STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor SOČ: 2. FYZIKA

Simulace přechodových dějů v RC článku

Vypracoval: Martin Novák

Škola: Střední škola spojů a informatiky, Tábor, Bydlinského 2474

Kraj: Jihočeský

Konzultant: Ing. Dana Almášiová

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v seznamu vloženém v práci SOČ. Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V ……dne…… ……

Podpis

Děkuji paní Ing. Daně Almášiové a panu Ing. Pavlu Musilovi za obětavou pomoc a podnětné připomínky, které mi během práce poskytovali.

Obsah

ANOTACE

Cílem této práce bylo modelování fyzikálních dějů na počítači, konkrétně simulace přechodových dějů v RC článku. Navržený program se zabývá měřením přechodových dějů, tzn. nabíjením a vybíjením ve stejnosměrném a střídavém napětí.

První část práce zahrnuje především teoretické poznatky o přechodovém jevu, popisuje RC článek jeho nabíjení a vybíjení. Pro lepší názornost jsou jednotlivé poznatky doplněny grafy. V praktické části popisuji vytvořený program pro simulaci přechodových dějů a jejich výpočtů. Program jsem vytvořil ve vývojovém prostředí Microsoft Visual Studio a je napsán v programovacím jazyce C#.

Výsledkem práce je možnost ověřit řešení přechodových dějů nejen matematicky, ale i simulačně. To by mělo umožnit studentům lepší, názornou představu pro pochopení. Součástí práce je funkční aplikace v podobě spustitelné verze, která se nachází na přiloženém nosiči.

Klíčová slova

Simulace, přechodový jev, RC článek, graf

ANNOTATION

The aim of this work was modeling of physical processes on computer, namely simulation of transition processes in RC circuit. The proposed program deals with the measurement of transition processes, ie. charging and discharging in DC and AC voltage.

The first part of the thesis includes mainly the theoretical knowledge of the transition phenomenon, describes the RC article and its charging and discharging. For better illustrative purposes, the individual findings are supplemented by graphs. In the practical part I describe created program for simulation of transition processes and their calculations. I created the program in the Microsoft Visual Studio environment and it is written in the C# programming language.

The result of this work is the possibility to verify the solution of transition processes not only mathematically but also by simulation. This should allow students

to have a better, clearer understanding. Part of the work is a functional application in the form of executable version, which is located on the attached carrier.

Keywords

Simulation, transition effect, RC circuit, graph

ÚVOD

Předmětem této práce je vytvoření programu na simulaci přechodových dějů v RC článku. Navržený program se zabývá měřením přechodových dějů, tzn. nabíjením a vybíjením ve stejnosměrném a střídavém napětí.

Práce je rozdělena do dvou částí. První část zahrnuje především teoretické poznatky o přechodovém jevu, popisuje RC článek jeho nabíjení a vybíjení. Pro lepší názornost jsou jednotlivé poznatky doplněny grafy.

V praktické části vytvářím program pro simulaci přechodových dějů a jejich výpočtů. Program jsem vytvořil ve vývojovém prostředí Microsoft Visual Studio, ve kterém jsem již aplikace vytvářel a mám s ním dobré zkušenosti. Aplikace je napsána v programovacím jazyce C#.

Cílem mé práce je možnost ověřit řešení přechodových dějů nejen matematicky, ale i simulačně. To by mělo umožnit studentům lepší, názornou představu pro pochopení.

TEORETICKÁ ČÁST

1 Přechodový jev

Přechodový jev nastane, přechází-li obvod z jednoho ustáleného stavu do jiného ustáleného stavu. (Bláhovec, 2005)

V elektrickém obvodu vzniká přechodný jev při náhlé změně napětí zdroje, nejčastěji vzniká přechodný jev při zapnutí nebo vypnutí obvodu, v prvcích, které jsou schopné hromadit (akumulovat) energii a potom ji předat do spotřebiče. Takovými prvky jsou cívka a kondenzátor.

Přechodový jev je stav, kdy se obvod nachází mezi dvěma ustálenými stavy, a to minulým a budoucím. Do budoucího ustáleného stavu se obvod dostane až po určité době, přesně po skončení přechodového jevu, po zániku přechodné složky stavové veličiny. Stavová veličina je veličina, která se s časem mění spojitě, je zachován její směr, není z hlediska času přerušena. Doba, která je nezbytně nutná k ukončení přechodového jevu, se pohybuje okolo 5τ, kde hodnota τ se označuje jako časová konstanta, usuzujeme z ní na rychlost zániku přechodné složky stavové veličiny. (Kindrát, 2011)

Přechodové jevy se znázorňují pomocí tzv. přechodové charakteristiky, ta znázorňuje závislost napětí (proudu) na čase a její reakci na skokovou změnu. Řešení přechodových dějů provádíme matematickým výpočtem, simulací, měřením na digitálním osciloskopu nebo analyzátoru.

