

**ZÁPADOČESKÁ UNIVERZITA V PLZNI  
FAKULTA ELEKTROTECHNICKÁ**

**Katedra elektromechaniky a výkonové elektroniky**

**BAKALÁŘSKÁ PRÁCE**

**Freeware simulační a vizualizační nástroje pro GNU / Linux**

**vedoucí práce: Ing. Martin Janda, Ph.D.  
autor: Lupínek Jiří**

**2012**

ZAPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2011/2012

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jiří LUPÍNEK**  
Osobní číslo: **E08B0050K**  
Studijní program: **B2644 Aplikovaná elektrotechnika**  
Studijní obor: **Aplikovaná elektrotechnika**  
Název tématu: **Freeware simulační a vizualizační nástroje pro GNU/Linux**  
Zadávací katedra: **Katedra elektromechaniky a výkonové elektroniky**

### Z á s a d y p r o v y p r a c o v á n í :

1. Vyhledání volně šiřitelných simulačních programů pro elektrotechnické úlohy.
2. Vyhledání volně šiřitelných programů pro zobrazení výsledků simulací.
3. Demonstrace použitelnosti nasimulováním jednoduchých elektrotechnických úloh.
4. Porovnání programů - uživatelský komfort, snadnost instalace...

Cílem práce je vytvořit přehled freeware simulačních a vizualizačních programů vhodných pro simulace elektrotechnických úloh pro začátečnické distribuce OS GNU/Linux (Ubuntu, Mandrake...) včetně popisu jejich instalace.

ZAPADOČESKÁ UNIVERZITA V PLZNI  
Fakulta elektrotechnická  
Akademický rok: 2011/2012

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Jiří LUPÍNEK**  
Osobní číslo: **E08B0050K**  
Studijní program: **B2644 Aplikovaná elektrotechnika**  
Studijní obor: **Aplikovaná elektrotechnika**  
Název tématu: **Freeware simulační a vizualizační nástroje pro GNU/Linux**  
Zadávající katedra: **Katedra elektromechaniky a výkonové elektroniky**

### Z á s a d y p r o v y p r a c o v á n í :

1. Vyhledání volně šiřitelných simulačních programů pro elektrotechnické úlohy.
2. Vyhledání volně šiřitelných programů pro zobrazení výsledků simulací.
3. Demonstrace použitelnosti nasimulováním jednoduchých elektrotechnických úloh.
4. Porovnání programů - uživatelský komfort, snadnost instalace...

Cílem práce je vytvořit přehled freeware simulačních a vizualizačních programů vhodných pro simulace elektrotechnických úloh pro začátečnické distribuce OS GNU/Linux (Ubuntu, Mandrake...) včetně popisu jejich instalace.

## **Anotace**

Diplomová práce je zaměřena na porovnávání freeware simulačních a vizualizačních nástrojů pro operační systém Linux. Práce se zabývá vyhledáním volně šiřitelných programů pro simulace elektronických úloh a jejich zobrazením. Dále také demonstraci použití nasimulováním jednoduché elektrotechnické úlohy a následným porovnáním z hlediska uživatele.

## **Klíčová slova**

Linux, elektrotechnická úloha, simulace, instalace, Pascal, C++, freeware, simulační nástroje, porovnání

## **Abstract**

The graduation thesis is about comparing of freeware simulation and visualization programs for operation system Linux (Ubuntu, Mandrake...). It's about searching for freeware programs for simulation of electrotechnical tasks and their visualization. It also include a use by demonstration on simple electrotechnical task and comparison from the view of the user.

## **Key words**

Linux, electrotechnical task, simulation, instalation, Pascal, C++, freeware, simulation tools, comparizon

## **Prohlášení**

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr bakalářského studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím literatury a pramenů uvedených v seznamu, který je součástí této práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské/diplomové práce, je legální.

## Obsah

ÚVOD.....	8
SEZNAM SYMBOLŮ A ZKRATEK.....	9
<b>1. LINUX.....</b>	<b>10</b>
1.1 ÚVOD DO PROGRAMOVÁNÍ.....	10
1.2 POPIS LINUXU.....	10
1.3 NEJZNÁMĚJŠÍ DISTIBUCE.....	11
1.4 INSTALACE LINUXU.....	11
1.5 VZHLED OPERAČNÍHO SYSTÉMU.....	12
<b>2. PROGRAMOVACÍ JAZYKY.....</b>	<b>14</b>
2.1 INSTALACE.....	14
2.2 LAZARUS.....	17
2.2.1 Základní informace.....	17
2.2.2 Používání databází.....	17
2.2.3 Porovnání s Delphi.....	18
2.3 QT4.....	21
2.3.1. Popis.....	21
2.3.2 Vývoj.....	21
2.3.3 Podpora programů.....	22
<b>3. REGULACE PROUDU V KOTVE DC MOTORU.....</b>	<b>22</b>
3.1 MATLAB.....	22
3.2 LAZARUS.....	22
<b>ZÁVĚR.....</b>	<b>24</b>
<b>SEZNAM LITERATURY A INFORMAČNÍCH ZDROJŮ.....</b>	<b>25</b>
<b>PŘÍLOHY.....</b>	<b>26</b>
<i>Příloha 1 – Původní program v Matlabu.....</i>	<i>26</i>
<i>Příloha 2 – Program předělaný pro Pascal v programu Lazarus.....</i>	<i>28</i>
<i>Příloha 3 – Hlavní struktura v programu C++.....</i>	<i>32</i>

## Úvod

Tato práce je zaměřena na porovnání freeware grafických a vizualizačních programů pro Linux.

Samotná práce má tři části. První popisuje použitý operační systém, jeho první instalaci a rady důležité pro práci s ním. Druhá se zabývá programy použitými pro vytvoření kódů pro hlavní program, jejich popisem, reálným využitím a instalací. Třetí část popisuje funkci programů a vizualizaci.



## Seznam symbolů a zkratk

Linux .....	operační systém
C++.....	programovací jazyk
Pascal.....	programovací jazyk
OS.....	operační systém
PDA.....	malý kapesní počítač
MAC.....	operační systém pro počítače Macintosh
MATLAB .....	Matrix laboratory

# 1. Linux

## 1.1 Úvod do programování

Freeware programy se rozvíjejí již od počátku výroby složitějších programů. Jedná se většinou o zjednodušené verze placených programů, které mají za úkol navnadit budoucí zákazníky a donutit je ke koupi vylepšené verze. V praxi se tento typ programů začal používat u operačních systémů Windows, ale postupně se přesunul i na ostatní platformy.

