}$$

2. Šířka pásma GBW [MHz]

Je to pásmo kmitočtů, pro které je zesílení větší než 0dB.



Obr. 4.2: Definice šířky pásma GBW

3. Rychlost přeběhu SR [V/μs]

Je to maximální rychlost změny výstupního napětí operačního zesilovače.

$$SR = \frac{\Delta u_{výst}}{\Delta t}[V/\mu s] \quad (4)$$

4. Napájecí napětí U_{CC}

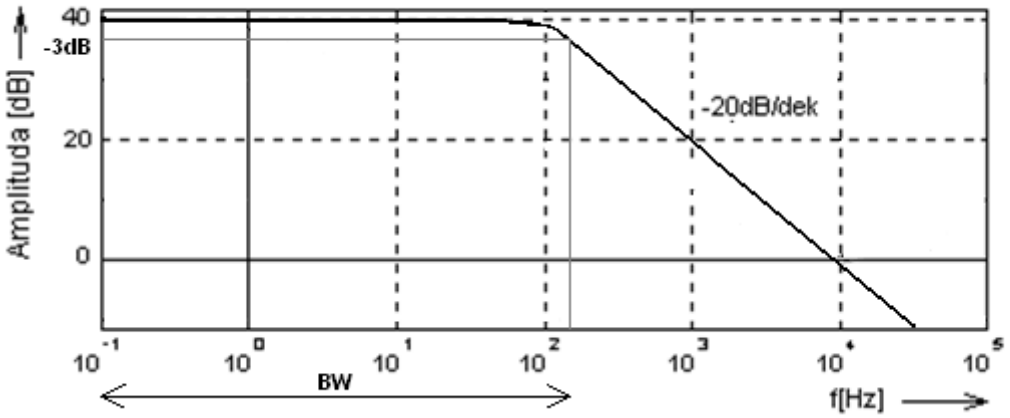
5. Proudový odběr I_{CC}

Je to maximální proud, který OZ odebírá ze zdroje.

4.4 CFOA

1. Šířka pásma BW [MHz]

Je to pásmo kmitočtů, pro které amplituda výstupního signálu neklesne o více než 3dB.



Obr. 4.3: Definice šířky pásma BW

2. Rychlost přeběhu SR [V/μs]

Je to maximální rychlost změny výstupního napětí operačního zesilovače.

$$SR = \frac{\Delta u_{výst}}{\Delta t}[V/\mu s] \quad (5)$$

3. Napájecí napětí U_{CC} [V]

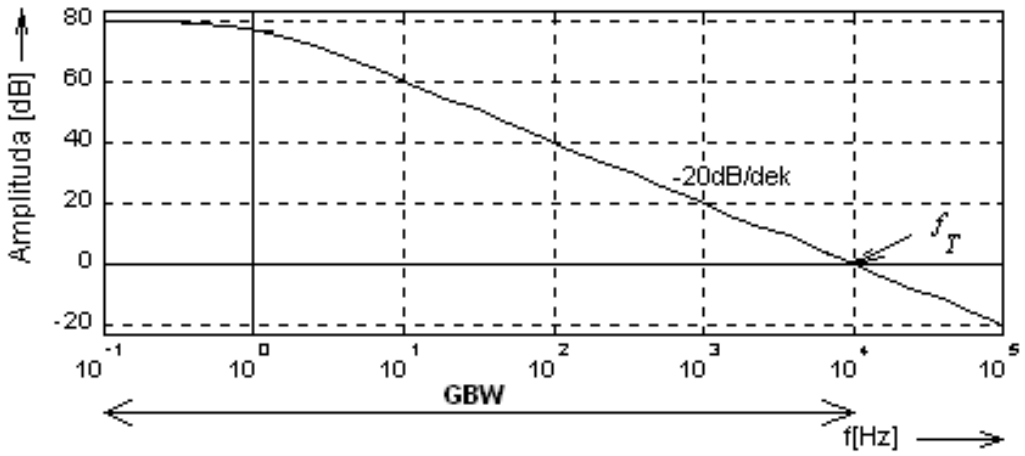
4. Vstupní odpor neinvertujícího vstupu $R_{IN}+$

5. Vstupní odpor invertujícího vstupu $R_{IN}-$

4.5 OTA

1. Šířka pásma GBW [MHz]

Je to pásmo kmitočtů, pro které je zesílení větší než 0dB.



Obr. 4.4: Definice šířky pásma GBW

2. Vstupní odpor R_{IN} [kΩ]

3. Výstupní odpor R_{OUT} [MΩ]

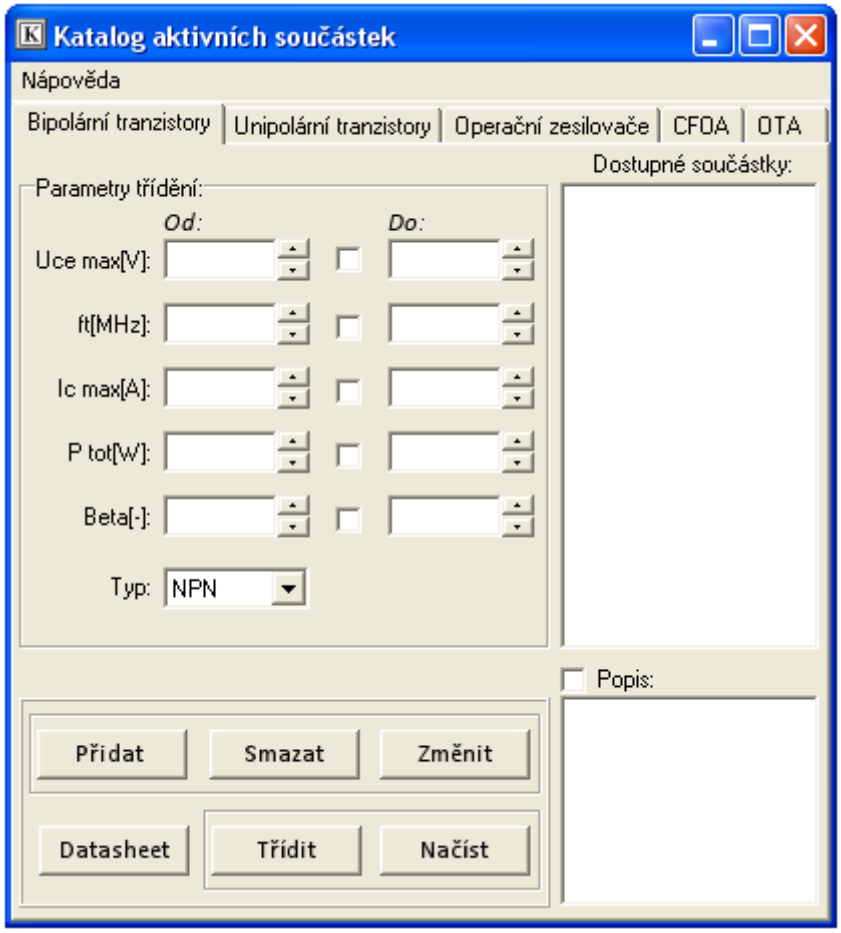
4. Výstupní proud I_{OUT} [mA]