V praxi rozeznáváme dva druhy obvodů s přechodovými jevy prvního řádu. Obvod RL, kde jako akumulační prvek používáme cívku. Obvod RC, kde jako akumulační prvek používáme kondenzátor. Ve své práci se věnuji přechodovým jevům v RC článku.

1.1 Přechodový jev RC obvodu

Obvod obsahuje kondenzátor jako prvek akumulační a rezistor jako prvek pasivní. Dále se skládá ze zdroje napětí a spínače, který slouží k zahájení přechodového jevu.

Přechodový jev RC obvodu je krátkodobý děj, který však nemůže trvat libovolnou krátkou dobu. Po připojení obvodu ke zdroji napětí nedosáhne proud okamžitě konečné hodnoty, protože s jeho vznikem je spojena existence elektrického pole, které má určitou energii. Při konečném výkonu zdroje je proto potřebné určit dobu pro vytvoření elektrostatického pole. Po odpojení zdroje od obvodu nemůže vytvořené elektrické pole okamžitě zaniknout. Opět je k tomuto zániku potřebná určitá doba. Proud proto klesá tak, jak klesá energie.

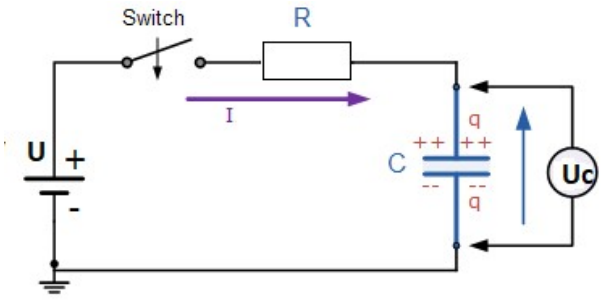
Zajímá nás tedy doba nabíjení a vybíjení kondenzátoru, která je vlastně dobou trvání přechodového děje. Závisí na velikosti nabíjecího a vybíjecího proudu, který je dán velikostí rezistoru a kapacitou kondenzátoru.

Rezistor je elektronická součástka, jejíž základní požadovanou vlastností je elektrický odpor žádané velikosti. Do obvodu zařazujeme rezistor za účelem snížení velikosti elektrického proudu.

Kondenzátor je elektrotechnická součástka, která má stanovenou hodnotu kapacity C [F]. Kondenzátor spotřebuje elektrickou energii pro vytvoření elektrického pole, energie zůstane nahromaděná v polarizovaném dielektriku ve formě energie elektrického pole. Při zániku elektrického pole se tato energie přenáší dál do obvodu ve formě vybíjecích proudů kondenzátoru. Vlastnost prvku elektrického obvodu udržet náboj při určitém napětí se nazývá kapacita.

Konstrukčně je kondenzátor tvořen dvěma elektrodami, mezi nimiž je nevodivá vrstva (dielektrikum), která nedovolí, aby se částice s nábojem dostaly do kontaktu, a tím došlo k vybití elektrických nábojů. Velikost náboje je na obou elektrodách stejná.

Aby je bylo možné analyzovat, je potřeba nahradit skutečný obvod ideálním. Na obrázku 1 je ideální rezistor v sérii s ideálním kondenzátorem.



1.2 Nabíjení RC článku

Obvod je složen ze zdroje stejnosměrného napětí U_0, ideálního kondenzátoru s kapacitou C, ideálního rezistoru s odporem R a spínače S. Obvodem před sepnutím spínače žádný proud neteče, a proto se kondenzátor nemůže nabíjet.

1. Při sepnutí spínače S, začne obvodem ihned procházet proud i_n dle vztahu

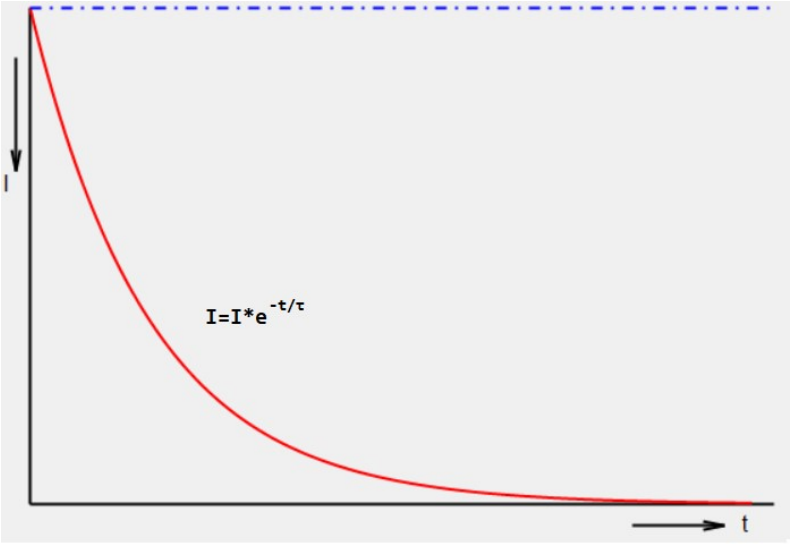
$$i_n = \frac{U - u_c}{R} \text{ [A, V,Ω]}$$

