VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Tvorba vlastných blokov pre balíček LaTeX – TikZ na úrovni PGF

Use of PGF for Creation of New Blocks in LaTeX – TikZ Package

2016

Laura Kubicová

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Laura Kubicová**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Tvroba vlastních bloků pro balíček LaTeX -- TikZ na úrovni PGF
Use of PGF for Creation of New Blocks in LaTeX -- TikZ Package

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat detailně jednotlivé kroky při tvorbě vlastních bloků na platformě PGF pro jejich další využití v prostředí TikZ.

1. Popište souvislosti a významné rozdíly mezi použitím PGF a TikZ.
2. Popište a na jednoduchých příkladech demonstrujte možnost tvorby uživatelských bloků v PGF. Vysvětlete, co je potřeba k tomu, aby bylo možné bloky správně proporcionálně zvětšovat či zmenšovat.
3. Popište tvorbu komplikovanějších bloků, např. s možností specifikace počtu vstupů apod...
4. Popište tvorbu "uzlů" a "kotev" u uživatelských bloků a pozicování bloků s využitím těchto uzlů a kotev v prostředí TikZ.

Seznam doporučené odborné literatury:

[1] Rybička, Jiří. *LaTeX pro začátečníky*. 3. vyd. Brno: 2003, Konvoj. 238 stran. ISBN 80-7302-049-1
[2] Návod k balíčku TikZ, dostupný pod odkazem Documentation na URL: http://www.texample.net/tikz/
[3] Manuál k balíčku TikZ a PGF. URL: http://www.texample.net/media/pgf/builds/pgfmanualCVS2012-11-04.pdf

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 29.04.2016

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

V Ostrave 1. července 2016 .....

Súhlasím so zverejnením tejto bakalárskej práce podľa požiadavok čl.26 odst.9 *Studijního a zkúšebního řádu pro studium v bakalářskych programech VŠB-TUO Ostrava.*

V Ostrave 1. července 2016 .....

Rada by som poďakovala vedúcemu bakalárskej práce Ing. Janovi Skapovi Ph.D. za trpezlivosť, ochotu a cenné rady pri tvorbe práce.

Abstrakt

Bakalárska práca je zameraná na typografický systém LaTeX a s ním súvisiacimi balíkmi Ti*k*Z &PGF pri tvorbe vlastných blokov z oblasti elektrotechniky. Cieľom práce bolo vytvoriť manuál, ktorý bude nadstavbou práce Tomáša Pavlorka.
Celá práca je rozdelená na kapitoly, ktoré sú radené podľa zložitosti, pričom každá z nich obsahuje niekoľko príkladov z oblasti elektrotechniky.

Klíčová slova: TeX, LaTeX,Ti*k*Z, PGF, PGFplots

Abstract

This Bachelor thesis is focused on the typesetting system LaTeX and within related packages Ti*k*Z &PGF in creating blocks from the field of electrical engineering. The main aim of bachelor thesis is to create manual that will be a continuation of the thesis of Tomas Pavlorek. This thesis as whole is divided into chapters that are as well sequenced in order, based on their complexity. Therefore each chapter includes a number of examples from the electrical engineering field.

Key Words: TeX, LaTeX, Ti*k*Z, PGF, PGFplots

Obsah

Seznam použitých zkratek a symbolů

Ti*k*Z	–	Ti*k*Z ist kein Zeichenprogramm
PGF	–	Portable Graphics Format
IEC	–	Štandard definovaný medzinárodnou elektrotechnickou spločnosťou
ISO	–	Medzinárodná organizácia pre normalizáciu

Zoznam obrázkov

Seznam výpisů zdrojového kódu

Úvod

Cieľom bakalárskej práce bolo vytvoriť manuál, ktorý detailne popisuje kroky pri tvorbe vlastných blokov na platforme PFG a ich ďalšie možné použitie v prostredí Ti*k*Z. Samotná práca ako nadstavba balíčka Ti*k*Z, obsahuje viacero kapitol a slúžiť by mala užívateľom, ktorí sa pre vykresľovanie od najjednoduchších grafických ilustrácií až po logické obvody rozhodli pre balík PGF. Obsahom práce je aj podkapitola, ktorá sa zaoberá použitím balíčkov, slúžiacich na vykreslenie jednotlivých obvodov.

Stručný prehľad kapitol:

- **1. Kapitola:** je úvodom do balíčka Ti*k*Z &PGF a PGFplots, kde sú predstavené základné pojmy a práca so základnými objektami.
- **2. Kapitola:** sa zaoberá tvorbou uzlov a kotiev a ich využitím v prostredí Ti*k*Z. Obdobne zahrňuje balíčky, ktoré slúžia na tvorbu elektrických a logických obvodov.
- **3. Kapitola:** pojednáva príklady tvorby blokov na vrstve PGF, ich proporcionálne zmenšovanie a zväčšovanie.
- **4. Kapitola:** popisuje tvorbu komplikovanejších blokov s možnosťou špecifikácie počtov vstupov.

1 Základné pojmy

Nemecký profesor Till Tantau je autorom balíčka Ti*k*Z &PGF, ktorý slúži k tvorbe vektorovej grafiky v systéme LaTeX. Medzi pozitíva balíka, ktorý je často označovaný za nástupcu nástroja `PSTricks`[1], patrí možnosť vytvárania súborov vo formáte PDF či PS, ale aj možná kompatibilita s balíkom `Beamer` slúžiacim na tvorbu prezentácií v systéme LaTeX.[1]
PGF (Portable Graphics Format) predstavuje súbor makier, ktoré umožňujú vytvárať grafiku v dokumentoch pomocou špeciálneho `\pgfpicture` prostredia.

Balík pozostáva z niekoľkých vrstiev:

- **Systémová vrstva:** je najnižšou vrstvou, ktorú bežný užívateľ priamo nevyužije. Táto vrstva slúži na modifikáciu systémových príkazov `\special` vzhľadom na použitý prekladač a s jej pomocou môžeme prekladať `.dvi` súbor do formátu `.ps` či `.pdf` nezávisle na zvolenom programe.
- **Základná vrstva:** poskytuje súbor základných príkazov, ktoré umožňujú užívateľovi vytvárať zložitejšiu grafiku jednoduchším spôsobom ako priamym použitím systémovej vrstvy. Základná vrstva pozostáva z:
  - **Jadro:** niekoľkých vzájomne závislých balíčkov, ktoré môžu byť načítané len spoločne.
  - **Moduly:** rozširuje jadro ďalšími špeciálnymi príkazmi ako napríklad príkazmi pre spravovanie uzlov.
- **Užívateľská vrstva Ti*k*Z:** je skupina príkazov vychádzajúca z kombinácie METAFONT[2] a PostScript[3] a ktorej primárnou úlohou je uľahčenie použitia základnej vrstvy.[1]
  Problémom pri priamom použití základnej vrstvy je, že kód napísaný pre túto vrstvu je až príliš rozsiahly. [2]

1.1 Rozširujúce balíčky

PGF balík ponúka niekoľko užitočných rozširujúcich balíčkov, ktoré nie sú primárne určené na vytváranie grafiky a súčasne môžu byť použité nezávisle na PGF. Sú zahrnuté v tomto balíku z konvencie, ale aj preto, že ich funkcionalita je úzko prepojená s PGF.[2]

1.1.1 PGFkeys

```
\usepackage{pgfkeys} % LaTeX{}
\usemodule[pgfkeys]% ConTeXt{}
```

[1] súbor makier, umožňujúcich vytváranie postskriptových obrázkov v prostredí LaTeX,TeX
[2] jazyk vytvorený na generovanie fontov, autor Donald Knuth
[3] jazyk slúžiaci na tvorbu vektorovej grafiky, vytvorený spoločnosťou Adobe Systems

Balík definuje flexibilný systém správy kľúčov, býva automaticky načítaný balíkmi Ti*k*Z aj PGF no môže byť použitý aj nezávisle na PGF. Typicky (ale nie nevyhnutne) ak je nejaký kód združený s kľúčom na spracovanie takéhoto kódu slúži príkaz `\pgfkeys`. Takýto príkaz vezme zoznam párov kľúčových hodnôt, ktoré sú volané. Každý pár je vo forme `\<key>=<value>`.
V istom zmysle je podobný balíku `\keyval`[6] a jeho nadstavbe `\xkeyval`[7].
I keď filozofia `\pgfkeys` balíka je od nich rozdielna, dokáže tento balík s nimi spolupracovať.