5. Napájecí napětí U_{CC} [V]

5. Popis programu

5.1 Návrh uživatelského rozhraní

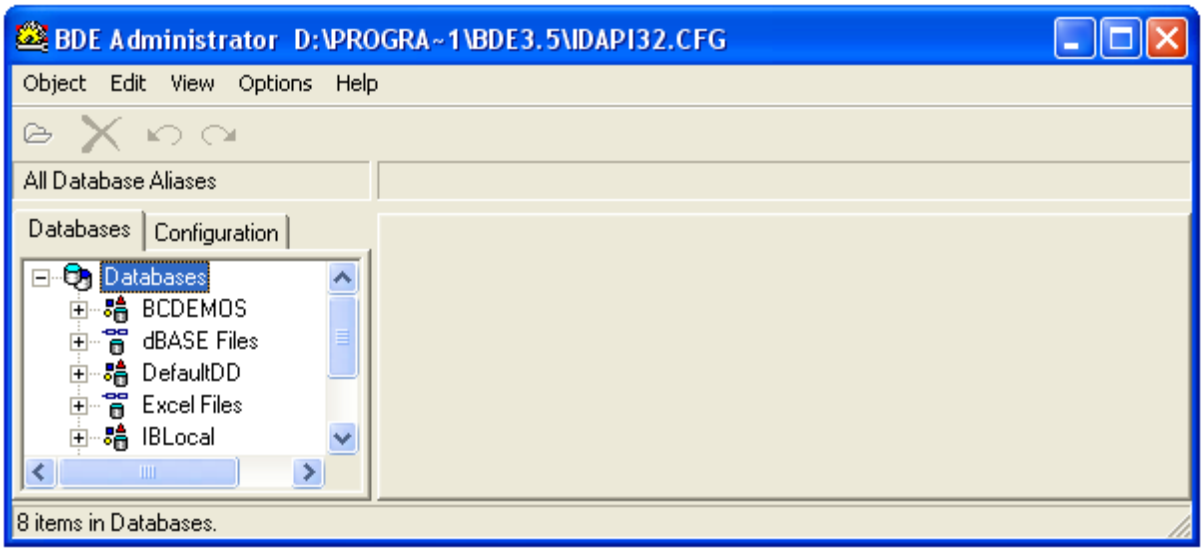
Na obrázku 5.1 je zobrazen formulář navrhovaného elektronického katalogu součástek. Je složen z pěti záložek, přičemž na každé z nich jsou umístěny komponenty pro daný typ součástky. V levé části formuláře jsou umístěny editační pole pro zadání parametrů třídění a komponenty pro výběr podkategorií jednotlivých typů součástek. V pravé části je pole, ve kterém se zobrazují dostupné součástky spolu s polem pro zobrazení popisu označené součástky. Ve spodní části jsou umístěna tlačítka pro práci s daty. Při návrhu byl kladen důraz na přehlednost a jednoduché ovládání.



Obr. 5.1: Vytvořený program

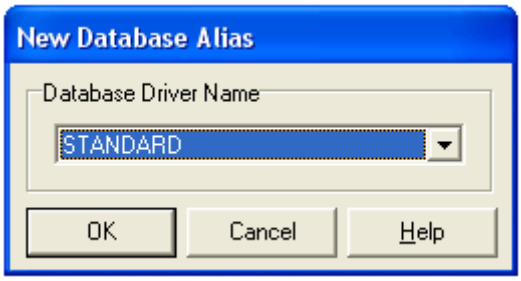
5.2 Borland Database Engine

Navržený program používá pro přístup k databázovým tabulkám nástroj Borland Database Engine (BDE), který musí být nainstalovaný na cílovém počítači. V něm je nutné vytvořit aliasy a nastavit cesty k jednotlivým databázovým tabulkám. Po nainstalování je tedy nutné spustit *bdeadministrator.exe.* Na obrázku 5.2 je zobrazen spuštěný BDE.



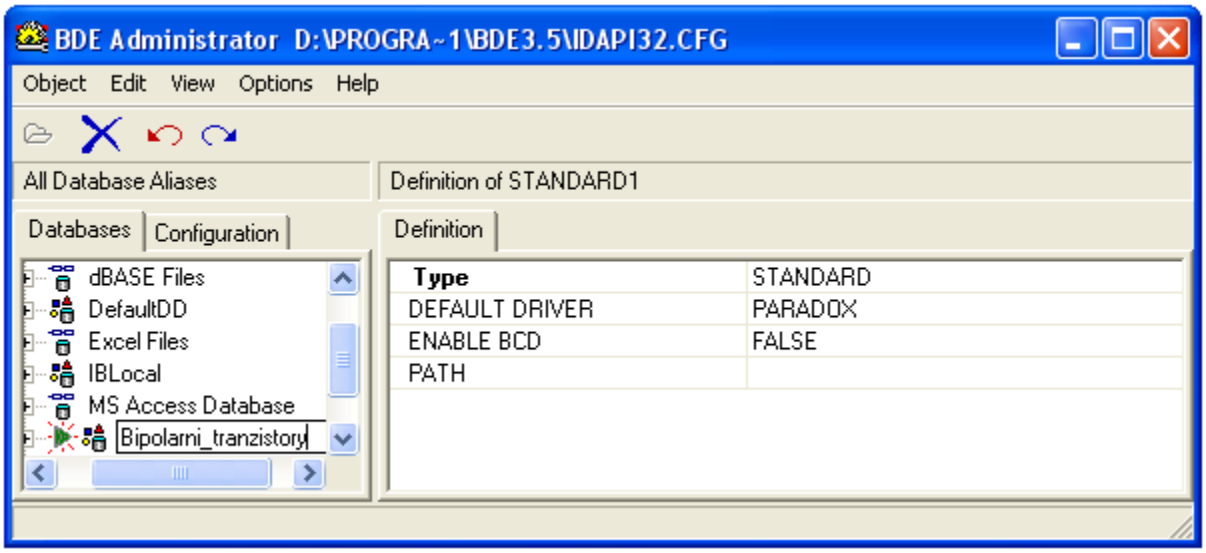
Obr. 5.2: BDE Administrator

Kliknutím pravým tlačítkem myši v pravé části se zobrazí kontextové menu, ve kterém je nutné zvolit *New*...Objeví se nabídka výběru databázového ovladače (Obr. 5.3), postačí nechat typ STANDARD.



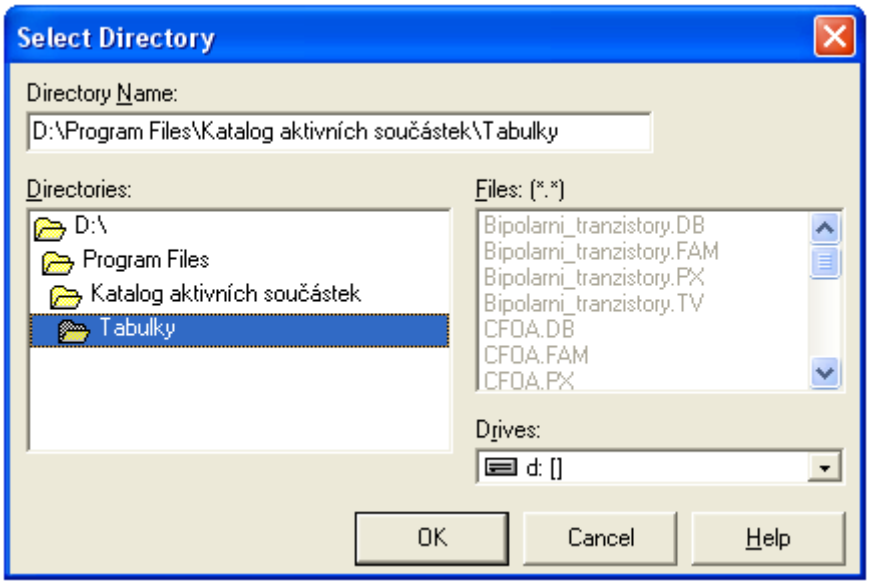
Obr. 5.3: Výběr databázového ovladače

Tímto krokem se vytvořil nový alias. Nyní je nutné ho pojmenovat stejně jako databázovou tabulku, viz. obrázek 5.4.