2. V čase t_0

- Je proud maximální a je dán velikostí odporu
- Kondenzátor je vybit – napětí u_c je rovno nule

V okamžiku připojení se každý nenabitý kondenzátor chová jako zkrat, obvodem prochází největší nabíjecí proud I_0, pro který platí:

$$I_0 = \frac{U}{R} \text{ [A]}$$

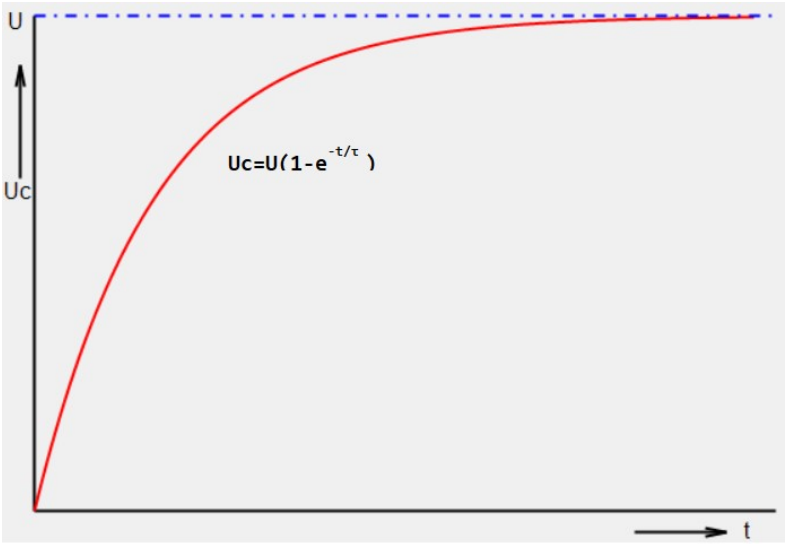


- Napětí na rezistoru je rovno napětí zdroje

Vzhledem k tomu, že počáteční napětí na kondenzátoru je 0, kondenzátor se jeví jako zkrat na vnějším obvodu. A obvodem protéká maximální proud.

3. Součet napětí na součástkách je roven napětí zdroje, proto když roste napětí na kondenzátoru, klesá napětí na rezistoru. Podle druhého Kirchhoffova zákonu platí pro uzavřený elektrický obvod:

$$u_R + u_c - U = 0$$



Jak se kondenzátor postupně nabíjí

- Roste napětí u_c – exponenciálně
- Klesá proud i v obvodu – exponenciálně
- Klesá napětí na rezistoru – exponenciálně

U přechodového jevu 1. řádu (v obvodech s jedním kondenzátorem) je řešením exponenciální funkce

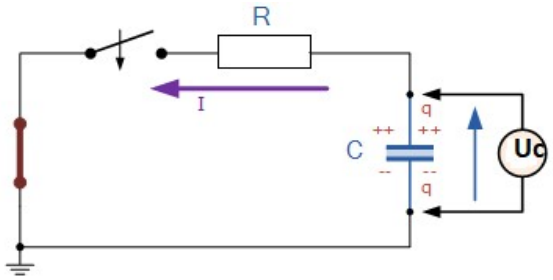
$e^{-t/\tau}$ nebo $1 - e^{-t/\tau}$

- Zavádíme časovou konstantu τ, která vyjadřuje dobu, za kterou by se nabil kondenzátor, kdyby napětí na něm vzrůstalo lineárně. Geometricky se jedná o tečnu k proudové křivce. Když rozvedeme vzorec $RC = R * \frac{Q}{U} = R * \frac{I*t}{I*R} = R * \frac{t}{R} = t$, tak zjistíme, že ze všech veličin nám zůstala pouze časová konstanta RC článku, která má jednotky v sekundách. Ta se označuje malým písmenem řecké abecedy τ [tau].

$\tau = RC$ [s]

4. Dobu ustálení přechodového děje počítáme za 3-5 tau. Teoreticky je tento děj nekonečný.

1.3 Vybíjení RC článku



Při odpojení napěťového zdroje a zkratováním obvodu se kondenzátor chová jako zdroj, a proto se začne vybíjet.

Obvodem začne procházet vybíjecí proud, který má opačnou hodnotu, než při nabíjení. V okamžiku odpojení stejnosměrného zdroje U_0 a zkratováním obvodu prochází zapojením největší možný vybíjecí proud, jeho velikost je dána vztahem $\frac{Uo}{R} = I_0$. Proud skokově vzroste na maximum opačné hodnoty, protože teče opačným směrem než při nabíjení.

Kondenzátor je nabit – napětí u_c je rovno původnímu napájecímu napětí, které začne exponenciálně klesat. Napětí na rezistoru se zmenšuje a „kopíruje“ průběh proudu.