S růstem poptávky po specializovaných programech pro specifická odvětví lidské činnosti se objevili i programy pro řešení nejdříve jednodušších elektrotechnických úloh a později i pro ty složitější. Nejednalo se jen o samotné výpočty, ale postupem času i o vizuální simulaci. Díky zdokonalování počítačových komponentů rostou i možnosti detailního zobrazení.

Běžní uživatelé ovšem nezvládají složitější a dle nich nelogické programy. Proto se začal brát ohled nejen na vzhled a zobrazení hodnot a výsledků, ale také na jednoduchost ovládání. Ruku v ruce s ovládáním se rozvinula i snadná instalace.

V této práci budeme pracovat a porovnávat programovací jazyky Pascal a C++. Oba mají možnost vizualizace a hodí se k naprogramování zadaných úloh dle potřeb uživatele. V našem případě byl použit freeware program Lazarus IDE v0.9.28.2-10 pro zpracování jazyka Pascal a program QT4 pro jazyk C++ .

## 1.2 Popis Linuxu

Linux je moderní operační systém (OS), jehož ovládání je stejně přívětivé jako u jiných systémů a který obsahuje velké množství ovladačů pro nejrůznější standardizovaný hardware.

- Na Linuxu není zpravidla nutné používat žádný komplikovaný zabezpečovací nebo antivirový systém.

- Linux neodesílá žádné citlivé informace bez vašeho vědomí. V naprosté většině případů systém nevyžaduje restart k projevení změn.
- Mezi aplikacemi každé větší linuxové distribuce najdete kancelářský balík, tzv. office.
- V každé linuxové distribuci naleznete velké množství nejrůznějších aplikací.
- Linux lze výkonově i vzhledově přizpůsobit každému požadavku.
- Linux je možné používat na velkém množství zařízení od PDA, přes notebooky, stolní počítače až po specializované servery. Díky jeho založení však není potřeba se na každém učit novému zacházení. Stále se jedná o principiálně stejný systém.
- Flexibilita Linuxu umožňuje, že při změně hardwaru nebo celého počítače nemusíte přeinstalovávat celý systém.
- Nemusíte se přihlašovat a odhlašovat při administraci systému. Pracovat může i několik lidí najednou.
- Na Linuxu je možné pracovat vzdáleně, a to mnoha různými způsoby.
- V podstatě všechny linuxové distribuce jsou k dispozici zdarma, protože se jedná o svobodný software.

### 1.3 Nejznámější distribuce

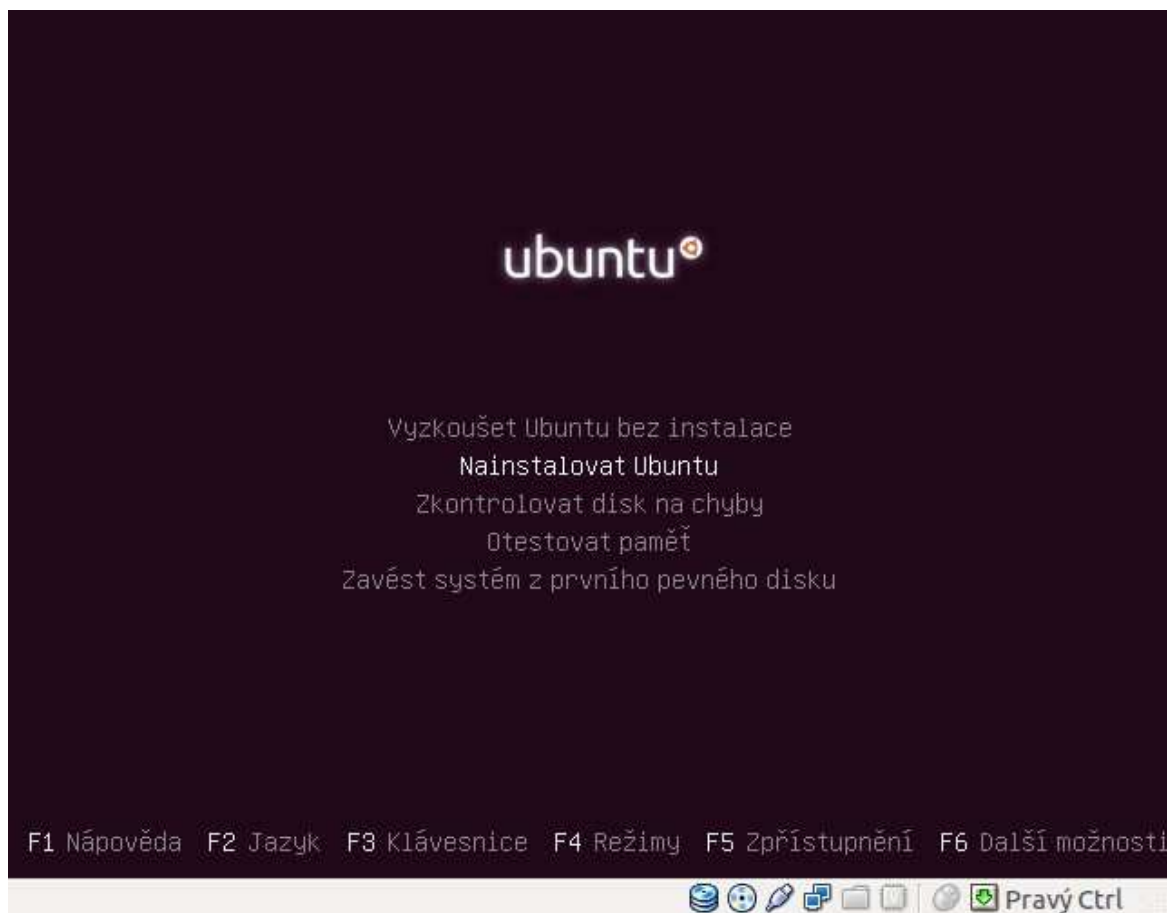
- Ubuntu – celkem mladá distribuce, vývoj pro osobní počítače jako desktopová aplikace
  - různé modifikace grafických prostředí, možnost pustění přes Live CD ( lze spustit Linux bez nutnosti instalace)
- Gentoo – pro pokročilé uživatele, možnost nastavit si distribuci na míru počítače a používání
- Debian – rozsáhlý, dlouho vyvíjený

### 1.4 Instalace Linuxu

Z internetu si stáhneme instalační soubory, například ze stránek <http://www.linux.cz/> . Po vypálení na cd / Dvd v režimu systémového disku není problém nainstalovat systém na počítač.

V našem případě jsem použil program Oracle Virtual Box, který nám umožňuje vytvořit si virtuální počítač v počítači s již nainstalovaným operačním systémem Windows. Odpadne tímto problém s rozdělováním pevných disků na více oddílů a nastavování spouštění jednotlivých operačních systémů.