Hlavné rozdiely medzi použitím `\ykeyval` a `\pgfkeys`:

- `\pgfkeys` organizuje kľúče do stromov, zatiaľ čo `\keyval` a `\xkeyval` do rodín
- `\pgfkeys` je mierne pomalší oproti `\keyval`, ale nie výrazne
- `\pgfkeys` podporuje kód, v ktorom je viac argumentov pre kľúč
- `\pgfkeys` podporuje handlery volané najmä pokiaľ kľúč nie je známy

V nasledujúcom príklade je definovaný handler kľúča, ktorý je volaný cez `/.code` kde sme nastavili 2 argumenty `{#1}` a `{#2}`, tie predstavujú parametre `{pekný}` a `{slnečný}`.

```
\pgfkeys{/my key/.code 2 args=Prajem #1 a #2 den.}
\pgfkeys{/my key={pekny}{slnecny}}
```

Výpis 1: Ukážka nastavenia handlera

Prajem pekný a slnečný deň.

1.1.2 PGFfor

```
\usepackage{pgffor} % LaTeX{}
\usemodule[pgffor]% ConTeXt{}
```

Balík, ktorý definuje užitočnú `\foreach` konštrukciu. Automaticky býva načítaný balíkom Ti*k*Z ale nie PGF, súčasne však funguje dobre s oboma.
Syntax príkazu je definovaná ako: `\foreach<premenná>` v `{<liste>}<príkaz>` Najjednoduchšia možnosť použitia tohto príkazu je keď `<premenná>` je jeden TEX - príkaz ako napríklad `\x` alebo `\point`.
Variantou najjednoduchšou pre `\<list>` sú hodnoty oddelené čiarkami a uzavreté sú zložených zátvorkách, možná alternatíva je definovanie si makrier, ktoré obsahujú list týchto hodnôt.

V nadchádzajúcom príklade je vytvorené makro, ktoré pozostáva z listu `\mojlist` a hodnôt uložených v zložených zátvorkách, ktoré sú volané konštrukciou `\foreach` a premenná `\x` ich

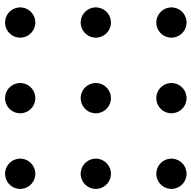
cez `\mojlist{[\x]}` vypíše.

[1][2][3][0]

```
\def\list{1,2,3,0}
\foreach \x in \mylist {[\x]}
```

Výpis 2: Výpis čísel definovaním makra

Variantou je aj `\foreach` konštrukcia priamo nasledovaná ďalšou `\foreach` konštrukciou, ktorá je v tomto prípade vyhodnotená ako `<príkaz>`. Príklad vykresľuje 3x3 štvorec vyplnený kruhmi s priemerom 0.2cm.



```
\foreach \x in {0,1,2}
  \foreach \y in {0,1,2}
    {
      \fill (\x,\y) circle (0.2cm);
    }
```

Výpis 3: Vykreslenie štvorca

1.1.3 PGFcalendar

```
\usepackage{pgfcalendar}% LaTeX{}
\usemodule[pgfcalendar]%ConTeXt{}
```

Obsahuje makrá pre tvorbu kalendárov. Tieto kalendáre sú typicky používané spolu s PGF grafickým rozhraním, ale môžeme ich tiež vsádzať pomocou normálneho textu. Rovnako aj tento balík môže byť použitý nezávisle na PGF.

- **1.:** poskytuje funkcie pre prácu s dátumom, s ktorých pomocou môžeme konvertovať dátum v ISO-štandarde[4] do Juliánskeho dátumu[3] a naopak, táto hodnota je uložená ako `{integer}`

[4]prvky sú zoradené v kalendári od najvýznamnejších po najmenej významné, YYYY-MM-DDThh:mm:ss±hh:mm(rok, mesic, deň v mesiaci, T- indikátor času, hodina, minúta, sekunda,± letný či zimný čas)

```
\pgfcalendardatetojulian{2015-08-12}{\mycount}
```

- **2.:** poskytuje makro pre sadzbu kalendárov, ktoré je jednoducho konfigurovateľné a flexibilné a nie je nutné, aby bolo volané v prostredí `{pgfpicture}`

V ukážke sme nastavili výpis z kalendára od 1.1.2016 do 15.1.2016 pričom sme podmienkou zaistili, že pokiaľ je nedeľa nastane koniec riadku.

01 02 03
04 05 06 07 08 09 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

```
\pgfcalendar{calend}{2016-01-01}{2016-01-31}
{
\pgfcalendarcurrentday\
\ifdate{Sunday}{\par}{}
}
```

Výpis 4: Ukážka výpisu mesiaca január

1.1.4 PGFpages

```
\usepackage{pgfpages} % LaTeX{}
```

Tento balík nie je primárne určený na tvorbu obrázkov, no napriek tomu je úzko spojený s PGF. V súčastnosti pracuje len v systéme LaTeX. Používaný na spojenie viacerých stránok do jednej ale umožňuje aj vkladať logá, vodotlače či obrázky, pridávať okraje na stránky a mnoho iného.

1.2 Práca s objektami

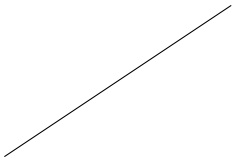
Pre prácu s balíkom PGF je nutné si najskôr tento balík aj s jeho rozširujúcimi súčasťami nainštalovať v `PACKAGE MANAGER`, kde sa nachádza pod názvom `pgf.sty`. Toto však neplatí pokiaľ sme si pri sťahovaní systému stiahli všetky balíčky a nainštalovali tak kompletný LaTeX systém. Je nutné si ho zaviesť do preambule, ktorá má platnosť na celý dokument, pomocou príkazu: `\usepackage{pgf}`.
Celá grafika tvorená v prostredí je hierarchicky štrukturovaná. Toto usporiadanie sa vo väčšine používa na identifikáciu určitej grafickej skupiny, ktorá používa rovnaký rozsah parametrov. Tak ak zmeníme farbu alebo typ čiary na prvky mimo tohto rozsahu nebude mať táto zmena na ne vplyv. Medzi najpoužívanejšie grafické prostredie patrí `\pgfpicture` do tohto prostredia sa vloží TeX box obsahujúci vykreslenú grafiku na danej pozícií. Vo väčšine prípadov musia byť príkazy z PGF balíčka zadefinované v tomto prostredí pričom všetky zmeny vykonané vo vnútri tohto prostredia sú vykonané na daný grafický objekt. Platí, že nie je možné nastaviť globálne grafické parametre mimo prostredia `\pgfpicture`.

Konštrukcia prostredia:

```
\documentclass[article]   % definovanie triedy projektu
\usepackage{pgf}  %zavedenie balíka PGF
\begin{document} %začiatok dokumentu
\begin{pfgpicture}  %zadefinované PGF prostredie
<obsah>
\end{pgfpicture} %ukončenie práce s prostredím
\end{document}  % koniec dokumentu
```

1.3 Vykresľovanie objektov

Podobne ako v balíku Ti*k*Z aj v PGF je najdôležitejšou entitou cesta angl.`\path`. Grafické objekty sú vytvárané z určitého počtu ciest a následne sú tieto objekty upravované. Cesty môžu byť uzavreté alebo otvorené, môžu sa pretínať alebo byť tvorené neukončenými časťami. Na začiatku sú cesty tvorené (angl. constructed) a neskôr používané (angl. used). V prípade, že chceme vytvoriť cestu použijeme príkaz `\pgfpath` a zakaždým, keď je tento príkaz volaný je cesta v nejakom smere rozšírená. Keď je cesta kompletne vytvorená je volaný príkaz `\pgfusepath` a podľa toho ako chceme objekt upraviť predávame tomuto príkazu parametre.



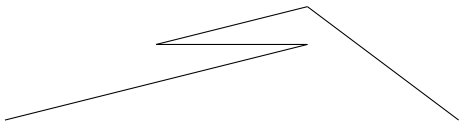
Obr. 1: Vykreslenie priamky

Na vykreslenie jednoduchej priamky nám stačí zadefinovať príkaz `\pgfline`, ktorý je ekvivalentný ku príkazu na konštrukciu cesty. V zátvorkách sa uvádza dvojica vektorov `x` a `y`, teda počiatočných a koncových bodov priamky, ktoré v tomto prípade začiatok predstavujú body(0,0) a koniec (3,2).

```
\begin {pgfpicture}
   \pgfline{\pgfxy(0,0)}{\pgfxy(3,2)};
\end {pgfpicture}
```

Výpis 5: Priamka

Ďalší príklad uvádza krivku, ktorá je zložená z viacerých bodov, kde začiatok je v bodoch (0,0) a koniec je v bodoch(6,0). Prepojenie medzi jednotlivými priamkami je zaistené bodom, ktorým jedna končí, druhá začína.