Napětí na kondenzátoru za dobu1τ klesne přibližně na 36,8 % své původní hodnoty, 2τ dosáhne hodnoty 13,5 % ze své původní hodnoty, 3τklesne na asi 5% své původní hodnoty, 4τ má hodnotu 2% své původní hodnoty a 5τ dosáhne hodnoty 0,67% ze své původní hodnoty. Po době 5τ je napětí na kondenzátoru již tak malé, že považujeme přechodný jev za ukončený.

1.4 Střídavé napětí

Do této chvíle jsem se věnoval pouze průběhu signálu při stejnosměrném napětí. Ke změně přechodového jevu bylo třeba odpojit obvod od zdroje. Pokud na zapojení, ale připojím střídavé napětí obdélníkového časového průběhu, který se mění pouze mezi dvěma hodnotami a to konkrétně z maximální hodnoty na minimální hodnotu, což je 0 voltů.

RC obvody tak mohou vytvářet zajímavé výstupní vlny ve tvaru trojúhelníku.

Při použití velkých časových period (malých frekvencí, protože frekvence se rovná převrácené hodnotě periody f=1/T; kde f je frekvence [Hz] a T perioda [s]) zůstane kondenzátor určitou delší dobu nabitý i vybitý. Například v případě použití periody dlouhé 9τ se kondenzátor po dobu 5τ nabijí a po dobu 4τ drží stálou konstantní velikost napětí. Obdobná situace nastává i při vybíjení, kondenzátor po dobu 5τ vybijí a opět drží po dobu 4τ stálé nulové napětí.

Naopak pokud užijeme časovou periodu nižší než 5τ, tak se kondenzátor nestihne nabít na plnou hodnotu. Napětí poté začne postupně klesat k nule. Výsledné napěťové vlny mají tvar trojúhelníkových zubů. Například v případě použití periody 3τ se kondenzátor v nabíjecí fázi stihne dostat jen na velikost 95% vstupního napětí. Poté po dobu 3τ postupně klesá k minimu.

1.5 Integrační RC článek

Integrace v elektrotechnice znamená hromadění, sdružování či slučování elektrické energie. V obvodu plní matematickou funkci integrování. V praxi tento dolnopropustný filtr převádí obdélníkové vstupní vlny signálu na trojúhelníkové křivky na výstupu.

Hlavní funkcí integračního článku je, že se vzrůstající frekvencí vstupního napětí výstupní napětí klesá. Dolnopropustný filtr RC je tedy frekvenčně závislý dělič napětí, u kterého výstupní napětí na kondenzátoru klesá z důvodu poklesu reaktance kondenzátoru při vyšších frekvencích.

Výstupní signál nabývá podoby trojúhelníkových tvarů v případě, že časová perioda se rovná pěti časovým konstantám tedy 5τ.

2 Microsoft Visual Studio

K simulaci přechodového děje používám vývojové prostředí od Microsoftu konkrétně Microsoft Visual Studio. To může být použito k vytvoření konzolových i grafických aplikací jako například webové stránky, webové aplikace či Windows Forms a mnoho dalších. Já ve své práci používám konkrétně platformu Windows Forms.

Microsoft Visual Studio podporuje celou řadu programovacích jazyků. Podporu umožňuje prostřednictvím jazykových služeb. V základní sestavě jsou vestavěny tyto jazyky C, C++, VB.NET a c#, ale spousta se jich dá přidat jako jazyková služba a následně se musí doinstalovat (např. Python, Ruby, F#).

Svou práci jsem dělal ve verzi Visual Studio 2017, která na rozdíl od starších verzí obsahuje funkci Live Unit Testing. Tato novinka umožňuje elegantně programátorům vidět, na kterých řádcích program „padá“. Verze také obsahuje snadnější vyhledávání v názvech souborů pomocí klávesové zkratky Ctrl+T.

Firma Microsoft nabízí celkem tři moderní edice vývojových prostředí Microsoft Visual Studio:

- Visual Studio Community – bezplatné integrované vývojové prostředí
- Visual Studio Professional – komerční obsahuje velké množství nástrojů pro týmovou spolupráci
- Visual Studio Enterprise – nejpokročilejší integrované vývojové prostředí od Microsoftu, které nabízí mimo jiné vývoj databází, spolupráci, nástroje pro architekturu, testování a reporting