Instalace Linuxu se příliš neliší od většině lidí známé instalace Windows. Poběhne prakticky bez zásahu uživatele a zabere necelou půlhodinu času.

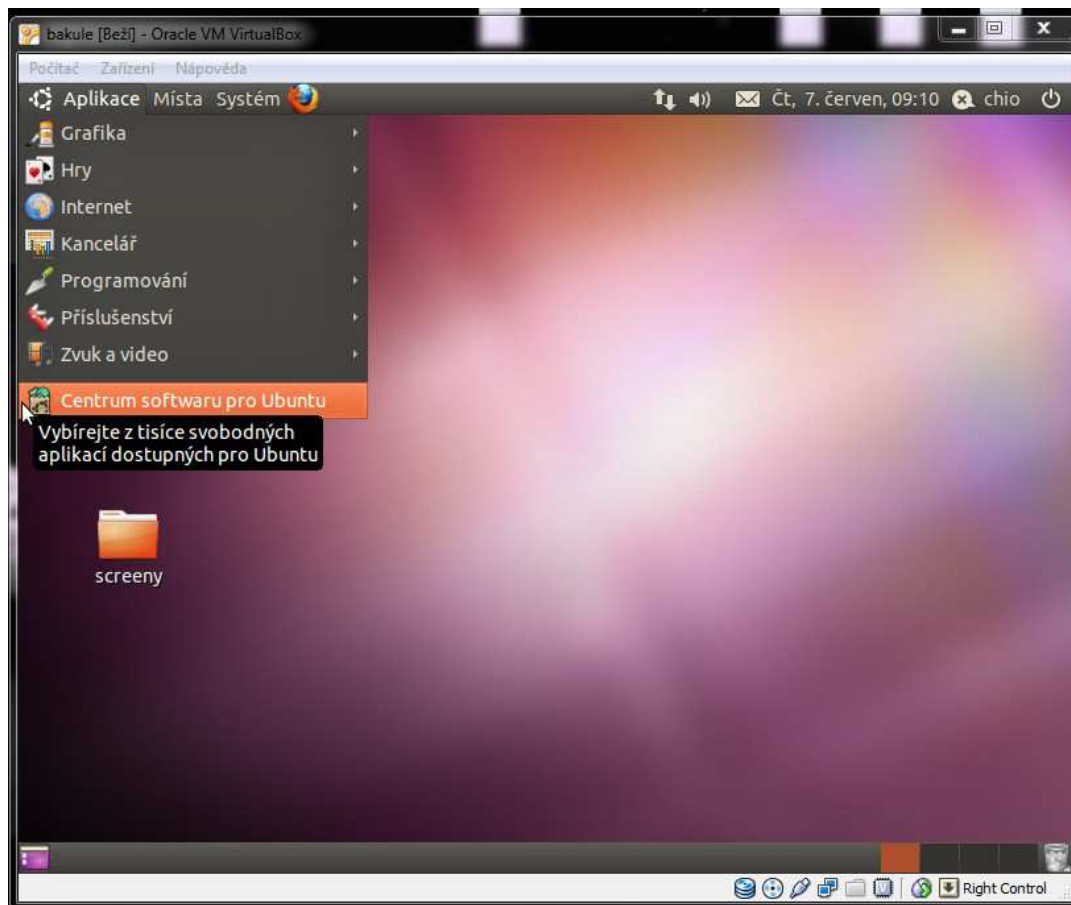


Obrázek 1.4 – Instalace Ubuntu

## 1.5 Vzhled operačního systému

Práce v systému je v případě distribuce Ubuntu prakticky totožná s Windows. Najdeme zde plochu, systémovou lištu monitorující pevné i připojené disky, nastavení obrazovek či klávesnice. V levém horním rohu obrazovky jsou nastavení systému, adresář základních složek ( Dokumenty, Počítač, Plocha, Síť ) a pro nás nejdůležitější – seznam aplikací.

V něm najdeme nejen nainstalované aplikace, ale také zde máme přístup ke stažení dalších programů pomocí Centra softwaru pro Ubuntu.