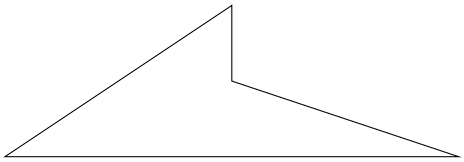


Obr. 2: Vykreslenie krivky

```
\begin {pgfpicture}
    \pgfline{\pgfxy(0,0)}{\pgfxy(4,1)}
    \pgfline{\pgfxy(4,1)}{\pgfxy(2,1)}
    \pgfline{\pgfxy(2,1)}{\pgfxy(4,1.5)}
    \pgfline{\pgfxy(4,1.5)}{\pgfxy(6,0)}
\end {pgfpicture}
```

Výpis 6: Vykreslenie krivky

Príkaz `\pgfclosestrokeclip` v našom príklade ukončil objekt, ale príkaz môže slúžiť tiež pre vyplnenie alebo pretínanie s inou cestou.



Obr. 3: Vykreslenie spojitého mnohouholníka

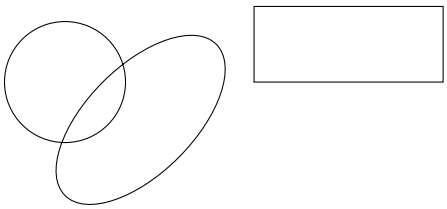
```
\begin{pgfpicture}
        \pgfmoveto{\pgfxy(0,0)}
       \pgflineto{\pgfxy(3,2)}
        \pgflineto{\pgfxy(3,1)}
       \pgflineto{\pgfxy(6,0)}
       \pgfclosestrokeclip
 \end{pgfpicture}
```

Výpis 7: Vykreslenie spojitého mnohouholníka

1.4 Preddefinované útvary a mriežky

Vytvárať tvary pomocou ciest je užitočné najmä v prípade, keď chce mať užívateľ definované tieto tvary podľa seba. V PGF však existujú už preddefinované tvary, ktoré uľahčujú užívateľovi prácu s vykresľovaním základných objektov.

- **Elipsa**: `\pgfpathellipse{<centrum>}{<prvá os>}{<druhá os>}`; pričom parametre príkazu sú vektory tvorené `{<centrum>}` a `{<prvá os>},{<druhá os>}`
- **Kruh**: `\pgfpathcircle{center}{radius}`;
- **Obdĺžnik**:`\pgfpathrectangle{roh}{diagonálny vektor}`; vykreslí obdĺžnik, ktorého cesta je udaná parametrom `\roh` a opačný roh je výsledkom `\roh+diagonálny vektor`



Obr. 4: Vykreslenie kružnice, elipsy a obdĺžnika

```
\pgfpathrectangle{\pgfpoint{2.5cm}{0.5cm}}{\pgfpoint{2.5cm}{1cm}}
\pgfpathellipse{\pgfpoint{1cm}{0cm}}
               {\pgfpoint{1cm}{1cm}}
                \pgfpoint{-0.5cm}{0.5cm}}
             \pgfpathcircle{\pgfpoint{0cm}{0.5cm}}{0.8cm}
\pgfusepath{draw}
```

Výpis 8: Výpis z vykreslenia objektov

Pokiaľ sme si skonštruovali objekt môžeme ho pomocou príkazov `\pgfstroke` a `\pgffill` vyplniť farbou. Farba výplne nemusí korešpondovať s farbou periférie objektov pričom toto nastavenie sa vykonáva pomocou príkazu `\color` a takéto nastavenie farby platí pre celú skupinu objektov.

Obr. 5: Vykreslenie vyplnenej kružnice

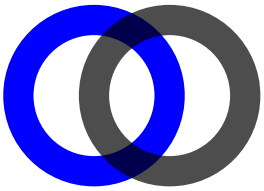
```
  \pgfsetlinewidth{1pt}
        \color{red}
        \pgfsetstrokecolor{black}
        \pgfpathcircle{\pgfpoint{1cm}{0cm}}{3mm} \pgfusepath{fill,stroke}
        \pgfpathcircle{\pgfpoint{2cm}{0cm}}{3mm} \pgfusepath{fill,stroke}
   \end{pgfpicture}
```

Výpis 9: Nastavenie farby

Príkazom `\pgfsetstrokeopacity{<hodnota>}` nastaví priehľadnosť periférií, pričom táto hodnota, by sa mala pohybovať v rozmedzí od 0 po 1, kde 0 predstavuje plnú nepriehľadnosť, 1 plnú priehľadnosť a hodnota 0.5 spôsobí, že objekty budú vyfarbené v polopriehľadnom odtieni farby.

```
\pgfsetlinewidth{4mm}
  \color{blue}
  \pgfpathcircle{\pgfpoint{0cm}{0cm}}{10mm} \pgfusepath{stroke}
  \color{black}
  \pgfsetstrokeopacity{0.7}
  \pgfpathcircle{\pgfpoint{1cm}{0cm}}{10mm} \pgfusepath{stroke}
```

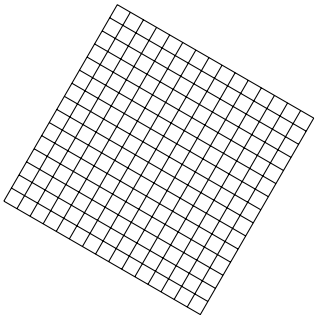
Výpis 10: Nastavenie priehľadnosti objektu



Obr. 6: Vykreslenie vyplnenej kružnice

Ďalším vykreslením je mriežka, ktorá býva obvykle rozširovaná (veľkým) množstvom častí, ktoré sú pridávané do ciest `\path`. Každá časť pozostáva z jednotného horizontálneho a vertikálneho segmentu úsečiek.
Kľúče, ktoré ovplyvňujú mriežku:

- `/pgf/stepx=<dimension>`popisuje horizontálne krokovanie;
- `/pgf/stepy=<dimension>`, popisuje vertikálne krokovanie;
- `/pgf/step=<vektor>`, nastavuje horizontálne krokovanie na x-ovú súradnicu a vertikálne na y-ovú súradnicu;



Obr. 7: Príklad vykreslenia transformovanej mriežka

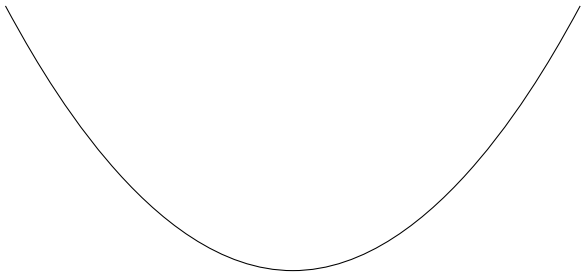
```
\begin{pgfpicture}
  \pgftransformrotate{-30}
  \pgfpathgrid[stepx=2mm,stepy=2mm]{\pgfpoint{0mm}{0mm}}{\pgfpoint{30mm}{30
      mm}}
  \pgfusepath{stroke}
 \end{pgfpicture}
```

Výpis 11: Príklad vykreslenia transformovanej mriežky

Transformácia mriežky je nastavená príkazom `\pgftransformrotate{-30}` a odstup jednotlivých buniek cez `\stepx=2mm` a `\stepy=2mm`.
Veľkosť objektu je udaná v mm `\pgfpoint{30mm}{30mm}`.

1.4.1 Vykresľovanie paraboly, sínusovej a kosínusovej funkcie

Pre vykreslenie základnej paraboly potrebujeme vektor lomu a koncový vektor. Príkaz na jej vykreslenie je v tvare: `\pgfpathparabola{<vektor lomu>}{koncový vektor}`.

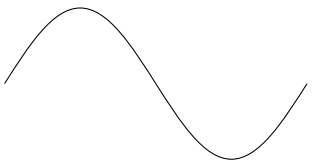


Obr. 8: Vykreslenie paraboly

```
\begin{pgfpicture}
          \pgfpathmoveto{\pgfpoint{2cm}{2cm}}
          \pgfpathparabola{\pgfpoint{3.8cm}{-3.5cm}}{\pgfpoint{3.8cm}{3.5cm}}
\pgfusepath{stroke}
\end{pgfpicture}
```

Výpis 12: Vykreslenie paraboly

Vykresľovanie funkcie sínus a kosínus je možné aj nastavením cez príkaz `\pgfpathsine` a `\pgfpathcosine`, nie len použitím balíčka PGFplots.
Zadefinovaný príkaz `\pgfpathsine{vektor}` prepája sínusovú krivku v intervale $[0, \frac{\pi}{2}]$. Sínusová krivka je natiahnutá alebo stlačená tak, aby krivka začínala v aktuálnom bode a končila v rovnakom bode + <vektor>. V prípade kosínusovej funkcie je príkaz `\pgfpathcosine{vektor}`. Použitím tejto série príkazov môžeme vytvoriť celú sínusovú či kosínusovú krivku. Toto vykreslenie je však pracné, preto sa odporúča skôr využitie PGFplots balíčka.



