UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Kodér/dekodér vybraných cyklických kódů

Knejp Lukáš

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš Knejp**
Osobní číslo:	**I10345**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Název tématu:	**Kodér/dekodér vybraných cyklických kódů**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

V teoretické části proveďte analýzu cyklických kódů z hlediska jejich vlastností a využití v praxi. Součástí teoretické části bude popis vlastního kódování a dekódování (kodéru/dekodéru).
V praktické části vytvořte SW modul umožňující pro zadanou či importovanou zprávu kódování/ dekódování této zprávy se zobrazením jednotlivých kroků kódování/dekódování. Součástí SW bude i možnost zanést chybu vzniklou při přenosu, čímž lze ověřit schopnosti detekce/opravy chyb daného kódu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. Adámek, J. Kódování, SNTL, 1989, Praha
2. Couch, L. W. Digital and Analog Communication systems, 7 th edition, Prentice Hall, 2007
3. Vlček, K. Komprese a kódová zabezpečení v multimediálních komunikacích, Praha, BEN, technická literatura, 2004, ISBN 80-86056-68-6
4. Dobeš, J., Žalud, V. Moderní radiotechnika, BEN, Praha 2006
5. S., Lin, Costello, D.J. Error Control Coding, 2ed, Perason edu, 2004

Vedoucí bakalářské práce: **Ing. Jan Pidanič, Ph.D.**
Katedra elektrotechniky

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 09. 05. 2013 Knejp Lukáš

Poděkování

Rád bych poděkoval vedoucímu této Bakalářské práce panu Ing. Janu Pidaničovi, Ph.D. za odborné vedení, konzultace a podnětné návrhy během tvorby této práce.

Anotace

Tato bakalářská práce je zaměřena na způsoby kódování a dekódovaní cyklických kódů. V teoretické části bude vysvětlena základní problematika kódování a dekódovaní různých typů kódů. Dále bude popsáno podrobně kódování a dekódovaní cyklických kódu a jejich použití. Praktická část bude obsahovat program pro kódovaní/dekódovaní cyklických kódu a jeho popis.

Klíčová slova

Kódování, Dekódování, Cyklické kódy, Kodér, Dekodér

Title

Encoder / decoder off selected cyclic codes

Annotation

This Bachelor thesis is focused on the encoding and decoding of cyclic codes. The theoretical part will explain the basic problems of encoding and decoding different types of codes. Encoding and decoding of the cyclic code and its use will be described there in detail. The practical part will include a program for encoding / decoding cyclic code and its description.

Keywords

Encoding, Decoding, Cyclic codes, Encoder, Decoder

Obsah

Seznam zkratek

ASCII	American Standard Code for Information Interchange
GUI	Graphical User Interface
CRC	Cyclic Redundancy Check
GF	Galois Field

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době vzrůstající digitalizace a přenosů dat je potřeba data různými způsoby kódovat kvůli moznosti je ukládat a přenášet. To vše je možné díky výsledkům mnoha vědních oborů.

Základy vědních oborů teorie informace a teorie kódování položil v roce 1948 Claude Shannona, který uveřejnil článek – *A Mathematical Theory of Communication*. Teorie informace je matematickou teorií, která se zabývá zákonitostmi přenosu a zpracování informace. Teorie kódování se zabývá konstrukcemi kódů a studiem vlastností kódů. První lineární kódy konstruovali Hamming a Golay. Hammingovy kódy, opravující jednoduché chyby, a Golayův kód, opravující trojnásobné chyby, dodnes patří mezi nejdůležitější kódy jak z praktického, tak i z teoretického hlediska. V padesátých letech objevili Reed a Muller první třídu kódů, opravující libovolný předepsaný počet chyb. Tyto kódy navíc mají elegantní a snadno proveditelnou metodu dekódování, tj. opravování předepsaných chyb. Později našli Bose, Chaudhuri a Hocquenghem další nesmírně důležité kódy, nazývané BCH kódy, které opravují libovolný počet chyb a mají lepší parametry než Reedovy-Mullerovy kódy. Mají však náročnější dekódování. [4]

Všeobecně Teorie kódování poskytuje řešení v podobě bezpečnostních kódu, které jsou schopné odhalovat a opravovat chyby. Výběr vhodného bezpečnostního kódu bývá kompromisem mezi stupněm požadovaného zabezpečení, jednoduchostí a cenou realizace příslušných kódových systému

Úsilí o nalezení kódových systému pro kódy s dlouhými kódovými slovy, které by byly realizovány co nejmenším počtem obvodových prvku, vedlo ke studiu tzv. cyklických kódů. Základní informace o těchto kódech, jejich vlastnostech a způsobu použití budou předmětem následujících kapitol.