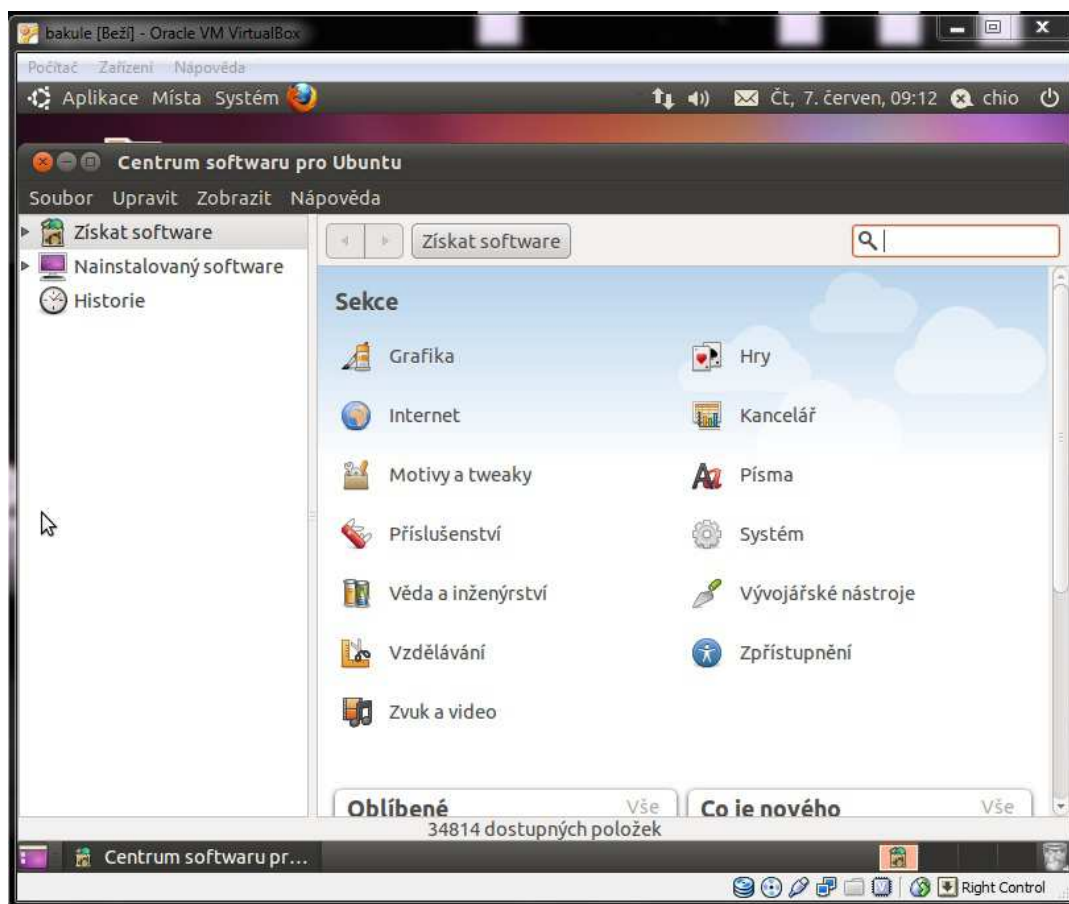
Obrázek 1.5 – Vzhled plochy v Ubuntu

## 2. Programovací jazyky

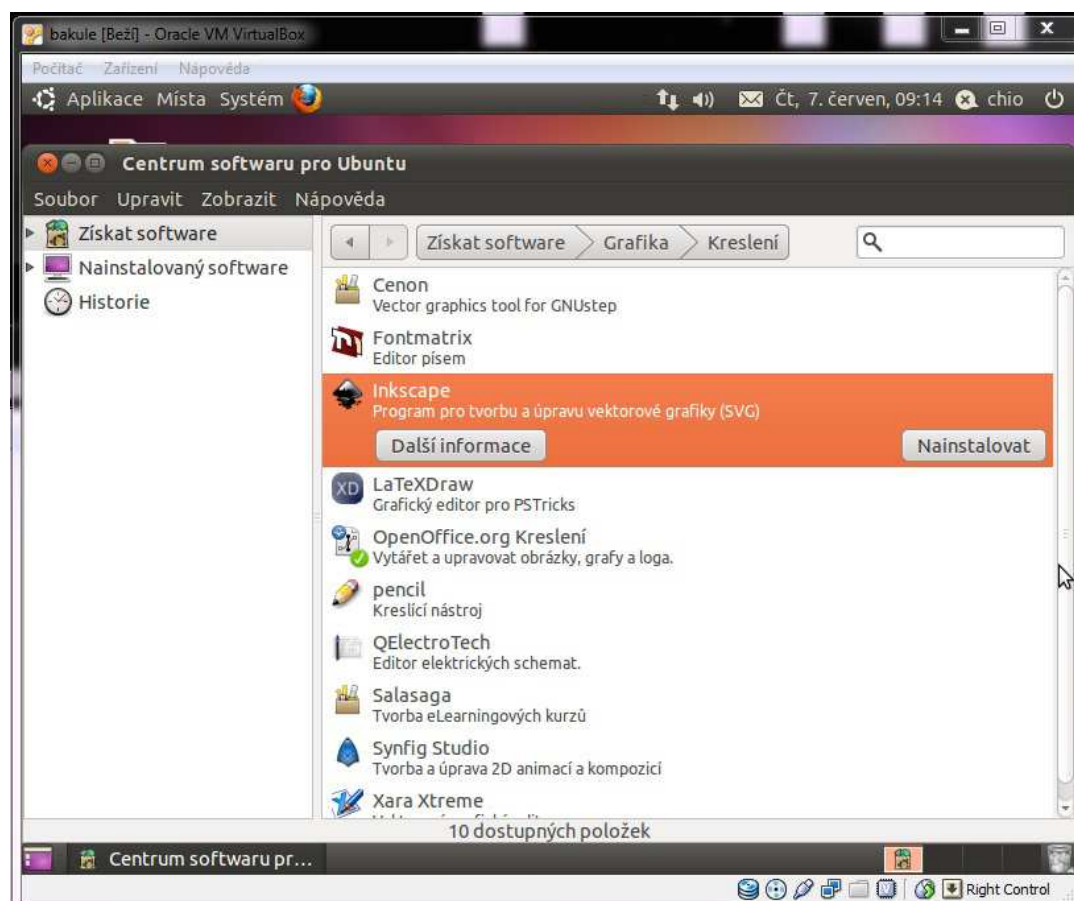
### 2.1 Instalace

Otevřeme si Centrum softwaru pro a zde si najdeme buď dle kategorií či přímo dle názvu vhodný program. Po nalezení programu se nam v možnostech nabídne zobrazení více informací a samotná instalace. Uživatel tak má možnost si najít program, který pokrývá celou škálu potřebných funkcí. Samotná instalace pak probíhá automaticky bez nutného zásahu od nás.

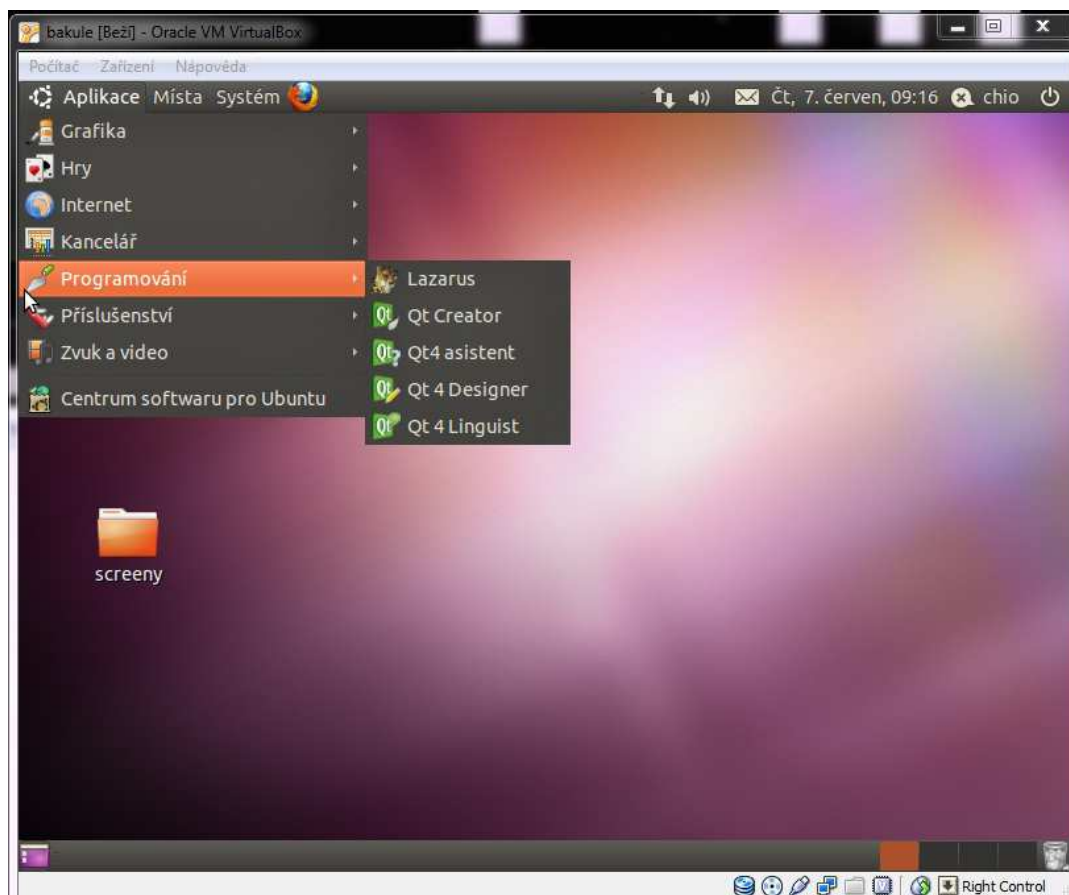
Pro náš další postup v úkolu si nainstalujeme oba zvolené programy. Lazarus pro programovací jazyk Pascal a QT4 pro C++. Po nainstalování pak programy najdeme v záložce Aplikace a v podzáložce Programování.