Obr. 5.4: Vytvoření nového aliasu

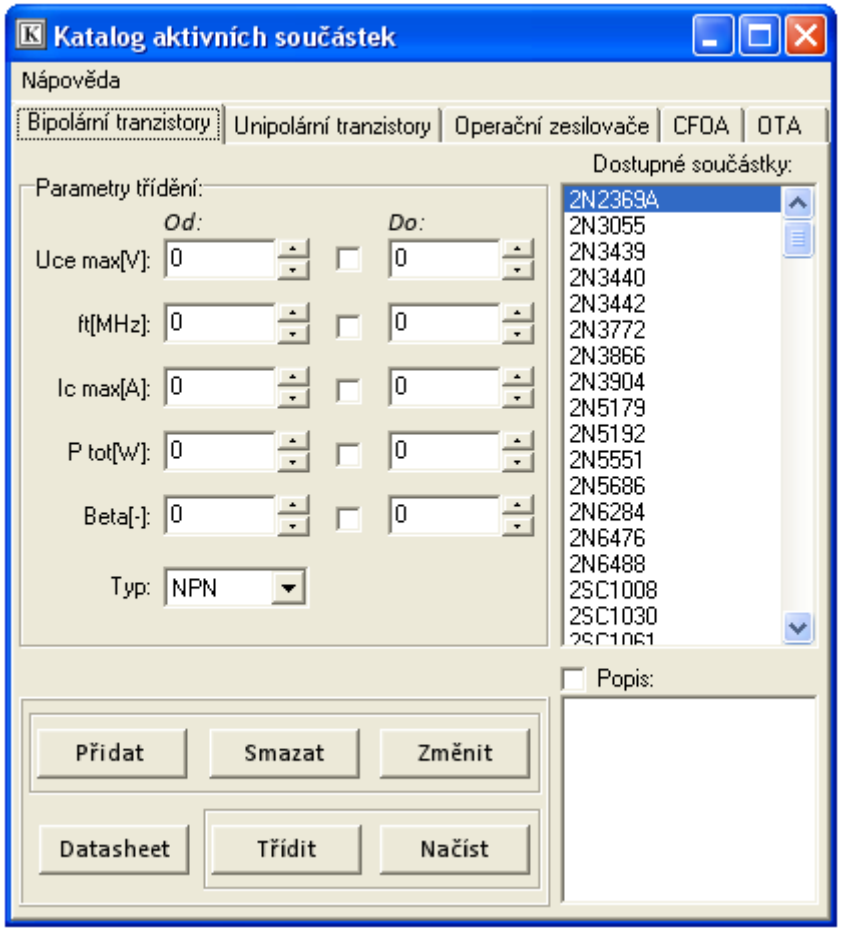
V pravé části BDE Administratoru jsou vypsány informace o vytvořeném ovladači, nyní je nutné nastavit cestku k adresáři, ve kterém je databázová tabulka uložena. Kliknutím do pole PATH se zobrazí okno, v němž se tato cesta nastaví. Toto okno je na obrázku 5.5. Výše popsaným způsobem je nutné vytvořit aliasy ke všem databázovým tabulkám.



Obr. 5.5: Nastavení cesty k databázovým tabulkám

5.3 Popis funkce programu

Na obrázku 5.6 je zobrazen spuštěný program. Do polí *Dostupné součástky* se při spuštění programu načtou všechny součástky obsažené v databázi, které patří do nastavené podkategorie, zde tedy všechny bipolární tranzistory NPN.



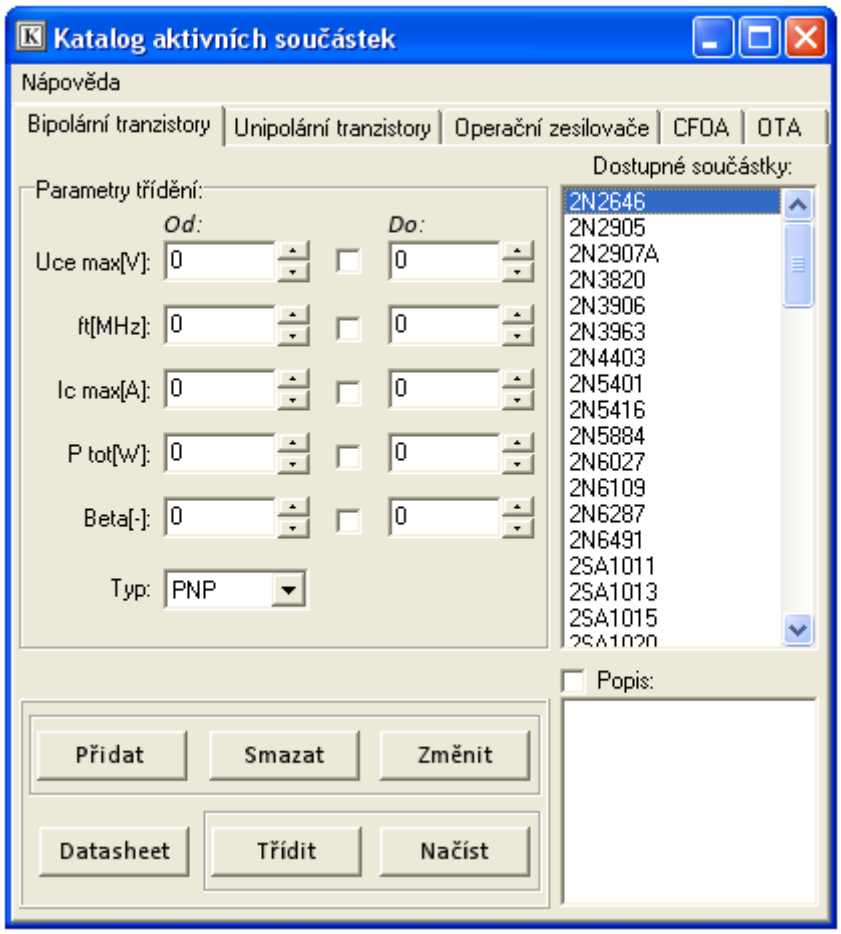
Obr.5.6: Spuštěný program

Změnou pole *Typ:* dojde ke znovunačtení tak, aby zobrazené součástky opět odpovídaly dané podkategorii, viz. obrázek 5.7.

Vložení nové součástky

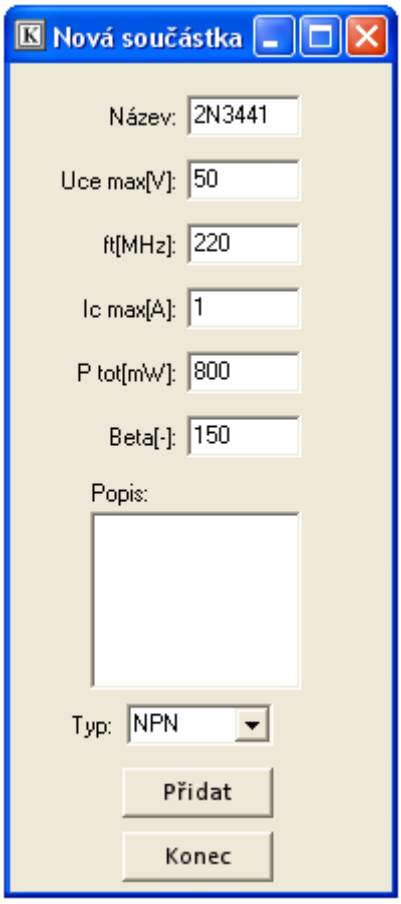
Kliknutím na tlačítko *Přidat* se zobrazí formulář, zobrazený na obrázku 5.8, na kterém se do editačních polí zadá název nové součástky, hodnoty jejich parametrů, volitelně popis součástky a nastaví se její typ.

Po stisknutí tlačítka *Přidat* dojde k vložení součástky a současně se aktualizuje pole *Dostupné součástky*, což je patrné z obrázku 5.11. Stisknutím tlačítka *Konec* se formulář zavře.