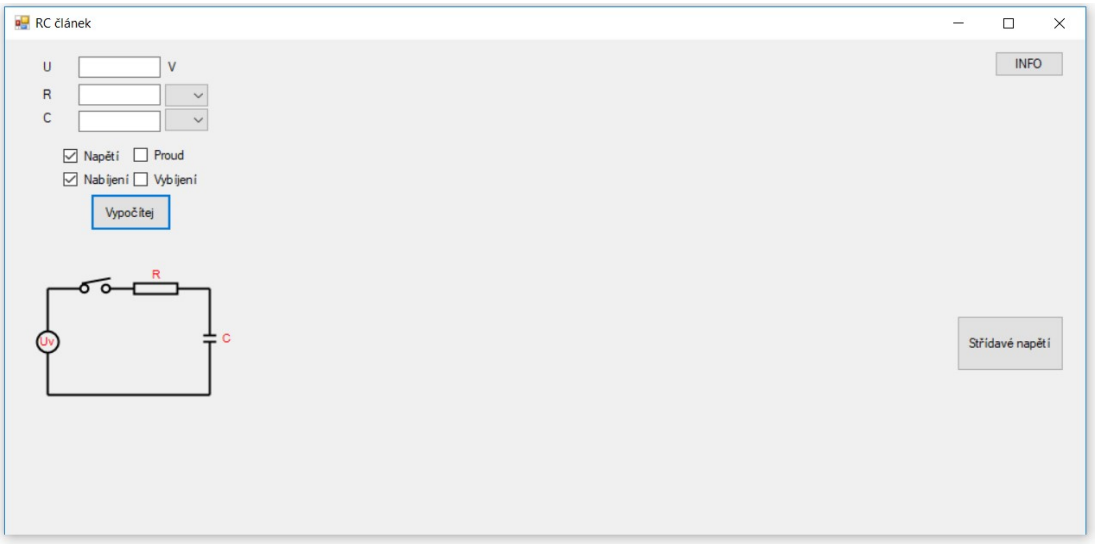
2.1 Programovací jazyk C#

C# je vysokoúrovňový programovací jazyk, který byl vytvořen firmou Microsoft. Jedná se o objektové orientovaný programovací jazyk, což znamená, že jednotlivé prvky jsou seskupeny do jednotlivých takzvaných entit. Objekty mají pevný stav a zároveň jsou přístupné jako metody pro volání. V jazyce C# také není vůbec důležité, v jakém pořadí deklarujeme metody. Dále je case sensitive, což znamená, že rozlišuje malá a velká písmena.

C# je staticky typovaný jazyk, což znamená, že všechny proměnné musí být nejdřív deklarovány a potom až použity. Výhodou staticky typovaných programovacích jazyků je ta, že obsahují kompiler, který před spuštěním kontroluje, jestli jsou všechny datové typy správně zapsány.

PRAKTICKÁ ČÁST

3 Vizuální stránka



Aplikace celkem obsahuje tři formuláře. Při spuštění aplikace je vidět pouze první hlavní (Form1) a zbylé dva jsou skryty.

Ve Form1 jsou v levém horním rohu umístěny tři textová pole pro zadávání hodnot na výpočet grafu, další čtyři zaškrtávací políčka pro výběr určitého přechodového děje, dvě rolovací tlačítka pro výběr velikosti jednotky a tlačítko pro vykreslení grafu. V levém dolním rohu je panel (panel2), v kterém je vykresleno schéma RC článku.

V prostřední části formuláře je umístěn hlavní vykreslovací panel (panel1). Tento panel je při spuštění aplikace neviditelný a zobrazuje se až poté co je stisknuto tlačítko na vykreslení grafu. V poli panelu se následně zobrazí určený přechodový jev. Původně jsem zamýšlel udělat čtyři panely, aby každý přechodový děj byl vykreslen na jiném panelu, mezi kterými by se dalo následně jednoduše přepínat. Tento způsob se ale ukázal jako neefektivní, protože vykreslování všech grafů trvalo moc dlouho.

Do pravého horního rohu jsem umístil informační tlačítko. Po jeho stisknutí se zobrazí třetí formulář (Form3) s informacemi o tom, jakým způsobem se dá program ovládat. Pod informačním tlačítkem je seznam (listBox1) a zaškrtávací tlačítko

(checkBox5). Tyto ovládací prvky umožňují uživateli zobrazit důležité body na grafu.

Jako poslední v pravé dolní části leží tlačítko (button3), které spouští druhý formulář na zobrazení nabíjení a vybíjení při přechodovém napětí. Tento formulář bude podrobněji rozebrán později.

3.1 První formulář - Form1.cs

```
public partial class Form1 : Form
{
    Form2 druhyForm=new Form2();
    Form3 třetíForm = new Form3();
    PointF[] body = new PointF[501];
    Font text = new Font("Arial", 8);
    float xPoint = 1000, yPoint = 1000, xLine, yLine, x2line, y2line, xObdélní
    double tau, R, R2, C, C2, nasobek1=1, nasobek2=1,proud,nap;
    int switchInt, switchInt2, switchInt3, switchInt4, listBoxInt,checkBoxInt;
    string U, listBoxText,U1,U2,U3,U4,U5;
```

Ve formuláři Form1 jsem nejprve vytvořil dvě třídy konkrétně formuláře Form2 a Form3, na které se budu později dovolávat.

Dále jsem si předpřipravil výčet bodů, který jsem pojmenoval body. Tento výčet čítá celkem 501 bodů. Na bodové pole používám příkaz PointF. Základní příkaz na body Point nepoužívám, protože ten pracuje s číselnou proměnnou int, která pracuje pouze s celými čísly. Jelikož jsem ve své práci potřeboval přesně počítat s desetinnými čísly, tak jsem použil PointF, která používá číselnou proměnnou float. Ta již se zmíněnými desetinnými čísly počítat umí.