Obrázek 2.1 – Centrum softwaru pro Ubuntu



Obrázek 2.1 – Detaily o programu



Obrázek 2.1 – Přehled nainstalovaných programů



## 2.2 Lazarus

### 2.2.1 Základní informace

Toto vývojové prostředí je dostupné nejen pro Linux, ale i pro windows. Používá se k vývoji v jazycích Pascal, ale také Delphi. Je tedy vhodný jak pro programování přes příkazový řádek, tak pro objektovou práci. Vývojové prostředí se podobá Delphi. Debug informace se ukládají přímo do exe souboru. Je stabilní s rozsáhlými možnostmi vizualizace. Lze kompilovat pod Linux, Windows a Mac. Dá se použít pro kompilaci aplikací pro mobilní telefony. Použití tohoto programu v roce 2011 udává tabulka.

Widget set	Supported Operating Systems	Status
Windows API, GDI	Windows	Mainstream use
Windows CE API, GDI	Windows CE	Mainstream use
GTK+ 1.2.x	Linux (via X11)	Deprecated
GTK+ 2.8+	Windows, Linux (X11 and framebuffer) and Mac OS X (via X11)	Mainstream use
Qt 4.5+	Windows, Linux (X11 and framebuffer), Mac OS X	Mainstream use. Works in Windows, Linux, Mac OS X, Maemo, etc
Cocoa	Mac OS X	In progress
Carbon	Mac OS X	Mainstream use
fpGUI	Windows, Windows CE, Linux (via X11)	Initial stage
Lazarus Custom Drawn Controls	Android, Windows, Linux (via X11), Mac OS X	Initial stage

Obrázek 2.2.1 – Tabulka použitelnosti v roce 2011

### 2.2.2 Používání databází

Lazarus také podporuje používání databází. Po nainstalování doplňků se může používat spolu s MySQL, PostgreSQL a dalšími. Lze ho také používat pro webový vývoj.

#### *Seznam použitelných databází s potřebnými komponenty v angličtině:*

- [PostgreSQL](#), with the [PSQL package](#)
- [dBase](#) and [FoxPro](#) can be supported without the need for an external server or library through the [TDbf component](#)
- [MySQL](#) works
- [InterBase](#) / [Firebird](#) works via standard SQL DB Package, and the Open Source IBX for Lazarus (see external link below)
- [SQLite](#) needs a single external library and the TSQLiteDataset component
- [MSSQL](#) is working with [Zeoslib](#). FPC versions after 2.6.0 will allow [FreeTDS](#) for MSSQL access.
- [InterBase](#) / [Firebird](#) also works with the latest [ZeosLib](#)
- [SQLdb](#) which is shipped by default provides connectivity against [PostgreSQL](#), [Oracle](#), [ODBC](#), [MySQL](#), [SQLite](#) and [InterBase](#)
- [ZeosDBO](#) originally written for Delphi, now works for Lazarus as well, providing connectors roughly as many as SQLdb

### 2.2.3 Porovnání s Delphi

Rozdíly mezi Lazarusem a Delphi:

- větší velikost souborů oproti Delphi kvůli ukládání debug informací do exe souboru. Lze zredukovat nastavením kompilátoru či použitím programů [Strip](#) či [UPX](#).
- Komponenty pro Delphi lze použít po konverzi i v Lazarusu
- Chybí propojení s MS Office (Excelové tabulky)
- Práce se síťovými prostředky je díky množství dodatkových balíčků na dobré úrovni
- Nepodporuje knihovny .NET, lze ovšem použít kód v nich obsažený
- Od verze 2.2.0 podporuje COM
- Nepodporuje dynamické balíčky
- Není plně kompatibilní s VCL kvůli designu, je třeba ručně dopravit

### 2.2.4 Aplikace

Anglický popis aplikací použitelných pro program Lazarus:

- [Audio X](#) is a media management tool that can organize and sort your media without using a separate database, so the data set is always synchronized. Many audio formats are usable directly but you can also organize your [LP](#) or [CD](#) collection with it. It stores data in [XML](#) files so they are also viewable with a web browser. Made with Lazarus/FPC.
- [Cactus Jukebox](#) is an audio player that comes with a database to organize your [MP3](#) file collection. It is platform independent and currently available for Linux and 32-bit Windows. Made with Lazarus/FPC.
- [Cartes du Ciel](#) is a free [planetarium](#) program for Linux, Mac OS X and Windows. The software maps out and labels most of the constellations, planets, and objects you can see with a telescope. It's fully written in Lazarus/FPC and released under GPL.
- [Becape](#) is an open source [backup](#) tool aimed at personal/desktop usage. It does incremental backups and stores the backup info in an [SQLite](#) database allowing the restoration of the exact state of the backed files at a chosen date. It's fully written in Lazarus/FreePascal.
- [CQRLOG](#) is an advanced [ham radio](#) logger based on a [Firebird](#) database. Provides radio control based on hamlib libraries (currently support of over 140 radio types and models), DX cluster connection, QRZ callbook (web version), a grayliner, ON6DP

QSL manager database support and a most accurate country resolution algorithm based on country tables developed by OK1RR. CQRLOG is strongly focused on easy operation and maintenance. Made with Lazarus/FPC.