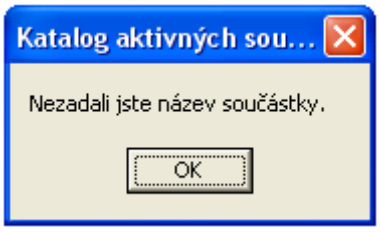


Obr.5.7: Změna podkategorie Typ

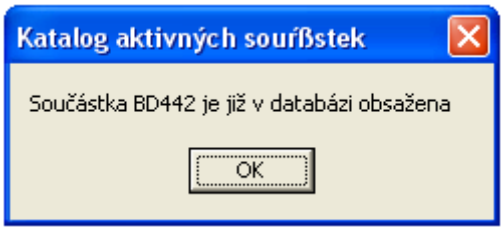
V případě, že uživatel nezadá název součástky, se po stisknutí tlačítka *Přidat* zobrazí okno s upozorněním a dokud nebude název zadán, součástka se neuloží. Obdobně při zadání součástky, která je již v databázi obsažena, je uživatel upozorněn a součástka se opět neuloží. To je patrné z obrázků 5.9 a 5.10. V případě operačních zesilovačů, CFOA a OTA je nutné specifikovat navíc typ napájení a u operačních zesilovačů i kategorii použití. V obou případech je možno vybrat víc možností, pokud není zvolena žádná, uživatel je rovněž upozorněn a součástka se neuloží, viz. obrázky 5.12 až 5.14. Úsek zdrojového kódu, realizující vkládání nových součástek, je přiložen v příloze č. 1.



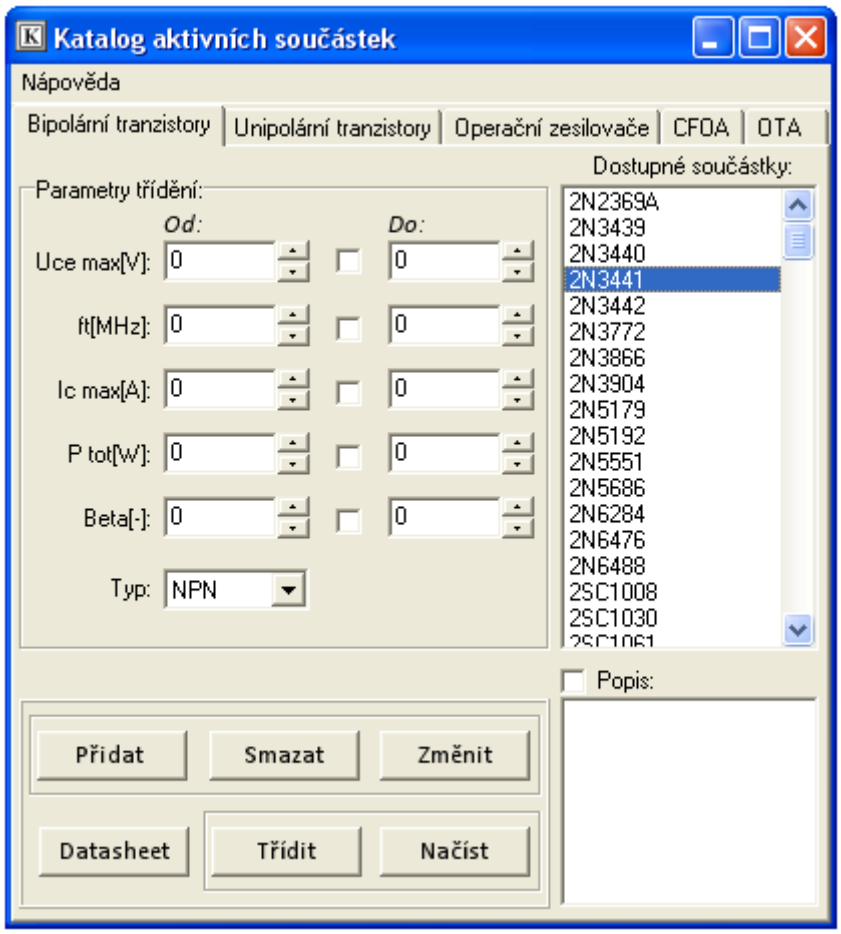
Obr. 5.8: Přidání bipolárního tranzistoru



Obr. 5.9: Upozornění na chybějící název součástky



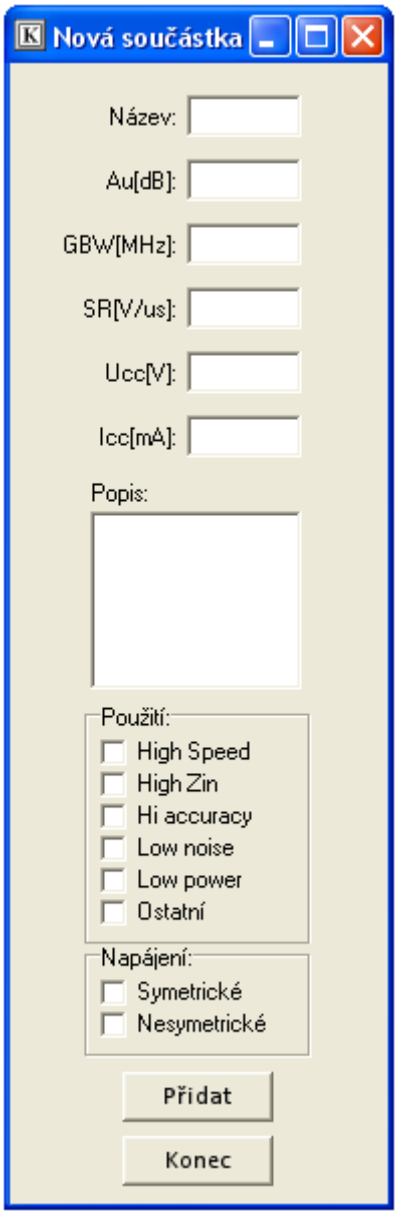
Obr. 5.10: Upozornění na již obsaženou součástku



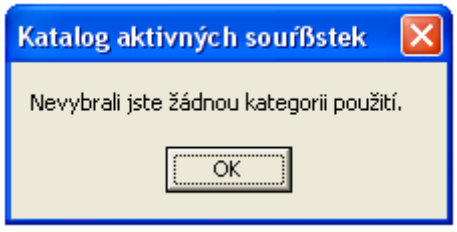
Obr. 5.11: Vložena nová součástka

Smazání součástky

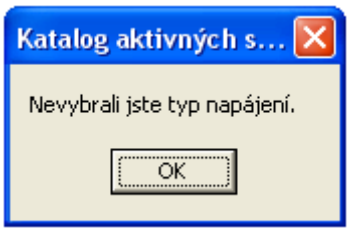
Pro smazání součástky je nutné vybrat danou součástku kliknutím v poli *Dostupné součástky:* a následně kliknout na tlačítko *Smazat*. Před smazáním je uživatel dotázán, chce-li opravdu součástku odstranit. Kliknutím na tlačítko *Yes* se součástka smaže. Mazání demonstrují obrázky 5.15 a 5.16, zdrojový kód v příloze č 2.