Dále jsem si deklaroval celou řadu různých proměnných, aby se daly použít ve všech objektech. Odborně se tyto předdefinované proměnné nazývají členské. V aplikaci jsem použil celkem čtyři typy proměnných - string, int, float, double.

Proměnná string jako jediná z nich umí pracovat s textem. Celočíselný int je aplikován u jednoduchého sčítání. Float je použit při práci s grafikou, protože alternativní int není tak přesný. Poslední užitá proměnná double umí také pracovat s desetinnými čísly, ale na rozdíl od proměnné flout umí počítat i s třídou Math, která poskytuje běžné matematické funkce.

3.2 Zaškrtávací políčka

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    switchInt++;
    switch (switchInt)
    {
        case 1:
            checkBox2.Checked = true;
            break;
        case 2:
            checkBox2.Checked = false;
            switchInt = 0;
            break;
    }
}
```

Na obrázku je vidět kód napěťového zaškrtávacího tlačítka (checkBox1). První čtyři checkBoxy mají stejný princip.

U tohoto prvku je použita přepínací metoda switch. Ten umožňuje velice přehledně rozvětvit kód. Tato metoda přepíná mezi určitými případy - case. V mém programu jsou pouze dva případy: zaškrtnuto a nezaškrtnuto.

CheckBox1 je hned na začátku automaticky zaškrtnut. Pokud uživatel na zaškrtávací tlačítko klikne, tak se spustí metoda switch a nastane case 1 (případ). Zaškrtnutí na checkBoxu se zruší a zaškrtne se checkBox2. Jakmile se na zaškrtávací pole klikne znovu, nastane case 2. Napěťové zaškrtávací políčko se opět označí jako „checked“ a proudové se odškrtne. Metoda switch se poté ukončí.

Díky této metodě jsem zabránil tomu, aby se nevykreslovalo více grafů najednou.

3.3 Tlačítko Vypočítej

Po zmáčknutí tlačítka se zobrazí panel1, listBox1 a checkBox5, což je umožněno povolením viditelnosti – nastavením vlastnosti visible na true.

Dále pomocí dvou jednoduchých if-else podmínek aplikace zjišťuje, jestli nejsou textová pole pro odpor a kapacitu prázdná či nulová. Tato věc by v práci způsobila potíže (dělení nulou).

Pokud je číslo uznáno za vhodné, tak je deklarované do proměnné – odpor do proměnné R a kapacita do proměnné C.

V textBoxech může být pouze číselný výstup, protože jsem v události keyPress povolil zpracování jen čísel, čárek a použití tlačítka backspace.

Do listBoxu je zapsáno 5 časových konstant. Na každou z nich se dá kliknout. Program pak následně vykreslí bod na grafu.

```
for (double cas = 0; cas <= šestTau; cas += krok)
```

Hlavní výpočet v aplikaci je udělán pomocí příkazu for. For je iterační příkaz na vytvoření smyčky (blok příkazů), který se ukončí tehdy, až je výraz vyhodnocen jako true. Do této smyčky jsem zapsal 501 časových hodnot (souřadnice na ose x) a k nim počítám velikost napětí v daném čase (souřadnice na ose y). Třída Math je použita k umocnění (Math.Pow) a u použití Eulerova čísla (Math.E). K těmto dvěma metodám bylo potřeba proměnné double.

```
Ufloat = (float)U * 350;
```

Pro zápis souřadnic do bodů je nutné použít proměnnou float. Přetypování je velice snadné. Před proměnnou se v závorce napíše zamýšlená finální proměnná. Na panelu je počátek souřadnic vlevo nahoře. Proto je třeba souřadnice na ose y odečíst od velikosti panelu a získat tím pro nás opačnou, ale správnou souřadnici.

```
PointF bodU = new PointF(Ucas, inverzníU);
body[u] = bodU;
u++;
```

Souřadnice jsou do výčtu bodů zapsány pomocí dalšího bodu „bodU“, jehož souřadnice se v průběhu smyčky mění, a proto se vždy zapíše do jiného bodu z tohoto výčtu.

```
panel1.Refresh();
```

Nakonec tohoto objektu je přidán příkaz na obnovení panelu s grafem. To umožňuje, aby se graf překreslil na požadovaný. Původně jsem chtěl obnovovat (refreshovat) celý program, ale vadilo mi občasné probliknutí aplikace, a tak jsem zvolil tuto, podle mě lepší metodu.

Tlačítka Info a Střídavý proud

```
private void button2_Click_1(object sender, EventArgs e)
{
    druhyForm.Show();
}
```

Pod tlačítky button2 a button3 se schovává pouze jeden jediný řádek jednoduchého kódu. Pomocí metody Show se ukáže požadovaný předpřipravený formulář.

3.4 Vykreslení grafu v panelu 1

Pro vykreslení grafu jsem použil ovládací prvek panel. Ten je 800px široký a vysoký 500px. Ke kreslení do prvku je použita událost paint.