- [Cheat Engine](#) is an open source memory scanner/hex editor/debugger. It is useful for cheating in computer games. Since version 6.0 it is compiled with Lazarus/FPC.
- [Dedalu](#) is a collection of small and simple projects developed in Lazarus/FPC by Giuseppe Ridinò (aka Pepecito). They are games, editors, utilities, etc.
- [DarGUI](#) is a graphical interface for Denis Corbin's [Dar](#) backup utility, developed using Lazarus/FPC.
- [HJSplit for Linux](#) is a freeware file splitter. HJSplit supports file sizes of over 100 Gigabytes, Split, Join/Recombine, MD5 checksums, file-compare and "run without install". Runs on Linux. Created using Free Pascal and Lazarus.
- [KComm](#) is a ham radio logging program developed specifically for Elecraft K2 and K3 transceivers. It works under Windows or Linux. It is designed for the average ham radio operator who wants an easy to use logging program also suitable for casual contest use. It supports CW Transmit from the keyboard, CW decoding, PSK31/63 transmit and receive. DX Cluster client and integration with CW Skimmer. Made with Lazarus/FPC.
- [LockHunter](#) a free utility used to force deletion of any files, even those that are locked by some processes. Made with Lazarus/FPC, currently supports Windows 32 / 64 bit only.
- [Master Maths](#) specializes in computer based training and maths. The third incarnation of their flagship product is developed using Lazarus, [Firebird](#) database and [tiOPF v2](#). The product has two parts. An Administration application and a Learner Browser (used to view and mark the teaching modules). The Learner Browser uses [Adobe Flash](#) extensively and is a CGI web application. The complete product runs under Linux and Windows. Made with Lazarus/FPC and uses the [fpGUI Toolkit](#) as its widgetset.
- [MRIcron](#) is a medical image visualization and analysis package. The software provides tools for drawing volumes of interest and volume rendering. In addition, it includes non-parametric statistical mapping ([npm](#)) and conversion of images from DICOM format to NIfTI format ([dcm2nii](#)). Made with Lazarus/FPC, it is currently

available for Windows (using WinAPI), Linux (GTK1, GTK2 or QT) and Mac OS X (Carbon or GTK1).

- [MRIcroGL](#) uses [GLScene](#) to provide hardware accelerated volume rendering. It can display grayscale images (e.g. CT, MRI) or full color images (e.g. photographs from the visible human dataset).
- [Music Player by Freebyte.com](#) is a freeware no-frills music player designed for Linux. It supports MP3, Wav, OGG and AIFF files. Created using Lazarus, Free Pascal and the Bass audio library.
- [MyNotex](#) is a free software for Gnu/Linux useful to take and to manage textual notes.
- [OutKafe](#) is a next-generation free and open source [cybercafe](#) management suite. OutKafe is licensed under the GNU GPL version 3, and is thus considered [free software](#). OutKafe is developed by A.J. Venter with sponsorship from OutKast I.T. Solutions C.C. and the kind contributions of several volunteers. OutKafe is running hundreds of cybercafe's at business, schools and other establishments around the world. Made with Lazarus/FPC.
- [Peazip](#) is an open source archiver, made with Lazarus/FPC
- [QFront](#) is a platform independent frontend for the CPU emulator [QEMU](#). Made with Lazarus/FPC.
- [SimThyr](#) is a continuous numerical simulation program for [thyrotropic feedback control](#) used for research and education (open source, cross platform for Mac OS X, Windows and Linux).
- [Smith4VNA](#) is a program for driving "miniVNA" Vector Analyzer via USB port. The results of measurements are both table and Smith Chart. Made with Lazarus/FPC, Linux + Windows32 platforms.
- [SPINA](#) is a [medical cybernetic](#) software package that allows for calculating constant [structure parameters](#) of [endocrine](#) feedback control systems from hormone levels obtained *in vivo*. This free software comes with source code in Lazarus/[FPC](#) and [PocketStudio](#).
- [TreePad Lite for Linux](#) is a freeware tree-based personal information manager for Linux. It has been designed to manage, store, edit, organize and browse any type of textual data, such as: notes, emails, articles, links, phone numbers, addresses, scraps pasted from the Web, etc. Created using Lazarus and Free Pascal.

- [TruckBites](#) business management software for independent trucking companies and owner/operators in the USA. Written with Lazarus/FPC under contract by Tony Maro for both Linux and Windows for "Partners in Trucking, LLC".
- [Virtual Magnifying Glass](#) is a free, open source, screen magnification tool for Windows, Mac OS X and Linux.
- [Double Commander](#) is a cross-platform open source file manager with two panels side by side.
- [Wireless Orders for Mini Bar Cafe](#) Win32 TCP/IP Application Server, Win32 TCP/IP Client, WinCE TCP/IP Client.

## 2.3 QT4

### 2.3.1. Popis

Přezdívaný taky jako cute ( milý) je multiplatformový framework pro aplikace. Pouívá se hlavně k vývoji softwaru aplikací s grafickým uživatelským rozhraním (GUI). Lze ho také využít pro konzolové aplikace či nástroje pro příkazový řádek pro servery. Používá se pro Adobe Photoshop, VLC Media Player, Virtual Box či Mathematica. Používají do firmy jako ESA, Dreamworks, Panasonic, Philips, Google, Hp nebo Siemens. Webový prohlížeč Opera dho používá jako toolkit pro vzhled v Linuxových platformách.

### 2.3.2 Vývoj

QT je vyvíjen jako [open source](#). Původně ve firmě Troltech, pak Nokia a později byla licence prodána společnosti Digia. I v dnešní době se vývojáři od Nokie aktivně podílí na vývoji.

V kódování se používá standartního jazyka C++, který je obohacen o speciální generátor kódu. Lze ho použít pro většinu mobilních či desktopových platform. Umí pracovat s SQL databázemi, zpracováním XML, podporou sítí, správou procesů a multiplatformový výstup pro zvládání různých typů souborů.

### 2.3.3 Podpora programů

Podporuje kompilaci jazyka C++ či kód pro Visual Studio. Při práci s C++ nebo Javou je k dispozici několik balíčků, které v sobě obsahují množství pomocných vychytávek, grafických textur, úprav stylů a dalších, potřebných k vývoji moderního vzhledu programů.

## 3. Regulace proudu v kotve DC motoru

### 3.1 Matlab

Zdrojový kód pro Matlab byl zadán jako součást úkolu. Naším cílem je předělat ho pomocí vybraných programů pod operační systém Linux. Jedná se o program eegulace proudu v kotve DC motoru napajeneho ze 4-kvadrantoveho pulsnio menice (H-mustku).

### 3.2 Lazarus

Jedná se o konzolovou aplikaci, která je pro začátečníky jednodušší k naprogramování. Nemusíme řešit objektové programování a stačí nám jednodušší příkazy. Kostra a schéma programu je prakticky totožné s Matlabem. Použijeme pouze příkazy pro Pascal. V mém případě došlo k problému s vykreslením finálních grafů z hodnot dvourozměrného pole. I přes veškerou mou snahu se mi nepodařilo vykreslit výsledek. Je to způsobeno malými znalostmi práce s grafikou a znalostí kompilačního programu.

### 3.3 QT4

Jde o grafickou aplikaci naprogramovanou v jazyce C++ s využitím knihovny Qt 4. Qt 4 vyžaduje od programátora objektový přístup a není tedy pro začátečníky tak jednoduché, jako například Lazarus (Pascal).