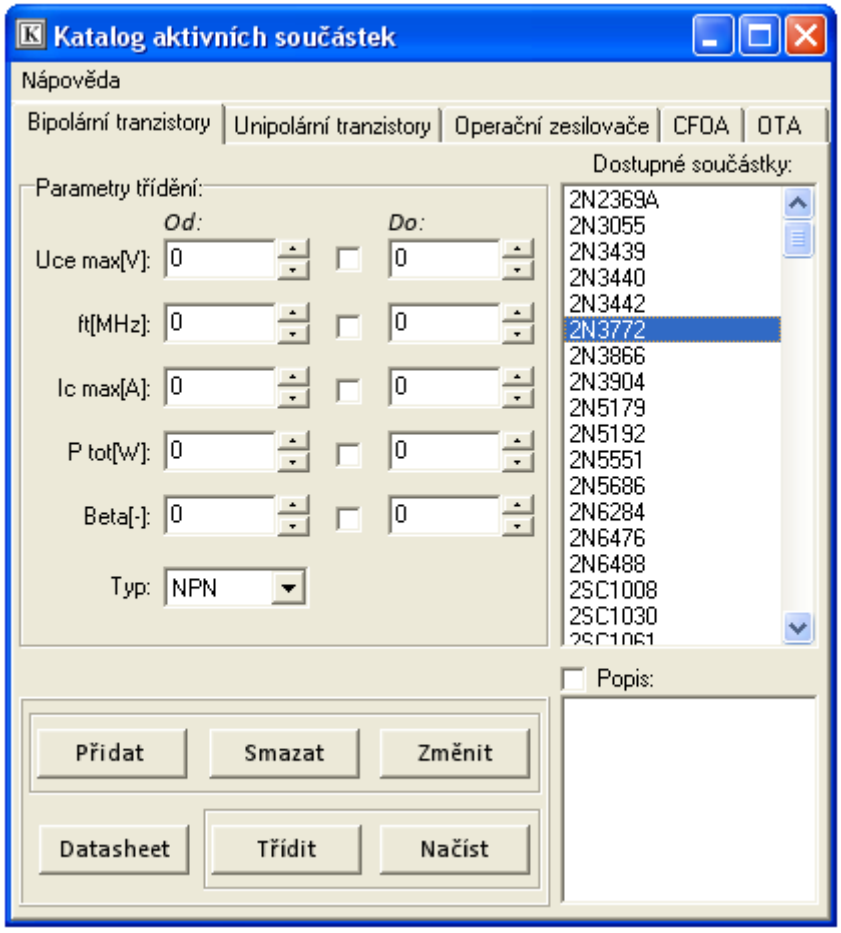
Obr. 5.12: Přidání operačního zesilovače



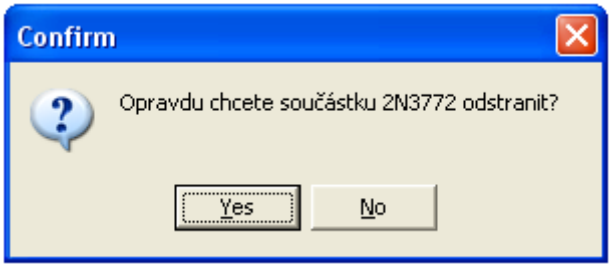
Obr. 5.13: Upozornění na nezadání kategorie použití



Obr. 5.14: Upozornění na nezadání typu napájení



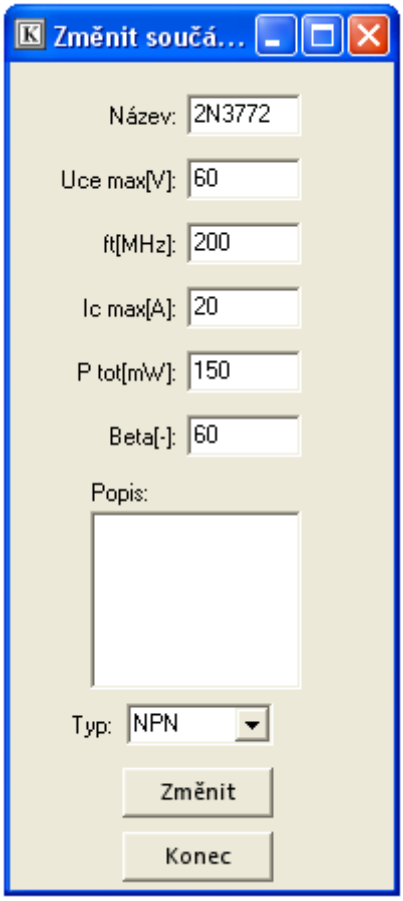
Obr. 5.15: Mazání součástky



Obr. 5.16: Potvrzení smazání součástky

Změna parametrů součástky

Pro změnu parametrů součástky, která je již v databázi, je opět nutné ji označit v poli *Dostupné součástky:* (obr. 5.15). po kliknutí na tlačítko *Změnit* se otevře formulář, v jehož editačních polích jsou již načteny hodnoty jednotlivých parametrů. Kliknutím na tlačítko *Změnit* se hodnoty aktualizují, *Konec* zavře formulář a aktualizaci zruší, viz. obrázek 5.17. Zdrojový kód této části programu je v příloze č. 3.



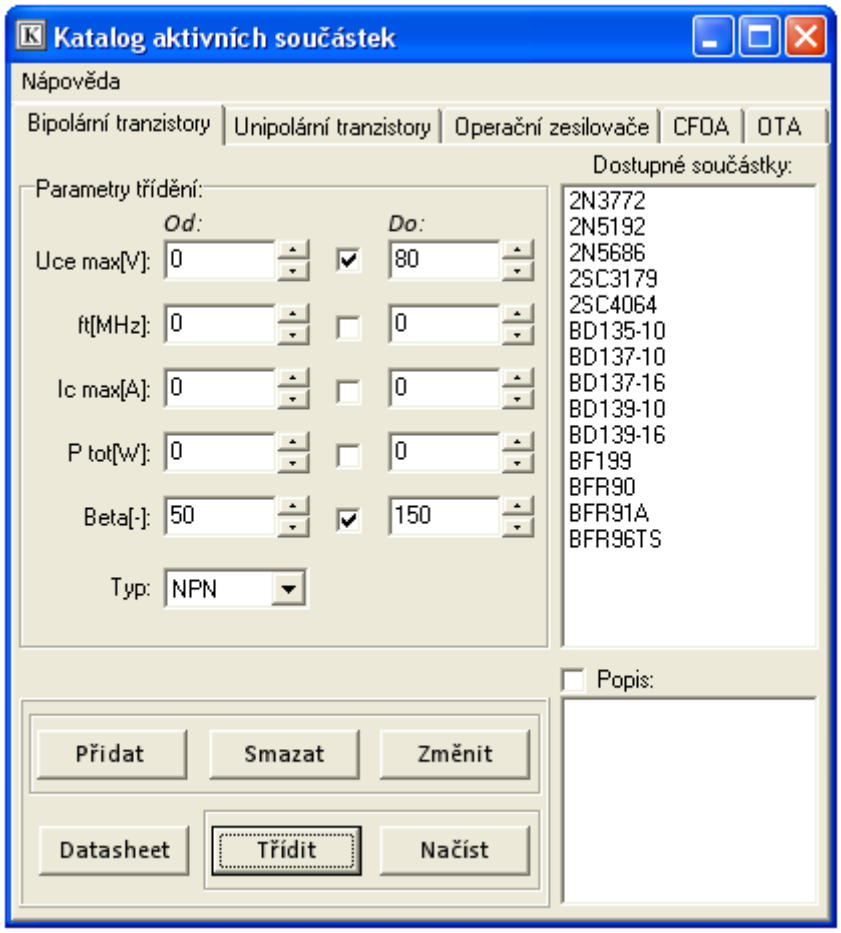
Obr. 5.17: Změna parametrů součástky

Zobrazení datasheetu

Program umožňuje zobrazování datasheetů ve formě pdf souboru. Ve složce, kde je program umístěn, je složka Datasheety, ve které jsou složky Bipolární tranzistory, Unipolární tranzistory, Operační zesilovače, CFOA a OTA. Pokud do nich uživatel umístí datasheety, které budou mít název shodný s názvem součástky, tak po označení součástky v poli *Dostupné součástky:* a kliknutí na tlačítko *Datasheet* dojde ke spuštění Acrobat Readeru a otevření příslušného pdf souboru, viz. příloha č. 4.