Obr. 9: Príklad z vykreslenia funkcie

```
\begin{pgfpicture}
      \pgfpathmoveto{\pgfpoint{0cm}{0cm}}
      \pgfpathsine{\pgfpoint{1cm}{1cm}}
      \pgfpathcosine{\pgfpoint{1cm}{-1cm}}
      \pgfpathsine{\pgfpoint{1cm}{-1cm}}
      \pgfpathcosine{\pgfpoint{1cm}{1cm}}
      \pgfusepath{stroke}
\end{pgfpicture}
```

Výpis 13: Výpis z vykreslenia funkcie

1.5 PGFplots

```
\usepackage{pgfplots} % LaTeX{}
```

PGFplots balík umožňuje vytvárať početné druhy grafov od lineárnych cez logaritmické, ale aj bodové či stĺpcové diagramy a tiež rôzne druhy značených ôs. Cez kľúče je možné nastaviť grafom označenia a legendy v prípade viacpočetných ôs. Hlavnou úlohou balíka PFGPlots je zjednodušiť vytváranie funkcií a grafov vo vysokej kvalite, vytvoriť konzistenciu dokumentu typu a fontu písma či umožniť priame použitie TEX matematický režim v popisoch osí.[4]

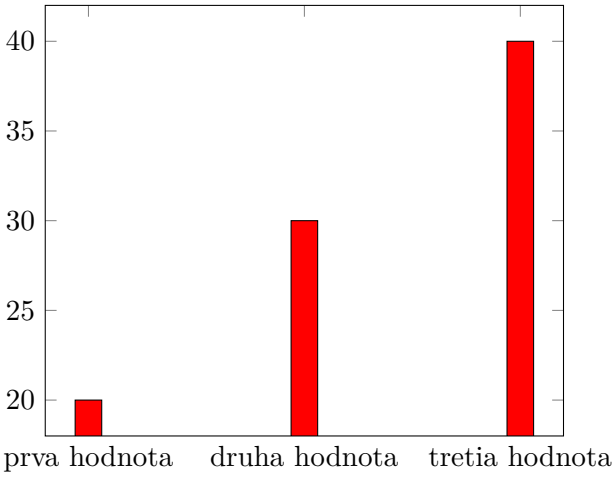
Balík tvorený dvoma zložkami:

- zložka pre vykresľovanie grafov;
- zložka pre vykresľovanie, formátovanie a spracovávanie numerických tabuliek;[5]

Graf v nasledujúcom príklade pozostáva z x-ovej osi, ktoré reprezentujú `{prva hodnota}` odkazujúca na číslo 20, `{druhá hodnota}` na číslo 30, `{tretia hodnota}` na 40.

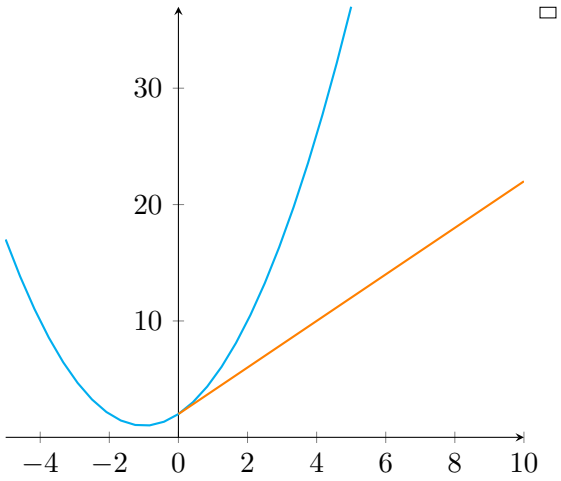
```
\begin{axis}
[symbolic x coords={prva hodnota, druha hodnota, tretia hodnota},xtick=data]
    \addplot[ybar,fill=red] coordinates
      {
        (prva hodnota,20)
        (druha hodnota,30)
        (tretia hodnota,40)
    };
\end{axis}
```

Výpis 14: Graf



Obr. 10: Graf hodnôt

V nasledujúcom príklade sme si skonštruovali kvadratickú a lineárnu funkciu použitím PGFplots balíčka podľa nasledujúcich predpisov: $f(x) = x^2 + 2 * x + 2; g(x) = 2 * x + 2$



Obr. 11: Vykreslenie kvadratickej a lineárnej funkcie

```
\begin{axis}[ymin = 0, axis x line = bottom, axis y line=middle, legend pos=
    outer north east]
\addplot [domain=-5:5,cyan, thick]{x^2 + 2*x + 2};
\addplot [domain=0:10,orange,thick]{2*x + 2};
\legend{$f(x)$, $g(x)$}
```

Výpis 15: Výpis kvadratickej a lineárnej funkcie

[5]nachádza sa v samostatnom balíku **pgfplotstable**

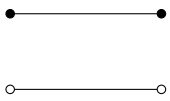
2 Tvorba obvodov

Pri tvorbe elektrických a logických obvodov je dostupných niekoľko knižníc, ktoré vzájomne spolupracujú. Tieto knižnice boli vytvorené pre užívateľa za účelom uľahčenia práce pri vykresľovaní obvodov a ich prednosťou je, že si stále zachovávajú vysokú kvalitu znázornenia.
Základnou knižnicou na vrstve Ti*k*Z je `\circuit`[2], pracujúca v prostredí `\{tikzpicture}`, ktorá bola inšpirovaná knižnicou `\circuitikz`[8].
Jednotlivé symboly sú definované rovnako ako v americkej norme aj v európskej a tie zahrňujeme do preambule podľa toho, ktorú z nich momentálne používame.

```
\usetikzlibrary{circuits.ee.US} % americký štandard
\usetikzlibrary{circuits.ee.IEC}%európsky štandard
```

2.1 Elektrické obvody

Elektrický obvod typicky pozostáva z niekoľkých elektrických prvkov, ktoré sú prepojené vodičmi.



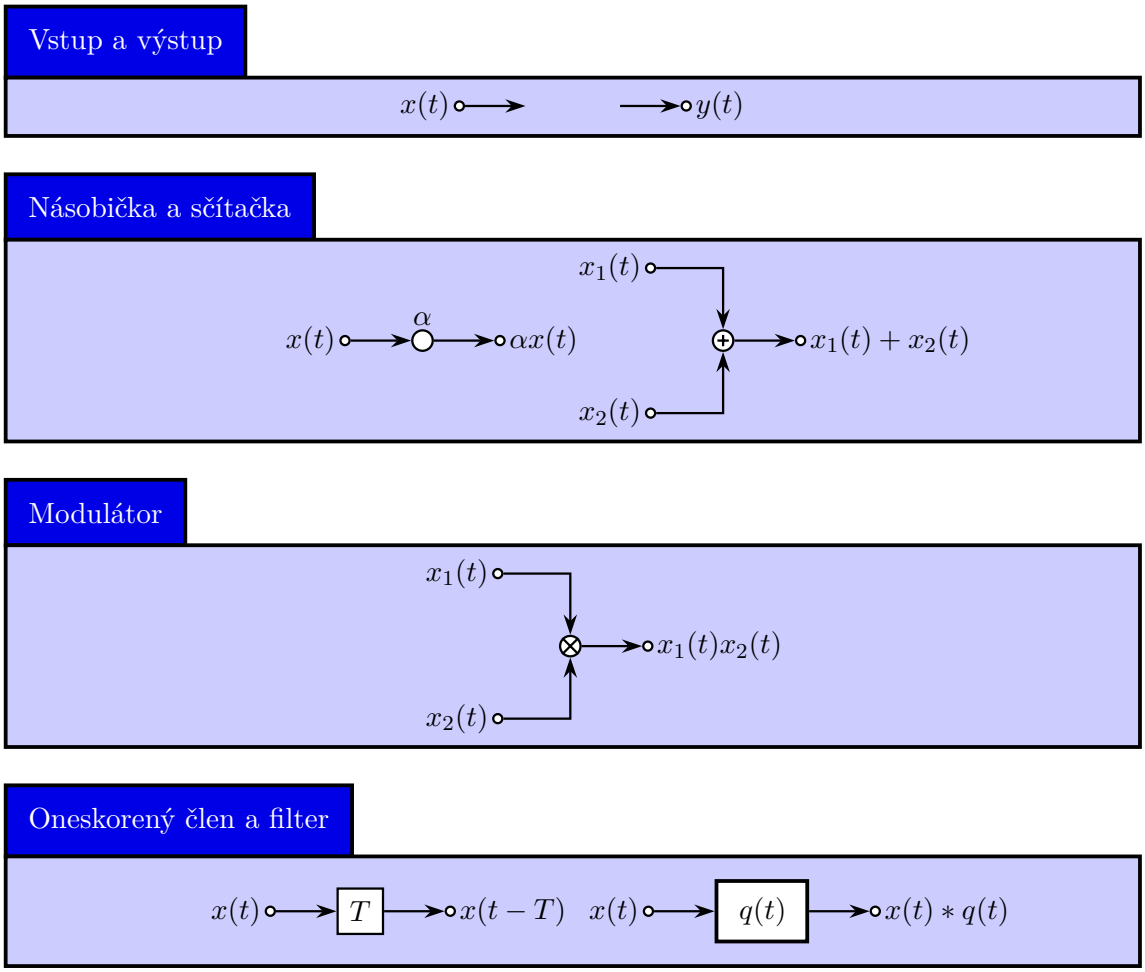
Obr. 12: Vodiče s použitím CircuitTi*k*Z

```
\begin{center}
\begin{circuitikz}
  \draw (-1,0) to[short,o-o] (1,0);
   \draw (-1,1) to[short,*-*] (1,1);
\end{circuitikz}
\end{center}
```