Vlastní kód výpočtu a uchování hodnot je velmi podobný kódu v Matlabu. Je pouze nutné dbát na správné datové typy, které jsou vyžadovány.

Problém však nastává při vykreslování grafů do aplikace, protože je nutný objektový přístup. Bylo by pravděpodobně nutné, napsat vlastní komponentu implementovanou

objektově, která se poté bude starat o vlastní vykreslení grafů. Tento postup rozhodně není vhodný pro začátečníka a také se mi bohužel nepodařilo dosáhnout požadovaného grafického výstupu.

## **Závěr**

Podářilo se mi úspěšně nainstalovat požadované programy k otestování. Při jejich porovnávání jsem zjistil, že snazší práce bude při kompilování v programu Lazarus vzhledem k jednoduššímu přístupu k aplikaci. Bohužel u obou programů jsem nedospěl ke grafickému výstupu, cožn je způsobeno nutností přídavných aplikací a mé pouze základní znalosti v programování.



## Seznam literatury a informačních zdrojů

- [1] Mareš, Amadeo: 1001 tipů a triků pro C# 2010, Computer Press, 2011
- [2] Pavel Herout: Učebnice jazyka C, Koop, 5.vydání, ISBN 978-80-7232-351-7
- [3] Putz, Karel : Pascal, Grada, 2007, ISBN 978-80-247-1255-0
- [4] <http://www.linuxexpres.cz/blog/nebud-lazar-bud-lazarus>
- [5] [http://wiki.freepascal.org/Lazarus\\_Documentation/sk](http://wiki.freepascal.org/Lazarus_Documentation/sk)
- [6] <http://en.wikipedia.org>
- [7] <http://www.linux.cz/>

## Přílohy

### Příloha 1 – Původní program v Matlabu

```
%regulace proudu v kotve DC motoru napajeneho ze 4-kvadrantoveho pulsniho
%menice (H-mustku)

clear;
dt=5e-6;%[s] %krok simulace

%parametry regulatoru
kr=2;
Taur=0.003;

fp=5000;%frekvence nosne pily, Hz
urmax=1;%rozsah ridiciho napeti mustku
U=50;%napajeci napeti mustku [V]
R=1;%odpor vinuti [Ohm]
L=0.05;%indukcnost vinuti [H]

dupila=4*fp;%derivace signalu nosne pily

%pocatecni podminky
t=0;i=0;sum=0;ur=0;iw=0;compareReg=0;
u=0;Ui=0;up=-1;

%vlastni vypocetni smycka
while t<0.02
    %definovani pozadovaneho prubehu proudu
    if t>0.002 iw=1; end;
    %definovani prubehu poruchy
    if t>0.01 Ui=30; end;

    %pila tvorici PWM (symetricka)
    up=up+dupila*dt;
    if up>urmax dupila=dupila*(-1);end;
    if up<-urmax
        dupila=dupila*(-1);
        compareReg=ur;%do compare registru PWM se priradi vystup regulatoru z minule
        %periody PWM - doba vypoctu algoritmu Pi regulatoru trva celou 1/fp (nejhorsi pripad)
        e=iw-i; %regulacni odchylka
        %PI (P) regulator
        if abs(ur)<urmax sum=sum+1/Taur*e/fp; end;%integrace reg. odchylky, pokud neni
        vystup omezen
        ur=kr*(e+sum);
        %omezovac regulatoru
        if ur>urmax ur=urmax; end;
        if ur<-urmax ur=-urmax; end;
    end;
end;
```

```
% PWM
if compareReg>up u=U; else u=-U; end;

% vypočet diference proudu (z rovnice  $u=R*i+L*di/dt$ )
di_dt=1/L*(u-R*i-Ui);

% numerická integrace Eulerovou metodou
i=i+di_dt*dt;

% zapis výsledku do pole
if t==0 vys=[t,iw,ur,u,Ui,i]';
else vys=[vys [t,iw,ur,u,Ui,i]']; end;
% posun času na další vypočetní krok
t=t+dt;
end;

subplot(3,1,1);
plot(vys(1,:),vys(2,:));hold on;
plot(vys(1,:),vys(6,:),'r');hold off;
legend('iw[A]','i[A]');
subplot(3,1,2);
plot(vys(1,:),vys(3,:));
legend('ur[V]');
subplot(3,1,3);
plot(vys(1,:),vys(4,:));hold on;
plot(vys(1,:),vys(5,:),'r');hold off;
legend('u[V]','Ui[v]');
set(gca,'YLim',[-U*1.1,U*1.1]);
xlabel('time [s]');
```