Třídění součástek

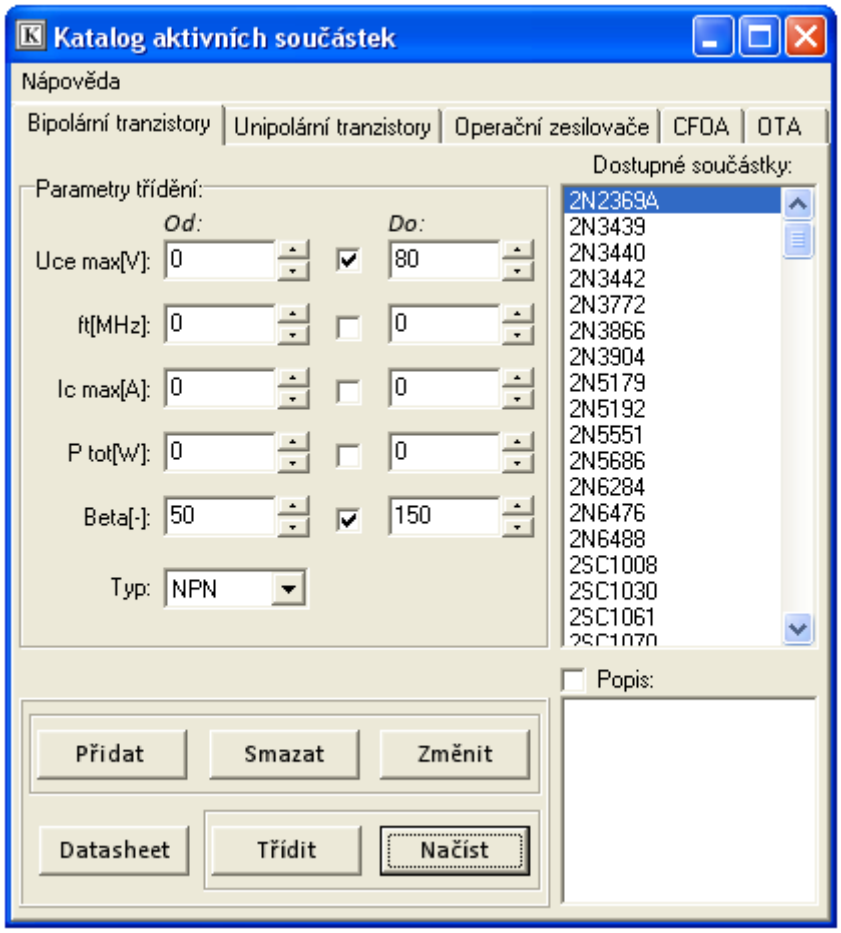
Pokud chce uživatel vypsat pouze ty součástky, jejichž parametry vyhovují požadavkům, pak v editačních polích odpovídajících daným parametrům zadá požadované hodnoty a po stisku tlačítka *Třídit* se tyto součástky zobrazí opět v poli *Dostupné součástky:*. Do třídění se zahrnou pouze hodnoty těch editačních polí, u nichž je zatrženo příslušné zatrhávací pole. Vypsané součástky navíc spadají pouze do podkategorií, které jsou momentálně vybrány, např. u bipolárních tranzistorů *Typ:*. Třídění součástek je zobrazeno na obrázku 5.18, zdrojový kód v příloze č. 5.



Obr. 5.18: Třídění součástek

Načtení součástek

Po vytřídění součástek je možné opětovně načíst do pole *Dostupné součástky:* všechny součástky, které odpovídají dané podkategorii. Toto se provede stisknutím tlačítka *Načíst*, viz. obrázek 5.19, příloha č. 6.



Obr. 5.19: Znovunačtení součástek

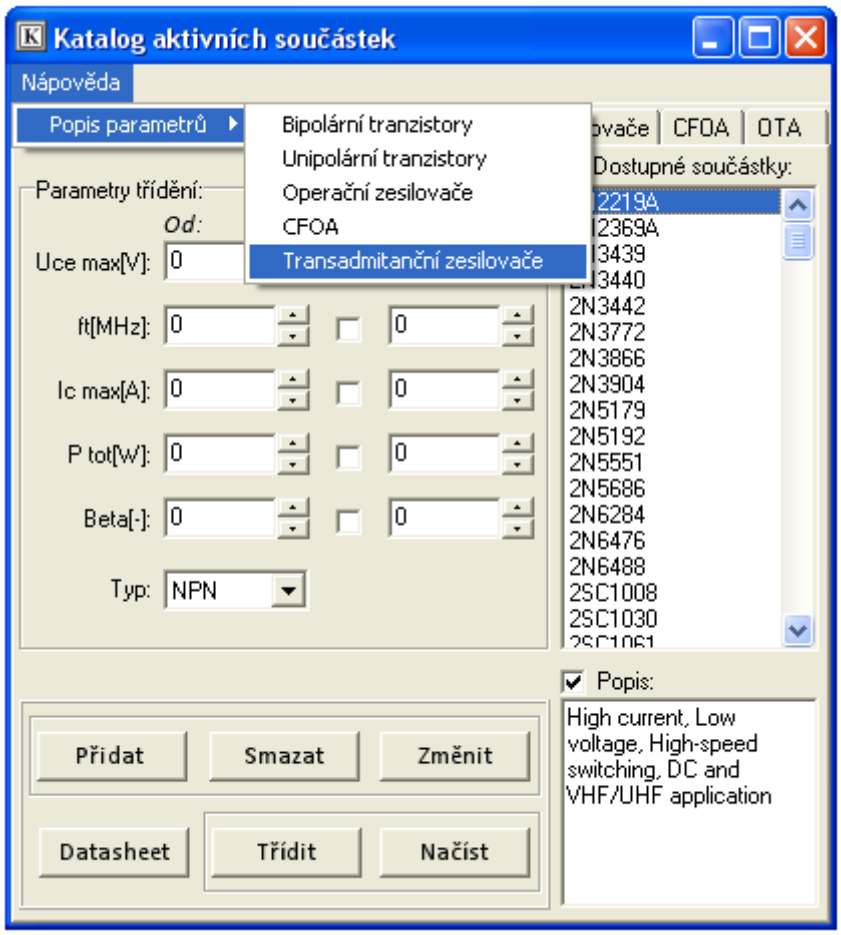
Zobrazení popisu součástky

Pokud uživatel při vložení nové součástky vyplní pole *Popis:*, pak je možné text z tohoto pole zobrazit v poli *Popis:* na hlavním formuláři. Tento text se zobrazí po kliknutí na danou součástku v poli *Dostupné součástky:*, pokud je zatrženo zatrhávací pole *Popis:*, viz. obrázek 5.20. Zdrojový kód pro zobrazení popisu je v příloze č. 7.

Zobrazení nápovědy

Program obsahuje nápovědu ve formě popisu parametrů, podle kterých probíhá třídění. Kliknutím na menu *Nápověda* se zobrazí nabídka *Popis parametrů*, která dále obsahuje nabídky *Bipolární tranzistory, Unipolární tranzistory, Operační zesilovače, CFOA* a *Transadmitanční zesilovače.* Po kliknutí na danou položku se zobrazí nápověda ve formě pdf

souboru. Struktura tohoto menu je zobrazena rovněž na obrázku 5.20, zdrojový kód je přiložen v příloze č. 8.



Obr. 5.20: Zobrazení popisu součástky a nápovědy

6. Závěr

Cílem tohoto projektu bylo vytvořit elektronickou databázi aktivních součástek, která bude umožňovat vkládat a mazat součástky, editovat jejich parametry a zejména vyhledávat součástky podle jejich nejdůležitějších parametrů. Požadavkem byla rovněž přehlednost a snadné intuitivní ovládání programu.