Výpis 16: Vodiče použitím CircuitTi*k*Z

2.2 Signalflow knižnica

Karlheinz Ochs je autorom knižnice, ktorá definuje rad stavebných blokov, ktoré môžu byť použité pre vykresľovanie diagramov toku signálu bežne použiteľných v digitálnom spracovaní signálov. Knižnica definuje niekoľko uzlových tvarov ako napríklad násobičku, terminál či tvary. Definuje aj prostredie `\tikzgrid`, ktoré umožňuje usporiadať uzly do poľa. Okrem toho aj niekoľko špecifických štýlov pre reálne či komplexné prvky[9].



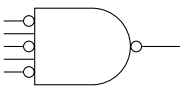
Obr. 13: Signalflow knižnica

2.3 Logické obvody

```
\usepgflibrary{circuits.logic.US} % LaTeX {}, \TeX{}, PGF
\usepgflibrary{circuits.logic.IEC} % LaTeX {}, \TeX{}, PGF
\usepgflibrary{circuits.logic.CDH} % LaTeX {}, \TeX{}, PGF
```

Logickými obvodmi sú obvody, ktoré sú tvorené logickými členmi - hradlami. Svojimi hradlami takmer identická s americkým štandardom okrem hradiel AND a NAND je CDH knižnica.

Nasledujúci príklad vykresľuje hradlo NAND podľa amerického štandardu a jeho veľkosť je udaná parametrom `[minimum height=0.75cm]`.
Veľkosť hradla je ovplyvnená parametrami: premenná n, ktorá definuje počet vstupov, ktoré do hradla vstupujú, polomer kružníc r, ktorý nastavujeme jednotlivým vstupom a s udávajúce vzdialenosť medzi stredmi vstupov.
Definovanie viacerých vstupov je uskutočnené použitím príkazu:`\logic gate inputs=inini`, v ktorom i predstavuje invertovaný vstup a n predstavuje normálny vstup. Invertovaný vstup je definovaný ako vstup, ktorý vykreslí kruh na logické hradlo.

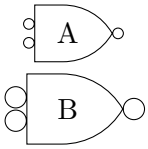


Obr. 14: NAND hradlo - US štandard

```
\begin{tikzpicture}[minimum height=0.75cm]
\node[nand gate US, draw,logic gate inputs=inini] (A) {};
\foreach \a in {1,...,5}
\draw (A.input \a -| -1,0) -- (A.input \a);
\draw (A.output) -- ([xshift=0.5cm]A.output);
```

Výpis 17: NAND hradlo - US štandard

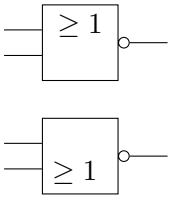
V nasledujúcom príklade sme definovali hradlu dva invertované vstupy, ktorým sme staticky nastavili veľkosť polomeru v bodoch, kde veľkosť hradla NAND A je $2pt$ a veľkosť hradla B je $4pt$.



```
\begin{tikzpicture}[minimum height=0.75cm]
\tikzset{every node/.style={shape=nand gate CDH, draw, logic gate inputs=ii}}
\node[logic gate inverted radius=2pt] {A};
\node[logic gate inverted radius=4pt] at (0,-1) {B};
```

Výpis 18: NAND hradlo CDH štandard

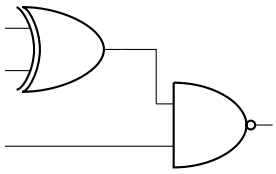
Hradlo v nasledujúcom príklade je nastavené na veľkosť `[minimum size=1cm]` v IEC štandarde. Vykreslenie hradla A sa uskutočňuje na pozícií (0,0). Rovnako sme obom hradlám nastavili popis, kde je táto vlastnosť udaná parametrom `align=bottom right` a môže nadobúdať pozície `align=top,bottom,left,right`.



Obr. 15: NAND hradlo - IEC štandard

```
\begin{tikzpicture}[minimum size=1cm, use IEC style logic gates]
\tikzset{every node/.style={nor gate, draw}}
\node (A) at (0,1.5) {};
\node [logic gate symbol align={bottom, right}] (B) at (0,0) {};
\foreach \g in {A, B}{
\foreach \i in {1,2}
\draw ([xshift=-0.5cm]\g.input \i) -- (\g.input \i);
\draw (\g.output) -- ([xshift=0.5cm]\g.output);
}
```

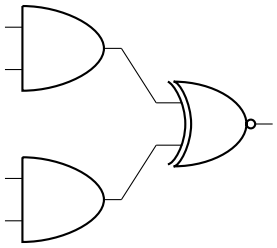
Výpis 19: NAND hradlo - IEC štandard



Obr. 16: Nastavenie vstupov v obvode

```
\begin{circuitikz}
  \draw (0,0) node[xor port](xor1) {}
     (2,-1) node[nand port] (nand1) {}
     (xor1.out) -| (nand1.in 1)
     (nand1.in 2) -- +(-2,0);
   \end{circuitikz}
```

Výpis 20: Nastavenie vstupov v obvode



Obr. 17: Nastavenie vstupov v obvode

```
\begin{circuitikz} \draw
      (0,2) node[and port] (and1) {}
      (0,0) node[and port] (and2) {}
      (2,1) node[xnor port] (xnor) {}
      (and1.out) -- (xnor.in 1)
      (and2.out) -- (xnor.in 2);
\end{circuitikz}
```

Výpis 21: Nastavenie viacerých vstupov

3 Tvorba uzlov a kotiev

```
\usepgfmodule{tvar}% LaTeX{},TeX{} a PGF
\usepgfmodule[tvar]% ConTeXt{} a PGF
```

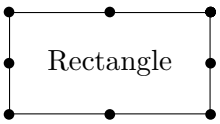
Je modul, zahrnutý v balíku PGF, ktorý definuje príkazy pre tvorbu uzlov a tvarov na tejto vrstve. Obyčajne býva automaticky načítaný PGF balíkom a manuálne ho načítavame len v prípade, že preambula obsahuje len `pgfcore`.

3.1 Uzly

Uzol predstavuje grafický objekt, ktorý je zvyčajne tvorený textový označením a prídavnými cestami, ktoré môžu byť vyplnené. Každý uzol má určitý tvar, ktorý môže byť preddefinovaný ako napr. obdĺžnik či kruh alebo si takýto tvar môže užívateľ definovať sám. Uzly poväčšine pozostávajú z `backgroundpath` a `foregroundpath` a textových štítkov. Najjednoduchšie uzly tvoria jeden textový štítok, komplikovanejšie uzly ako napríklad UML diagram obsahujú viaceré textové štítky.
Deklarácia uzla: `pgfnode{<tvar>}{<kotva>}{<meno>}{<textový štítok>}{<cesta>}`
Meno uzla nastavujeme na neskoršie odkazovanie naň. Pokiaľ definuje daný tvar cestu potom je možné tvaru nastaviť aj `backgroundpath` a `foregroundpath`.

3.2 Kotvy

Dôležitou vlastnosťou uzlov a tvarov sú kotvy tvoriace ich pozíciu v danom objekte. Pri vytváraní uzla definujeme kotvu ako pozíciu bodu, v ktorom táto kotva leží.