## Příloha 2 – Program předělaný pro Pascal v programu Lazarus

```
program motor;

{$mode objfpc}{$H+}

uses
  {$IFDEF UNIX}{$IFDEF UseCThreads}
  cthreads,
  {$ENDIF}{$ENDIF}
  Classes, SysUtils, CustApp, CRT, Graph
  { you can add units after this };

var  kr,pole,j,n:integer;
     ovladacg,modg:smallint;
     Taur,fp,urmax,U,R,L,dupila:extended;
     di_dt,dt,t,e:extended;
     i,sum,ur,iw,compareReg,um,Ui,up:extended;
     vys: array [1..100,1..6]of smallint;

type

  { grafy }

  grafy = class(TCustomApplication)
  protected
    procedure DoRun; override;
  public
    constructor Create(TheOwner: TComponent); override;
    destructor Destroy; override;
    procedure WriteHelp; virtual;
  end;

  { grafy }

procedure grafy.DoRun;
var
  ErrorMessage: String;
begin
  // quick check parameters
  ErrorMessage:=CheckOptions('h','help');
  if ErrorMessage<>'' then begin
    ShowException(Exception.Create(ErrorMessage));
    Terminate;
    Exit;
  end;

  // parse parameters
  if HasOption('h','help') then begin
```

```
WriteHelp;
Terminate;
Exit;
end;

{ add your program here }

begin
// parametry regulatoru
kr:= 2;
Taur:= 0.002;
dt := 5e-6; // krok simulace
fp:= 5000.0; // frekvence nosne pily, Hz
urmax:= 1.0; // rozsah ridiciho napeti mustku
U := 50.0; // napajeci napeti mustku [V]
R := 1.0; // odpor vinuti [Ohm]
L := 0.05; // indukcnost vinuti [H]

dupila:= 4 * fp; // derivace signalu nosne pily

// pocatecni podminky
t := 0.0; i := 0.0; sum := 0.0; ur := 0.0;
iw := 0.0; compareReg := 0.0;
um := 0.0; Ui := 0.0; up := -1.0;
pole := 1;
end;
// vlastni vypocetni smycka
while (t < 0.02) do begin

// definovani pozadovaneho prubehu proudu
if (t > 0.002) then iw := 1.0;

// definovani prubehu poruchy
if (t > 0.01) then Ui := 30;

// pila tvorici PWM (symetricka)
up := up + dupila * dt;
if (up > urmax) then dupila := -1 * dupila;

if (up < -urmax) then
begin
dupila := -1 * dupila;
compareReg := ur; // do compare registru PWM se priradi vystup regulatoru z minule
periody PWM - doba vypoctu algoritmu Pi regulatoru trva celou 1/fp (nejhorsi pripad)
e := iw - i; // regulacni smycka
end;
// PI (P) regulator
if (Abs(ur) < urmax) then sum := sum + 1 / Taur * e / fp; // integrace reg. odchylky,
pokud neni vystup omezen
```

```
    ur := kr * (e + sum);

    // omezovac regulatoru
    if (ur > urmax) then ur := urmax;

    if (ur < -urmax) then ur:=-urmax;

// PWM
if (compareReg > up) then u := U
else um := -U;

// vypocet difference proudu (z rovnice u=R*i+L*di/dt)
di_dt := 1 / L * (u - R * i - Ui);

// numericka integrace Eulerovou metodou
i := i + di_dt * dt;

// zapis vysledku do pole

vys[pole,1] := round(t);
vys[pole,2] := round(iw);
vys[pole,3] := round(ur);
vys[pole,4] := round(u);
vys[pole,5] := round(Ui);
vys[pole,6] := round(i);

// posun casu na dalsi vypocetni krok
t := t + dt;
Inc (pole);
end;

Clrscr;
Initgraph(ovladacg,modg,' ');
Writeln('závislost proudu na čase ');
for j:= 1 to n do begin
    setcolor(3);
    putPixel(100+vys[j,1],vys[j,1],10);
    setcolor(2);
    putPixel(100+vys[j,1],vys[j,2],10);
    setcolor(1);
    putPixel(100+vys[j,1],vys[j,6],10);

    end;
Closegraph;
{
    putPixel(pole,3,10);
    putPixel(pole,4,10);
    putPixel(pole,5,10);
```

```
    putPixel(pole,6,10);

subplot(3,1,1);
plot(vys(1,:),vys(2,:));hold on;
plot(vys(1,:),vys(6:),'r');hold off;
legend('iw[A]','i[A]');
subplot(3,1,2);
plot(vys(1,:),vys(3,:));
legend('ur[V]');
subplot(3,1,3);
plot(vys(1,:),vys(4,:));hold on;
plot(vys(1,:),vys(5:),'r');hold off;
legend('u[V]','Ui[v]');
set(gca,'YLim',[-U*1.1,U*1.1]);
xlabel('time [s]');*/
}

end;
// stop program loop
{ Terminate; }

constructor grafy.Create(TheOwner: TComponent);
begin
    inherited Create(TheOwner);
    StopOnException:=True;
end;

destructor grafy.Destroy;
begin
    inherited Destroy;
end;

procedure grafy.WriteHelp;
begin
    { add your help code here }
    writeln('Usage: ',ExeName,' -h');
end;

var Application: grafy;

begin
    Application:=grafy.Create(nil);
    Application.Title:='grafy';
    Application.Run;
    Application.Free;
end.
```

### Příloha 3 – Hlavní struktura v programu C++

```
#include "MainWindow.h"

#include <QDebug>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent)
{
    ui.setupUi(this);

    float dt = 5e-6; // krok simulace

    // parametry regulatoru
    int kr = 2;
    float Taur = 0.002f;

    float fp = 5000.0f; // frekvence nosne pily, Hz
    float urmax = 1.0f; // rozsah ridiciho napeti mustku
    float U = 50.0f; // napajeci napeti mustku [V]
    float R = 1.0f; // odpor vinuti [Ohm]
    float L = 0.05f; // indukcnost vinuti [H]

    float dupila = 4 * fp; // derivace signalu nosne pily

    // pocatecni podminky
    float t = 0.0f, i = 0.0f, sum = 0.0f, ur = 0.0f, iw = 0.0f, compareReg = 0.0f;
    float u = 0.0f, Ui = 0.0f, up = -1.0f;

    // vlastni vypocetni smycka
    while (t < 0.02f)
    {
        // definovani pozadovaneho prubehu proudu
        if (t > 0.002f)
            iw = 1.0f;

        // definovani prubehu poruchy
        if (t > 0.01f)
            Ui = 30.0f;

        // pila tvorici PWM (symetricka)
        up = up + dupila * dt;
        if (up > urmax)
            dupila = -1 * dupila;

        if (up < -urmax)
        {
            dupila = -1 * dupila;
        }
    }
}
```



```
compareReg = ur; // do compare registru PWM se priradi vystup regulatoru z minule
periody PWM - doba vypoctu algoritmu Pi regulatoru trva celou 1/fp (nejhorsí pripad)
float e = iw - i; // regulacni smycka

// PI (P) regulator
if (qAbs(ur) < urmax)
    sum = sum + 1 / Taur * e / fp; // integrace reg. odchylky, pokud neni vystup omezen

ur = kr * (e + sum);

// omezovac regulatoru
if (ur > urmax)
    ur = urmax;

if (ur < -urmax)
    ur=-urmax;
}

// PWM
if (compareReg > up)
    u = U;
else
    u = -U;

// vypocet difference proudu (z rovnice  $u=R*i+L*di/dt$ )
float di_dt = 1 / L * (u - R * i - Ui);

// numericka integrace Eulerovou metodou
i = i + di_dt * dt;

// zapis vysledku do pole
Vysledek vys;
vys.vys[0] = t;
vys.vys[1] = iw;
vys.vys[2] = ur;
vys.vys[3] = u;
vys.vys[4] = Ui;
vys.vys[5] = i;

pole_vysledku.append(vys);

qDebug() << t << iw << ur << u << Ui << i;

// posun casu na dalsi vypocetni krok
t = t + dt;
}

/*subplot(3,1,1);
plot(vys(1,:),vys(2,:));hold on;
plot(vys(1,:),vys(6,:),'r');hold off;
```

```
    legend('iw[A]', 'i[A]');  
    subplot(3,1,2);  
    plot(vys(1,:), vys(3,:));  
    legend('ur[V]');  
    subplot(3,1,3);  
    plot(vys(1,:), vys(4,:)); hold on;  
    plot(vys(1,:), vys(5,:), 'r'); hold off;  
    legend('u[V]', 'Ui[v]');  
    set(gca, 'YLim', [-U*1.1, U*1.1]);  
    xlabel('time [s]');*/  
}
```