Vytvořený program umožňuje přidávat nové součástky, mazat součástky, editovat jejich parametry a třídit je podle jejich nejdůležitějších parametrů. Dále pak zobrazovat jejich popis a zobrazovat datasheety. V programu je rovněž obsažena nápověda k parametrům vybraným pro třídění. Cíle dané zadáním se tedy podařilo splnit.

Hlavní přínos tohoto programu spočívá v jednoduchém a rychlém vyhledávání obsažených typů součástek bez nutnosti navštěvovat webové stránky jednotlivých výrobců těchto součástek.

7. Použitá literatura

[1] LIBERTY, Jesse. *Naučte se C++ za 21 dní*. Brno: Computer Press, 2007. 796 s. ISBN 978-80-251-1583-1.

[2] KADLEC, Václav. *Učíme se programovat v Borland C++ Builder a jazyce C++*. Brno : Computer Press, 2004. 385 s. ISBN 80-7226-550-4.

[3] *Interval.cz* [online]. 2000 [cit. 2009-04-17]. Dostupný z WWW: <http://interval.cz/clanky/databaze-a-jazyk-sql/>.

[4] *C++ Builder virtuální učebnice* [online]. 2004 [cit. 2009-04-19]. Dostupný z WWW: <http://www.sfiles.host.sk/builder/>.

[5] *Analog Devices,Inc* [online]. 1995 [cit. 2009-05-08]. Dostupný z WWW: <www.analog.com>.

[6] *Texas Instruments* [online]. 1995 [cit. 2009-05-10]. Dostupný z WWW: <www.ti.com>.

[7] *GM Electronic* [online]. 1998 [cit. 2009-05-03]. Dostupný z WWW: <www.gme.cz>.

[8] *Linuxsoft.cz* [online]. 2003 [cit. 2009-05-16]. Dostupný z WWW: <www.linuxsoft.cz>.

8. Seznam příloh

Přílohy

1. Zdrojový kód: vložení nové součástky

```
//---------------------------------------------------------------------------

void __fastcall TfrmPridejBT::btnPridejBTClick(TObject *Sender)
{
AnsiString Typ = Form1 -> ComboBox1BT -> Items -> Strings[Form1 -> ComboBox1BT ->
ItemIndex];
AnsiString TypBT = ComboBox1PridejBT -> Items -> Strings[ComboBox1PridejBT ->
ItemIndex];
AnsiString a,b,c,d,e,f,g,h,Nazev;
AnsiString edity[] = {Edit2BT -> Text, Edit3BT -> Text, Edit4BT -> Text, Edit5BT
-> Text, Edit6BT -> Text};
AnsiString promenne[] = {b,c,d,e,f};

a = Edit1BT -> Text;
b = Edit2BT -> Text;
c = Edit3BT -> Text;
d = Edit4BT -> Text;
e = Edit5BT -> Text;
f = Edit6BT -> Text;
g = Memo1BT -> Text;

for (int i=0; i<5; i++)
 {
 if(edity[i] == "")
   {
   promenne[i] = -10;
   }
 else
   promenne[i] = edity[i];
 }

if(Edit1BT -> Text == "")
 ShowMessage("Nezadali jste název součástky.");
else
{
Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+a+"' ");
Query1 -> Open();

Nazev = DataSource1->DataSet->FieldByName("Nazev_soucastky")->AsString;

if(a == Nazev)
{
ShowMessage("Součástka "+Nazev+" je již v databázi obsažena");
}
else
{
Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("INSERT INTO Bipolarni_tranzistory (Nazev_soucastky,
Max_napeti_CE, Tranzitni_kmitocet, Max_proud_kolektorem, Max_vykonova_ztrata,
Proudovy_zes_cinitel, Popis, Polarita) VALUES ('"+a+"', "+promenne[0]+",
"+promenne[1]+", "+promenne[2]+", "+promenne[3]+", "+promenne[4]+",'"+g+"',
'"+TypBT+"')");
Query1 -> ExecSQL();

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Polarita = '"+Typ+"'");
Query1 -> Open();

Form1->DBListBox1->Items->Clear();
Form1->DataSource1->DataSet->FindFirst();
do
  {
Form1->DBListBox1->Items->Add(Form1->DataSource1->DataSet-
>FieldByName("Nazev_soucastky")->AsString);
Form1->DataSource1->DataSet->Next();
  }
while (!Form1->DataSource1->DataSet->Eof);
```

```
  }
  }
Form1 -> Memo1 -> Clear();
}
//---------------------------------------------------------------------------

void __fastcall TfrmPridejBT::FormShow(TObject *Sender)
{
Memo1BT -> Clear();
}
//---------------------------------------------------------------------------
```

2. Zdrojový kód: smazání součástky

```
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSmazBTClick(TObject *Sender)
{
AnsiString smazatBT;
AnsiString TypBT = ComboBox1BT -> Items -> Strings[ComboBox1BT -> ItemIndex];

if(DBListBox1 -> ItemIndex!= -1)

smazatBT = DBListBox1 -> Items -> Strings[DBListBox1 -> ItemIndex];

else return;

if(MessageDlg(AnsiString ("Opravdu chcete součástku "+smazatBT+" odstranit?"),
mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0) == mrYes)
{
DataSource1 -> DataSet = Query1;
Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("DELETE FROM Bipolarni_tranzistory WHERE Nazev_soucastky =
'"+smazatBT+"'");
Query1 -> ExecSQL();
}
Memo1 -> Clear();

DataSource1 -> DataSet = Query1;

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Polarita = '"+TypBT+"'");
Query1 -> Open();

DBListBox1->Items->Clear();
DataSource1->DataSet->FindFirst();
do
  {
DBListBox1->Items->Add(DataSource1->DataSet->FieldByName("Nazev_soucastky")-
>AsString);
DataSource1->DataSet->Next();
  }
while (!DataSource1->DataSet->Eof);

DataSource1 -> DataSet = Table1;
}
//---------------------------------------------------------------------------
```