Obr. 18: Poloha uzlov v obdĺžniku

```
\begin{tikzpicture}
\pgfset{inner ysep=0.5cm}
\pgfset{inner xsep=0.5cm}
\pgfnode{rectangle}{center}{Rectangle}{x}{\pgfusepath{stroke}}
\pgfpathcircle{\pgfpointanchor{x}{north}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{south}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{east}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{west}}{2pt}
```

```
\pgfpathcircle{\pgfpointanchor{x}{north east}}{2pt}

\pgfpathcircle{\pgfpointanchor{x}{south west}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{south east}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{north east}}{2pt}
\pgfpathcircle{\pgfpointanchor{x}{north west}}{2pt}
\pgfusepath{fill}
```

Výpis 22: Poloha kotiev v uzle Rectangle

V príklade je vytvorený uzol, ktorý nadobúda tvar obdĺžnika. `pgfset{inner ysep=0.5}` nastavuje horizontálnu vnútornú vzdialenosť medzi textovým štítkom a `\backgroundpath` na hodnotu 0.5pt. Rovnako nastavená vertikálna vzdialenosť `pgfset{inner ysep=0.5}`. Príklad má vykresľovať postavenie kotiev v obdĺžniku na jednotlivých pozíciach vo veľkosti 2*pt*. Na polohy kotiev v tvare odkazuje `\pgfpointanchor` , ktorý vracia súradnicu danej kotvy v uzle.

4 Tvorba vlastných tvarov

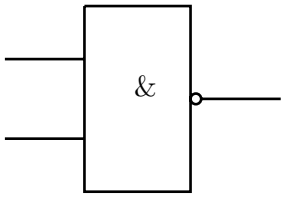
Vytváranie vlastných tvarov a blokov na základnej vrstve PGF nie je práve najjednoduchším procesom. Tvary by sa mali v závislostiach od okolností meniť a mali by byť skonštruované tak, aby ich vedel prekladač rýchlo preložiť.

Definovaný tvar by mal obsahovať:

- názov tvaru
- kód pre výpočet uložených kotiev a uložených rozmerov
- kód pre výpočet výpočet pozícií kotiev podľa uložených kotiev
- nepovinný kód pre vykresľovanie ciest v prostredí `\backgroundpath` a `\foregroundpath`
- nepovinný kód pre vykresľovanie objektov mimo týchto prostredí
- nepovinný zoznam uzlov

Deklarácia vlastného tvaru na vrstve PGF sa uskutočňuje príkazom:
`\pgfdeclareshape{<názov tvaru>}{<špecifikácia>}`, kde definujeme jeho názov a špecifikácia obsahuje TEX príkaz, ktorého úlohou je volať iné príkazy.
Príkaz `\nodeparts{<zoznam častí uzlov>}` deklaruje, ktoré časti tvoria uzly tvara. Príkaz mal obsahovať aj `\savedanchor{<prikazy><kod>}` definujúci uložené kotvy, kde argument by mal byť tvorený TEX makrom ako napríklad `\savedanchor{\centerpoint}`.
Podobný `\saveddimen{<prikaz><kod>}` je `\savedanchor`, ktorý nastavuje `\pgfpoint{x}{y}` na rozdiel od neho v tomto príkaze nastavujeme iba hodnotu x.

4.1 NAND hradlo



Obr. 19: Schematická značka hradla NAND

```
%deklaracia tvaru
\pgfdeclareshape{NAND}
{
  % kotvy zdedit z obdlznika (rectangle)
   \inheritsavedanchors[from=rectangle]
   \inheritanchor[from=rectangle]{center}
   \inheritanchor[from=rectangle]{north}
   \inheritanchor[from=rectangle]{south}
   \inheritanchor[from=rectangle]{east}
   \inheritanchor[from=rectangle]{west}
   \inheritanchor[from=rectangle]{north east}
   \inheritanchor[from=rectangle]{north west}
   \inheritanchor[from=rectangle]{south east}
   \inheritanchor[from=rectangle]{south west}
   \inheritanchorborder[from=rectangle]
```

Výpis 23: Deklarácia tvaru

Dedenie z iného tvaru sa vykonáva cez `\inheritsavedanchors[from={<nazov tvaru>}]`, ktorý využívame pokiaľ novo vytvorený tvar je len malou modifikáciou tvaru, ktorý v potrebujeme.
Dediť môžeme kotvy z viac ako jedného tvaru a to volaním tohto príkazu viackrát. Kód sa vykonáva v prostredí `\backgroundpath`, ktoré vykresľuje daný objekt pomocou ciest, ktoré sú upravované.

```
    \pgf@process{\northeast} % zaciname suradnicami z praveho horneho rohu
       tvaru
         \pgf@xa = \pgf@x \pgf@ya = \pgf@y % spocita pravy horny a lavy dolny
             roh obdlznika
         \pgf@xa = -0.6\pgf@xa %nastavenie suradnic
         \pgf@ya = 0.9\pgf@ya %nastavenie suradnic
         \pgf@xb = \pgf@x \pgf@yb = \pgf@y
         \advance\pgf@xb by -1.2\pgf@xb
         \advance\pgf@yb by -0.8\pgf@yb
         \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}} % presun
         \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@ya}} % vykreslenie stran
            obdlznika
         \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}
         \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@yb}}
```

```
            \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}
```

Výpis 24: Vykreslenie obdĺžnika

```
\pgf@process{\northeast}
            \pgf@xc = \pgf@x \pgf@yc = \pgf@y % vypocitat stred kruznice, pravej
                strane obdlznika v strede
            \pgf@yc = 0.55\pgf@yc
            \advance\pgf@xc by -1.2\pgf@xc %pozicia kruh
            \advance\pgf@xc by 2pt            % polomer bude 2pt
            \pgfpathcircle{\pgfpoint{\pgf@xc}{\pgf@yc}}{(2pt)} % vykreslenie
               kruhu
```

Výpis 25: Kruh

```
      %vykreslenie prvej ciary
      \pgf@process{\northeast}
            \pgf@xb = \pgf@x \pgf@yb = \pgf@y
            \pgf@xa = \pgf@x
            \advance\pgf@xb by -1.9\pgf@xb
            \advance\pgf@yb by -0.3\pgf@yb
            \pgf@ya = \pgf@yb
            \pgf@xa = -0.6\pgf@xa %velkost ciary
            \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
            \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}} % vlastna ciara
```

Výpis 26: Vykreslenie prvého vstupu

```
%vykreslenie 2 ciary
   \pgf@process{\northeast}
            \pgf@xb = \pgf@x \pgf@yb = \pgf@y
            \pgf@xa = \pgf@x
            \advance\pgf@xb by -1.9\pgf@xb
            \advance\pgf@yb by -0.6\pgf@yb

            \pgf@ya = \pgf@yb
            \pgf@xa = -0.6\pgf@xa
            \pgfpathmoveto{\pgfpoint{\pgf@xb}{\pgf@yb}}
            \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}} % vlastna ciara
```

Výpis 27: Vykreslenie druhého vstupu

```
    \pgf@process{\northeast}
        \pgf@xc = \pgf@x \pgf@yc = \pgf@y
        \pgf@yc = 0.55\pgf@yc%pozicia ciary k obdlzniku
        \advance\pgf@xc by -1.2\pgf@xc
        \advance\pgf@xc by 4pt                    % polomer bude 2pt, teda
           priemer je 4pt, o tolko ideme doprava so zaciatkom
        \pgf@xb = \pgf@xc \pgf@yb = \pgf@yc
        \advance\pgf@xb by 0.3\pgf@x
        \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}} % presun
        \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}} % vlastna ciara
\pgfusepath{draw}
```

Výpis 28: Vykreslenie vstupnej čiary

4.2 Vytvorenie pozície v hradle

Logické hradlo NAND z predchádzajúcej ukážky je vytvorené dedením z obdĺžnika a jednotlivých čiar, ktoré tvoria vstupy a výstup hradla. Rovnako je tvorený aj textovým poľom, ktoré mu vytvorilo popis definovaný za pomoci kotvy `\Znak` a s jej presne udanými parametrami. Na takúto kotvu sme sa neskôr v prostredí `\tikzpicture` odkazovali.

```
\anchor{Znak}
{% kotva s nastavenymi poziciami
   \northeast
   \pgf@xb = \pgf@x \pgf@yb = \pgf@y
   \advance\pgf@xb by -1.35\pgf@xb
   \advance\pgf@yb by -0.4\pgf@yb
   \advance\pgf@xb by -2pt
   \pgfpoint{\pgf@xb}{\pgf@yb}
}
```

Výpis 29: Kotva v hradle NAND

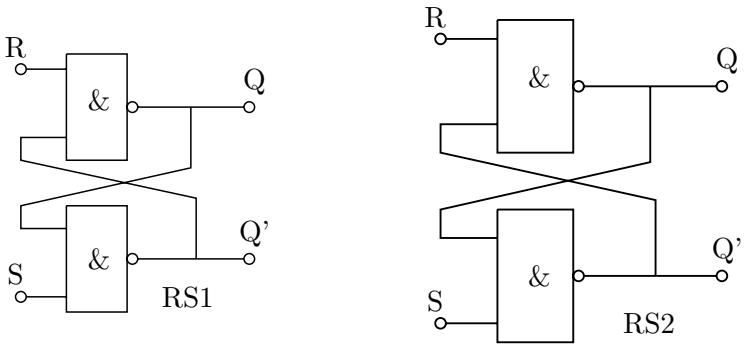
```
\begin{tikzpicture}
\draw (0, 1) node[NAND, minimum size=7cm](NAND) {};
\node at (NAND.Znak) {\&}; %uzol,ktory odkazuje na poziciu kotvy v objekte a
   ktory ma nastaveny parameter \&
```