```
Graphics g =.e.Graphics;
```

Pro vykreslování je použita třída Graphics. Pro zjednodušení zápisu kódu jsem si dal odkaz na metodu kreslení v „paintu“.

```
g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
```

Pomocí vlastnosti SmoothingMode jsem nastavil nejvyšší kvalitu pro vykreslování grafiky. Hlavní účinek této vlastnosti je vidět u křivek, které jsou mnohem vyhlazenější a méně „zubaté“.

Nevýhodou je delší doba potřebná k vykreslení grafu.

```
Pen modrépero = new Pen(Color.Blue, 2);
modrépero.DashPattern = new float[] {4,3,1,3};
```

Níže jsou nadefinovány čtyři nástroje „pera“ pomocí, kterých je určována barva a tloušťka vykreslovaného objektu. U jednoho z těchto nástrojů – konkrétně modrépero – je nastavena čárkovaná styl vykreslování na 4px čáry, 3 px mezera, 1px čáry a zase 3px mezera.

Na osy x a y je použit příkaz DrawLine s černou barvou o tloušťce 2px.

```
g.DrawCurve(červenépero, body);
```

Na samotnou křivku znázorňující průběh přechodového děje je užit příkaz DrawCurve. Křivka má nastavenou červenou barvu o šířce 2 px a bude procházet všemi vypočítanými body.

Na zápis textu v panelu je použit příkaz DrawString, který „vykreslí“ písmo na určené souřadnice. Těmto textům jsem nastavil Font Arial a velikost písma na 8px.

Obdobnými příkazy je vytvořen i panel2. Ten má pouze na rozdíl od panelu1 nastavenou vlastnost viditelnosti (visible) na true. Prvek je ihned po spuštění vidět a nemusí se mačkat žádné tlačítko na vykreslování, aby se zviditelnil.

3.5 Seznam položek

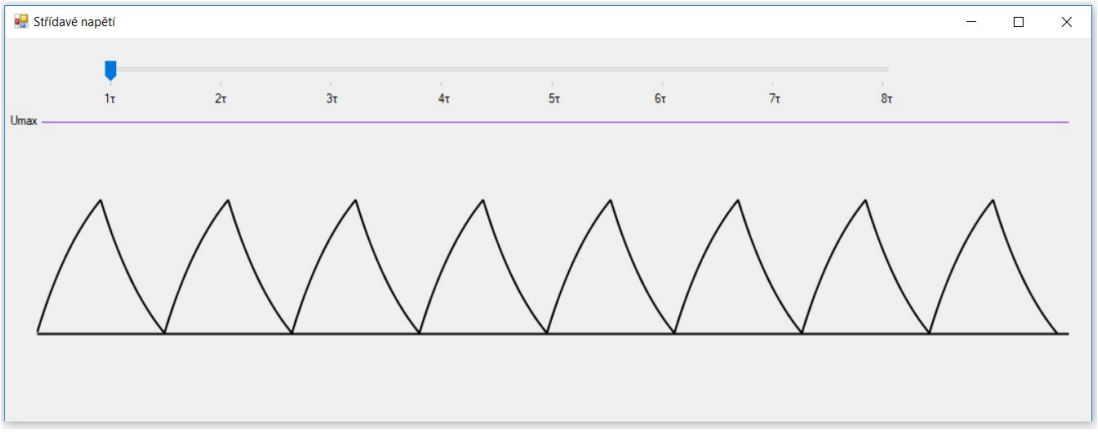
Položky jsou zobrazeny pomocí prvku ListBox. V programu se nachází pouze jeden tento prvek (listBox1). Ten zobrazuje celkem 5 položek. V každé řádce zobrazuje určitou časovou konstantu a dobu, za jak dlouho nastane. Každá řádka se dá označit a na hlavním grafickém plátně (panel1) se daný bod vykreslí a vypíše své souřadnice.

Na označení určité řádky prvku je použita událost MouseClick. Ta nastane pokaždé, když se klikne na některý řádek.

V kódu prvku se na základě podmínek (if) vypočítávají souřadnice bodu a místa, kde se velikost časové konstanty vypíše.

Pod seznamem se nachází další zaškrtávací tlačítko (checkBox5). Po jeho zaškrtnutí se do grafu vykreslí všech prvních pět časových konstant přechodového jevu.

3.6 Druhý formulář - Form2.cs



Druhý formulář aplikace se zobrazí po stisknutí tlačítka „Střídavé napětí“ na hlavním formuláři (Form1). Na tomto formuláři může uživatel vidět stoupání a klesání napětí při vybíjení a nabíjení kondenzátoru. Nahoře na formuláři je umístěn posuvník (trackBar1). Na tomto prvku se dá nastavit celkem 8 hodnot. Každá hodnota představuje periodu (dobu, jak dlouho bude obvod připojen na zdroj, a jak dlouho bude zkratován). Každá z těchto dob je počet časových konstant od 1τ do 8τ.

Na formuláři se ještě nachází celkem 8 prvků label, které slouží pouze jako popisky označující body na posuvníku.

Největší plochu formuláře zabírá panel (panel1). Ten má délku 1350px a výšku 300px. Na tento prvek se vykresluje graf přechodového děje. Tento panel jsem udělal takto dlouhý, protože jsem chtěl, aby se na něm dobře zobrazovala simulace dějů o čase až 16 časových konstant.

TrackBar má ve vlastnostech nastavené minimum na 1 a maximum na 8.