3. Zdrojový kód: změna parametrů součástky

```
//---------------------------------------------------------------------------
void __fastcall TfrmZmenitBT::FormShow(TObject *Sender)
{
Memo1BT -> Clear();

AnsiString pomocna = -10;
AnsiString zmenitBT;
if(Form1 -> DBListBox1 -> ItemIndex!= -1)

zmenitBT = Form1 -> DBListBox1 -> Items -> Strings[Form1 -> DBListBox1 ->
ItemIndex];

else return;

DataSource1 -> DataSet = Query1;

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit1BT -> Text = DataSource1->DataSet->FieldByName("Nazev_soucastky")->AsString;
DataSource1->DataSet->Next();


Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Max_napeti_CE FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit2BT -> Text = DataSource1->DataSet->FieldByName("Max_napeti_CE")-> AsString;

if( Edit2BT -> Text == pomocna)
{
Edit2BT -> Text = "";
}

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Tranzitni_kmitocet FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit3BT -> Text = DataSource1->DataSet->FieldByName("Tranzitni_kmitocet")-
>AsString;

if( Edit3BT -> Text == pomocna)
{
Edit3BT -> Text = "";
}

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Max_proud_kolektorem FROM Bipolarni_tranzistory
WHERE Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit4BT -> Text = DataSource1->DataSet->FieldByName("Max_proud_kolektorem")-
>AsString;

if( Edit4BT -> Text == pomocna)
{
Edit4BT -> Text = "";
}

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Max_vykonova_ztrata FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit5BT -> Text = DataSource1->DataSet->FieldByName("Max_vykonova_ztrata")-
>AsString;

if( Edit5BT -> Text == pomocna)
{
```

```
Edit5BT -> Text = "";
}

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Proudovy_zes_cinitel FROM Bipolarni_tranzistory
WHERE Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Edit6BT -> Text = DataSource1->DataSet->FieldByName("Proudovy_zes_cinitel")-
>AsString;

if( Edit6BT -> Text == pomocna)
{
Edit6BT -> Text = "";
}

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Popis FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

Memo1BT -> Text = DataSource1->DataSet->FieldByName("Popis")->AsString;

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Polarita FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+zmenitBT+"'");
Query1 -> Open();

}
//---------------------------------------------------------------------------

void __fastcall TfrmZmenitBT::btnZmenBTClick(TObject *Sender)
{
AnsiString a,b,c,d,e,f,g,TypBT;
AnsiString edity[] = {Edit2BT -> Text, Edit3BT -> Text, Edit4BT -> Text, Edit5BT
-> Text, Edit6BT -> Text};
AnsiString promenne[] = {b,c,d,e,f};

a = Edit1BT -> Text;
b = Edit2BT -> Text;
c = Edit3BT -> Text;
d = Edit4BT -> Text;
e = Edit5BT -> Text;
f = Edit6BT -> Text;
g = Memo1BT -> Text;

for (int i=0; i<5; i++)
 {
 if(edity[i] == "")
   {
   promenne[i] = -10;
   }
 else
   promenne[i] = edity[i];
 }

TypBT = ComboBox1ZmenBT -> Items -> Strings[ComboBox1ZmenBT -> ItemIndex];

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("UPDATE Bipolarni_tranzistory SET Nazev_soucastky = '"+a+"',
Max_napeti_CE = "+promenne[0]+", Tranzitni_kmitocet = "+promenne[1]+",
Max_proud_kolektorem = "+promenne[2]+", Max_vykonova_ztrata = "+promenne[3]+",
Proudovy_zes_cinitel = "+promenne[4]+", Popis = '"+g+"',Polarita = '"+TypBT+"'
WHERE Nazev_soucastky = '"+a+"'");
Query1 -> ExecSQL();
}
//---------------------------------------------------------------------------
```

4. Zdrojový kód: zobrazení datasheetu

```
//---------------------------------------------------------------------------

void __fastcall TForm1::btnZobrazPdfBTClick(TObject *Sender)
{
AnsiString zobrazitBT;
if(DBListBox1 -> ItemIndex!= -1)

zobrazitBT = DBListBox1 -> Items -> Strings[DBListBox1 -> ItemIndex];

else return;
AnsiString Pdf = ExtractFilePath(Application->ExeName)+"/Datasheety/Bipolární
tranzistory/"+zobrazitBT+".pdf";
ShellExecute(NULL,"open",Pdf.c_str(),NULL,NULL,SW_SHOWNORMAL);
}
//---------------------------------------------------------------------------
```

5. Zdrojový kód: třídění součástek

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnTriditBTClick(TObject *Sender)
{
Memo1 -> Clear();
AnsiString BTa,BTb,BTc,BTd,BTe,BTf,BTg,BTh,BTi,BTj;

BTa = Edit1BT -> Text;
BTb = Edit2BT -> Text;
BTc = Edit3BT -> Text;
BTd = Edit4BT -> Text;
BTe = Edit5BT -> Text;
BTf = Edit6BT -> Text;
BTg = Edit7BT -> Text;
BTh = Edit8BT -> Text;
BTi = Edit9BT -> Text;
BTj = Edit10BT -> Text;

AnsiString TypBT = ComboBox1BT -> Items -> Strings[ComboBox1BT -> ItemIndex];
AnsiString dotaz = "SELECT Nazev_soucastky FROM Bipolarni_tranzistory";
AnsiString texty[] = {"Max_napeti_CE >= "+BTa+" AND Max_napeti_CE <=
"+BTb,"Tranzitni_kmitocet >= "+BTc+" AND Tranzitni_kmitocet <= "+BTd,
"Max_proud_kolektorem >= "+BTe+" AND Max_proud_kolektorem <= "+BTf,
"Max_vykonova_ztrata >= "+BTg+" AND Max_vykonova_ztrata <= "+BTh,
"Proudovy_zes_cinitel >= "+BTi+" AND Proudovy_zes_cinitel <= "+BTj};
AnsiString pomocna_prom = {" AND Polarita = '"+TypBT+"'" };
bool checkboxy[] = {CheckBox1BT->Checked, CheckBox2BT->Checked, CheckBox3BT-
>Checked, CheckBox4BT->Checked, CheckBox5BT->Checked  };



int p = 0;
for (int i = 0; i < 5; i++)
 {
if (checkboxy[i])
   {
if (p==0)
     {
     dotaz += " WHERE ";
     }
   if (p >= 1)
      {
       dotaz += " AND ";
      }
    dotaz += texty[i];
    p++;
    }
  }
dotaz = dotaz + pomocna_prom;

DataSource1 -> DataSet = Query1;

if (p > 0)
{
Query1 -> SQL -> Clear();
Query1 -> SQL -> Add(dotaz);
Query1 -> Open();
}

if (p==0)
{
Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Max_napeti_CE = -99");
Query1 -> Open();
}

DBListBox1->Items->Clear();
DataSource1->DataSet->FindFirst();
do
     {
DBListBox1->Items->Add(DataSource1->DataSet->FieldByName("Nazev_soucastky")-
>AsString);
```

```
DataSource1->DataSet->Next();
      }
while (!DataSource1->DataSet->Eof);

DataSource1 -> DataSet = Table1;
}
//---------------------------------------------------------------------------
```

6. Zdrojový kód: načtení součástek

```
//---------------------------------------------------------------------------

void __fastcall TForm1::btnNactiBTClick(TObject *Sender)
{
Memo1 -> Clear();

AnsiString TypBT = ComboBox1BT -> Items -> Strings[ComboBox1BT -> ItemIndex];

DataSource1 -> DataSet = Query1;

Query1 -> SQL -> Clear();
Query1 -> SQL -> Add("SELECT Nazev_soucastky FROM Bipolarni_tranzistory WHERE
Polarita = '"+TypBT+"'");
Query1 -> Open();

DBListBox1->Items->Clear();
DataSource1->DataSet->FindFirst();
do
  {
DBListBox1->Items->Add(DataSource1->DataSet->FieldByName("Nazev_soucastky")-
>AsString);
DataSource1->DataSet->Next();
  }
while (!DataSource1->DataSet->Eof);

DataSource1 -> DataSet = Table1;
}
//---------------------------------------------------------------------------
```

7. Zdrojový kód: zobrazení popisu součástky

```
//---------------------------------------------------------------------------

void __fastcall TForm1::DBListBox1Click(TObject *Sender)
{
AnsiString popisBT;

if (CheckBoxPopisBT->Checked)

if(DBListBox1 -> ItemIndex!= -1)

popisBT = DBListBox1 -> Items -> Strings[DBListBox1 -> ItemIndex];

else return;

Query6 -> SQL -> Clear();
Query6 -> SQL -> Add("SELECT Popis FROM Bipolarni_tranzistory WHERE
Nazev_soucastky = '"+popisBT+"'");
Query6 -> Open();

Memo1 -> Lines -> Clear();
Memo1 -> Lines -> Add(DataSource6->DataSet->FieldByName("Popis")->AsString);
}
//---------------------------------------------------------------------------
```

8. Zdrojový kód: zobrazení nápovědy

```
//---------------------------------------------------------------------------

void __fastcall TForm1::BT_napovedaClick(TObject *Sender)
{
AnsiString Pdf = ExtractFilePath(Application->ExeName)+"/Help/BT.pdf";
ShellExecute(NULL,"open",Pdf.c_str(),NULL,NULL,SW_SHOWNORMAL);
}
//---------------------------------------------------------------------------
```