Výpis 30: Vytvorenie pozície kotvy

4.3 Vytvorenie RS klopného obvodu

Sekvenčné obvody nazývané tiež sekvenčné automaty sú digitálne elektronické obvody, pri ktorých je stav výstupov podmienený aktuálnym stavom vstupov, ale rovnako aj minulým stavom vstupov. Tieto obvody majú pamäťové vlastnosti. Najjednoduchším sekvenčným obvodom je preklápací obvod nazývaný aj klopný obvod. Klopné obvody tiež R-S obvody(angl. SR lanch) majú vo svojom základnom prevedení dva vstupy: R(angl. Reset - nulovanie) a S(angl. Set - nastavenie). Hodnota sa ukladá na výstup Q, kde existuje aj negovaný výstup Q.

V nasledujúcom príklade sme na vykreslenie obvodu využili predošlé vytvorené hradlo.



Obr. 20: RS obvod so zmenou tvaru

```
\anchor{R}% kotva na vstup RS - kruh R
{
   \northeast
   \pgf@xb = \pgf@x \pgf@yb = \pgf@y
   \advance\pgf@xb by -1.9\pgf@xb
   \advance\pgf@yb by -0.2\pgf@yb
   \advance\pgf@xb by -2pt
   \pgfpoint{\pgf@xb}{\pgf@yb}
}
\anchor{S}% kotva na vstup RS - kruh S
{
   \northeast
   \pgf@xa = \pgf@x \pgf@ya = \pgf@y
   \advance\pgf@xa by -1.9\pgf@xa
   \advance\pgf@ya by -1.7\pgf@ya
   \advance\pgf@xa by -2pt
   \pgfpoint{\pgf@xa}{\pgf@ya}
}
```

```
\anchor{Q}% kotva na vystup - kruh Q
{
   \northeast
   \pgf@xc = \pgf@x \pgf@yc = \pgf@y
   \advance\pgf@xc by -1.2\pgf@xc
   \advance\pgf@xc by 0.7\pgf@x
   \advance\pgf@xc by 8pt
   \pgf@yc = 0.55\pgf@yc
   \pgfpoint{\pgf@xc}{\pgf@yc}
}
%
\anchor{NQ}% kotva na vystup - kruh Q'
{
   \northeast
   \pgf@xc = \pgf@x \pgf@yc = \pgf@y
   \advance\pgf@yc by -1.45\pgf@yc
   \advance\pgf@xc by -1.2\pgf@xc
   \advance\pgf@xc by 8pt
   \advance\pgf@xc by 0.7\pgf@x
   \pgfpoint{\pgf@xc}{\pgf@yc}
}
```

Výpis 31: Nastavenie pozícií pre jednotlivé vstupy pomocou kotiev

```
\anchor{text}
{
   \northeast
      \pgf@xc = \pgf@x \pgf@yc = \pgf@y
      \advance\pgf@yc by -1\pgf@yc
      \advance\pgf@xc by -1\pgf@xc
      \advance\pgf@yc by -0.7\pgf@y
      \advance\pgf@xc by 0.2\pgf@x
      \pgfpoint{\pgf@xc}{\pgf@yc}
}
```

Výpis 32: Kotva pre nastavenie popisu tvaru

Pri vytváraní obvodu vychádzame z prechádzajúceho hradla NAND, ktoré sme si definovali. Toto hradlo je doplnené o ďalší obdĺžnik, ktorý vzájomne s predošlým pomocou `\pgfpathlineto` prepájame.

```
\pgf@process{\northeast}
        \pgf@xa = \pgf@x \pgf@ya = \pgf@y %obdobne spocitame pravy a lavy
            horny roh obdlznik
        \pgf@xa = -0.6\pgf@xa %nastavene jednotlive pozicie obdlznika
        \pgf@ya = -0.1\pgf@ya
        \pgf@xb = \pgf@x \pgf@yb = \pgf@y
        \advance\pgf@xb by -1.2\pgf@xb
        \advance\pgf@yb by -1.8\pgf@yb
        \pgfpathmoveto{\pgfpoint{\pgf@xa}{\pgf@ya}} % presun
        \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@ya}} % nasleduje vykreslenie
        \pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}
        \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@yb}}
        \pgfpathlineto{\pgfpoint{\pgf@xa}{\pgf@ya}}
```

Výpis 33: Obdĺžnik S

```
    \pgf@process{\northeast} %nastavenie kruhu v pravej strane obdlznika v
        polovici vysky
        \pgf@xc = \pgf@x \pgf@yc = \pgf@y
        \advance\pgf@yc by -1.45\pgf@yc
        \advance\pgf@xc by -1.2\pgf@xc
        \advance\pgf@xc by 2pt %nastavenie velkosti kruhu
        \pgfpathcircle{\pgfpoint{\pgf@xc}{\pgf@yc}}{(2pt)}%vykreslenie tvaru
```

Výpis 34: Kruh na pravej strane obdĺžnika S

```
    \pgf@process{\northeast}
        \pgf@xa = \pgf@x \pgf@ya = \pgf@y
        \advance\pgf@xa by -1.9\pgf@xa
        \advance\pgf@ya by -1.7\pgf@ya
        \pgfpathcircle{\pgfpoint{\pgf@xa}{\pgf@ya}}{(2pt)}
```

Výpis 35: Kruh na dolnej strane obdĺžnika S

```
    \pgf@process{\northeast}
        \pgf@xc = \pgf@x \pgf@yc = \pgf@y
        \advance\pgf@yc by -1.45\pgf@yc
        \advance\pgf@xc by -1.2\pgf@xc
        \advance\pgf@xc by 4pt
        \pgf@xb = \pgf@xc \pgf@yb = \pgf@yc
        \advance\pgf@xb by 0.7\pgf@x
```

```
\pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}}
\pgfpathlineto{\pgfpoint{\pgf@xb}{\pgf@yb}}
```

Výpis 36: Čiara na pravej strane obdĺžnika S

```
\pgf@process{\northeast}
    \pgf@xc = \pgf@x \pgf@yc = \pgf@y
    \advance\pgf@yc by -1.45\pgf@yc
    \advance\pgf@xc by -1.2\pgf@xc
    \advance\pgf@xc by 6pt
    \advance\pgf@xc by 0.7\pgf@x
    \pgfpathcircle{\pgfpoint{\pgf@xc}{\pgf@yc}}{(2pt)}
```

Výpis 37: Kruh Q

```
\pgf@process{\northeast}
    \pgf@xa = \pgf@x \pgf@ya = \pgf@y

    \pgf@xc = \pgf@xa \pgf@yc = \pgf@ya % stred S output line
    \advance\pgf@yc by -1.45\pgf@yc
    \advance\pgf@xc by -1.2\pgf@xc
    \advance\pgf@xc by 6pt
    \advance\pgf@xc by 0.35\pgf@xa
    \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}}

    \pgf@yc = -0.1\pgf@ya %nastavenie lomu
    \advance\pgf@yc +0.05\pgf@ya % velkost priamky kolmej na S output in
    \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}  % vykreslenie

    \advance\pgf@yc +0.25\pgf@ya
    \pgf@xc = \pgf@xa
    \advance\pgf@xc by -1.9\pgf@xa
    \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}} % nastavenie a
       vykreslenie sikmej priamky

    \advance\pgf@yc +0.15\pgf@ya
    \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}%ciara zasa kusok dohora
       - na vysku spodnej "tretiny" strany R obdlznika

    \pgf@xc = \pgf@xa
    \pgf@xc = -0.6\pgf@xc
```

```
        \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}} % vodorovna ciara k
            lavej zvislej strane R obdlznika
```