```
PointF[] napětí = new PointF[1601];
```

V tomto formuláři je předdefinován výčet o 1601 bodů s názvem napětí.

```
if (trackBar1.Value == 8)
{
    int i = 0;
    int i2 = 0;
    for (double index = 0; index <= 799; index += 1)
    {
        double u = (1 - Math.Pow(Math.E, -index / 100));
        float u2 = (float)(200 - 200 * u);
        float fIndex = (float)index * 0.625F;
        PointF na = new PointF(fIndex, u2);
        napětí[i] = na;
        i++;
    }
    for (double index = 0; index <= 800; index += 1)
    {
        double u = (1 - Math.Pow(Math.E, -index / 100));
        float u2 = (float)(200 * u);
        float fIndex = (float)(index + 800) * 0.625F;
        PointF na = new PointF(fIndex, u2);
        napětí[800 + i2] = na;
        i2++;
    }
}
```

Na obrázku je vidět vypočítání souřadnic při periodě 8τ – podmínka, když je hodnota na posuvníku rovna 8. Při této periodě se je na grafu vidět pouze jedno nabíjení a vybíjení. Pomocí příkazů for na vytvoření blokové smyčky vypočítávám všech 1601 bodů, kterými „křivka“ prochází.

Ačkoli je hned na začátku hodnota na posuvníku nastavená na 1, tak se při spuštění formuláře přechodový děj pro časovou konstantu 1τ nevykreslí, a proto jsem výpočet pro hodnotu 1τ nakopíroval do metody spuštění formuláře. Formulář se tak spouští již s vykresleným grafem.

Do události paint u prvku panel1 jsem použil opět odkaz na grafickou třídu a příkaz na přesnější vykreslování. Dále se zde vykresluje křivka z vypočítaných bodů a dvě rovné čáry, které ukazují napěťové minimum a maximum.

```
private void Form2_FormClosing(object sender, FormClosingEventArgs e)
{
    Hide();
    e.Cancel = true;
}
```

Na závěr tohoto formuláře jsem do události FormClosing, která se aktivuje zavřením formuláře pomocí „křížku“ v pravém horním rohu aplikace, přidal dva jednoduché příkazy. Příkaz Hide() dělá to, že se formulář skryje a e.Cancel=true; brání tomu, aby se formulář stal tzv. „Disposed“. To znamená, že při zavření by se stal formulář ukončeným a již by znovu nešel vyvolat (odhalit) pomocí tlačítka „Střídavé napětí“ na hlavním formuláři. Pokud by se uživatel pokusil formulář opět spustit, nastala by chyba a aplikace by tzv. „spadla“.

ZÁVĚR

V této práci jsem vytvářel program na simulaci přechodových dějů v RC článku, tedy na grafické znázornění nabíjení a vybíjení ve stejnosměrném i střídavém napětí za použití programovacího jazyka C#.

Jsem rád, že jsem si vybral tento projekt. Obohatil mě velkým množstvím zkušeností z oboru, získal jsem znalosti o programování v jazyce C#. Při práci jsem si procvičil práci s různými prvky ve Visual Studiu a práci s proměnnými. Nejvíce času jsem ale strávil počítáním souřadnic a následným vykreslováním grafu.

Během mé práce se mi naskytlo velké množství problémů, nejobtížnější bylo vymyslet postup, jak při zavření vedlejších formulářů zabránit, aby se nestaly tzv. Disposed, tedy ukončenými. Tyto formuláře by se poté už nedaly znovu vyvolat.

Do aplikace jsem zahrnul skoro všechny části, které jsem plánoval. Díky této práci jsem se naučil rychle odlaďovat a hledat chyby v programu, optimalizovat kód. Je to pro mě důležitá zkušenost, kterou rozhodně v budoucnosti hodlám využít.

Domnívám se, že výsledný program je připraven k běžnému používání, například ve školách při výuce.

Seznam použité literatury

BLAHOVEC, A. *Elektrotechnika II*. 5. vydání Praha: Informatorium, 2005, ISBN 80-7333-044-X.

Internetové zdroje

KINDRÁT A.*Učební texty pro Elektrotechniku RC*[online]. 2011 [cit. 2019-03-02]. Dostupné z: https://merkur.isste.cz/files/Uebn_texty_pro_Elektrotechniku_RC_0.pdf

C# - How do I make a textbox that only accepts numbers? - Stack Overflow [online][cit. 2019-03-16]. Dostupné z: https://stackoverflow.com/a/463335

Seznam příloh

Příloha 1

Obsah vloženého CD

- Text práce ve formátu *.pdf*
- Složka s kompletním zdrojovým kódem aplikace