Výpis 38: Nastavenie lomenej cesty z S output line

```
    % Lomena ciara z polovice R output line na vrchnu tretinu lavej strany
        S obdlznika
    \pgf@process{\northeast}
        \pgf@xa = \pgf@x \pgf@ya = \pgf@y
        \pgf@xc = \pgf@xa \pgf@yc = \pgf@ya
        \pgf@yc = 0.55\pgf@ya
        \advance\pgf@xc by -1.2\pgf@xc
        \advance\pgf@xc by 4pt
        \advance\pgf@xc by 0.35\pgf@xa
        \pgfpathmoveto{\pgfpoint{\pgf@xc}{\pgf@yc}} %stred R output line
        \pgf@yc = +0.1\pgf@ya
        \advance\pgf@yc +0.05\pgf@ya
        \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}}
        \advance\pgf@yc -0.25\pgf@ya
        \pgf@xc = \pgf@xa
        \advance\pgf@xc by -1.9\pgf@xa
        \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}} % ciara sikmo nadol
        \advance\pgf@yc -0.15\pgf@ya
        \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}} %priamka na dol - do
            vysky vrchnej "tretiny" strany S obdlznika
        \pgf@xc = \pgf@xa
        \pgf@xc = -0.6\pgf@xc
        \pgfpathlineto{\pgfpoint{\pgf@xc}{\pgf@yc}} % vodorovna ciara k
            lavej zvislej strane S obdlznika
```

Výpis 39: Nastavenie lomenej cesty z R output line

Rovnako ako v NAND hradle tak aj vo vykreslení RS obvodu sme využili kotvy, definujúce popisy hradiel.

```
\anchor{Zn}[label=Latex,caption=Nastavenie kotiev pre hradla R a S]
{%pozicia kotvy pre znak v hradle R
    \northeast
    \pgf@xb = \pgf@x \pgf@yb = \pgf@y
    \advance\pgf@xb by -1.35\pgf@xb
    \advance\pgf@yb by -0.4\pgf@yb
```

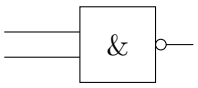
```
    \advance\pgf@xb by -2pt
    \pgfpoint{\pgf@xb}{\pgf@yb}
  }

    \anchor{Zn1}
  {%pozicia kotvy pre znak v hradle s
    \northeast
    \pgf@xb = \pgf@x \pgf@yb = \pgf@y
    \advance\pgf@xb by -1.35\pgf@xb
    \advance\pgf@yb by -1.45\pgf@yb
    \advance\pgf@xb by -2pt
    \pgfpoint{\pgf@xb}{\pgf@yb}
  }
```

```
  \begin{tikzpicture}
  \draw (0,12) node[RSobvod,minimum size=4cm] (RS1) {};%Hradlu sme definovali
      konstatnu velkost
    \node at (RS1.text) {RS1};
    \node at (RS1.Zn) {\&};
    \node at (RS1.Zn1) {\&};
  \draw (6, 12) node[RSobvod,minimum size=5cm] (RS2) {};%Hradlu sme definovali
       konstatnu velkost
  \node at (RS2.text) {RS2};
    \node at (RS1.text) {RS2};
    \node at (RS2.Zn) {\&};
    \node at (RS2.Zn1) {\&};
\end{tikzpicture}
```

Výpis 40: Volanie objektu v prostredí tikzpicture

4.4 Zmena počtu vstupov



Obr. 21: Vykreslenie hradla s konštantným počtom vstupov

```
\begin{tikzpicture}
```

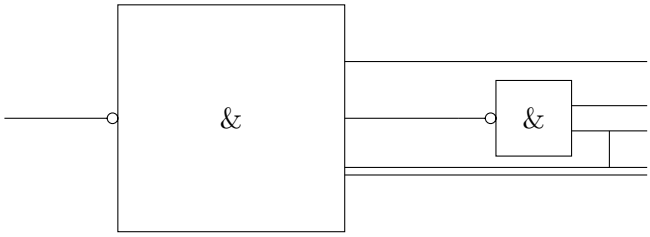
```
\draw (6, 2) node[box, minimum size=1cm,box pins = 2, label=center:\&, rotate
    =360] (b2) {\&};
\coordinate (pm1) at (b2.pin 1);
\coordinate (pm2) at (b2.pin 2);

\draw ($(pm1)$) -- ($(pm1) + (-1, 0)$);
\draw ($(pm2)$) -- ($(pm2) + (-1, 0)$);

\end{tikzpicture}
```

Výpis 41: Volanie objektu v prostredí tikzpicture

V nasledujúcom príklade sme vytvorili schému prepojenia 2 hradiel NAND, z ktorých jedno hradlo má 4 vstupy a druhé 2. Na prepojenie sme použili príkaz `\coordinate`, ktorý sa používa na odkazovanie na špeciálny bod vo vnútri alebo na okraji skôr definovaného uzla.



Obr. 22: Prepojenie hradla so 4 vstupmi s hradlom s 2 vstupmi

```
\draw (2, 2) node[box, minimum size=3cm, box pins = 3, label=center:\&, rotate
    =180] (b1) {\&1};%uzol je tvoreny popisom, 3 vstupmi
\draw (6, 2) node[box, minimum size=1cm,box pins = 2, label=center:\&, rotate
    =360] (b2) {\&2};%uzol je tvoreny popisom, 2 vstupmi
\coordinate (p1) at (b1.pin 1);%nastavenie vstupu pre p1 do pin 1
\coordinate (p3) at (b1.pin 3);%nastavenie vstupu pre p3 do pin 3
\coordinate (pm1) at (b2.pin 1);
\coordinate (pm2) at (b2.pin 2);

\draw (b1.pin 2) -- (b2.output);
\draw ($(p1)$) -- ($(p1) + (4, 0)$);%vykreslenie vystupu p1
\draw ($(p3)$) -- ($(p3) + (4, 0)$);%vykreslenie vystupu p3

\draw ($(pm1)$) -- ($(pm1) + (1, 0)$);%vykreslenie vystupu \&2,pin1
\draw ($(pm2)$) -- ($(pm2) + (1, 0)$);%vykreslenie vystupu \&2,pin2
```

```
\draw($(p3) + (0, +0.1)$) -- ($(p3) + (4, 0.1)$);
\draw($(p3) + (3.5, +0.1)$) -- ($(pm2) + (0.5, 0 )$);
```

Výpis 42: Vykreslenie hradla so 4 a 2 vstupmi

Záver

Cieľom bakalárskej práce bolo vytvoriť manuál v typografickom systéme LaTeX na úrovni PGF k sadzbe ilustrácií z prostredia elektrotechniky. Pred začiatkom písania tejto práce som so systémom LaTeX mala skúsenosti z vyučovania, avšak nie s prácou s balíkom PGF.
Práca s týmto balíkom nepatrí medzi najjednoduchšie a tento manuál má slúžiť ako sprievodca užívateľovi pri vytváraní objektov na tejto vrstve pričom jednotlivé kapitoly rozdelené podľa náročnosti od vytvorenia najjednoduchšieho tvaru až po komplikovanejší obvod.
Rovnako sa tento manuál zameriava na balíčky, ktoré spolupracujú s Ti*k*Z &PGF, a ktoré mnohokrát uľahčujú prácu menej skúseným užívateľom. Ku práci som využívala manuál[2] a bakalársku prácu Tomáša Pavlorka, avšak k niektorým zložitejším vykresleniam som dospela aj za pomoci ukážkových príkladov, ktoré riešili iní skúsenejší užívatelia.
Aj keď zo začiatku nebolo jednoduché orientovať sa v tomto balíku neskôr mi práca s balíkom PGF ukázala veľa možností a funkcií, ktoré tento balík ponúka pre sadzbu technických ilustrácií a textu.

Literatura

[1] Pavlorek, Tomáš. *Použití balíčku TikZ pro sazbu LaTeXových knih s obrázky*, 2013.

[2] Tantau, Till. *TikZ &PGF* , 2015.
http://mirrors.nic.cz/tex-archive/graphics/pgf/base/doc/pgfmanual.pdf

[3] *http://kalendar.beda.cz[online]*
http://kalendar.beda.cz/iso-8601/

[4] Feuersanger, Christian. *Manual for package PGFPLOTS*, 2015.
http://pgfplots.sourceforge.net/pgfplots.pdf

[5] Feuersanger, Christian. *Manual for package PGFPLOTSTABLE*, 2011.
ftp://ftp.fu-berlin.de/tex/CTAN/graphics/pgf/contrib/pgfplots/doc/pgfplots.pdf

[6] Carlisle, David. *The keyval package*, 2014.
http://tug.ctan.org/macros/latex/required/graphics/keyval.pdf

[7] Adriaens,Hendri. *The xkeyval package*, 2014.
http://mirrors.nic.cz/tex-archive/macros/latex/contrib/xkeyval/xkeyval.pdf

[8] Redaelli, Massimo. *CircuiTikZ*, 2016.
http://get-software.net/graphics/pgf/contrib/circuitikz/doc/circuitikzmanual.pdf

[9] Ochs, Karlheinz. *Signal flow building blocks [online]*, 2007.
http://www.texample.net/tikz/examples/signal-flow-building-blocks/