První kapitola bude obsahovat základní popis kódování, kódů a jejich druhů. Dále bude obsahovat popis bezpečnostních kódů a druhů možných chyby.

V druhé kapitole budou teoretické podklady potřebné pro definici cyklických kódů. V kapitole budou popsány matematické základy grup, okruhů, těles, polynomů a konečných těles. Dále budou popsány základní rozdělení, kvalitativní parametry kódů a jejich principy potřebné pro cyklické kódy.

V třetí kapitole budou podrobně popsány cyklické kódy. Je zde popsána konstrukce systematických a nesystematických kódů a jejich dekódování. Také budou popsány kódovací a dekódovací zařízení. Dále bude uvedena ukázka pro konkrétní případ kódování a dekódování.

Čtvrtá kapitola bude obsahovat popis praktické části práce. Zde bude popsán program pro kódování a dekódování cyklických kódů a jeho vlastnosti s podrobným popisem prostředí a použitých funkcí.

1 Kódy a kódování

1.1 Úvod

Kódování je proces, při kterém se datům přiřazují určité kombinace znaků za účelem jejich ochrany, možnosti převodu do jiných soustav a ukládání na média.

Kód je předpis, podle nějž se informace převádějí do jiné reprezentace za účelem jejich přenosu či záznamu. Je to tedy způsob, jakým se informace sdělují nebo zapisují na různá média. Často se slovem kód označuje také sama zakódovaná informace.[1]

Příklad jednoduchého kódovaní si můžeme ukázat na Morseově abecedě, která převádí písmena na sekvence teček a čárek (Obr 1.). Tyto sekvence se dále mohly vysílat jako sekvence signálu, který se mohl snadno přenášet po mediu a na výstupní straně mel akustickou podobu krátkých a dlouhých tónů, které se dali jednoduše dekódovat.

A	·—	N	—·	1	·————
B	—···	O	———	2	··———
C	—·—·	P	·——·	3	···——
D	—··	Q	——·—	4	····—
E	·	R	·—·	5	·····
F	··—·	S	···	6	—····
G	——·	T	—	7	——···
H	····	U	··—	8	———··
Ch	————	V	···—	9	————·
I	··	W	·——	0	—————
J	·———	X	—··—		
K	—·—	Y	—·——		
L	·—··	Z	——··		
M	——				

Obrázek 1 - Morseova abeceda

1.2 Druhy kódovaní

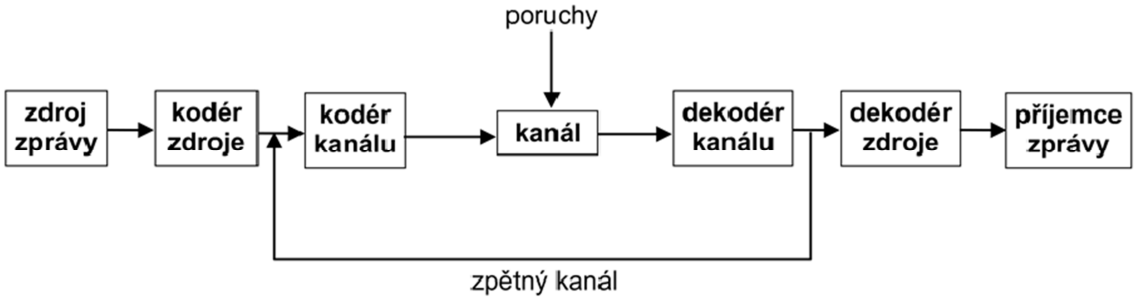
Máme tři základní typy kódování/dekódování:

- Zdrojové kódovaní
- Kódování kanálu
- Šifrování

Zdrojové kódování má za úkol odstranění nadbytečné informace ve zprávách a snižuje tak objem přenášených dat. Například komprimační metody, které dokážou vhodnými algoritmy popsat opakující se část textu.

Kódování kanálu má za úkol ochranu dat před chybami, které mohou vznikat vlivem rušení v přenosovém kanálu. Tyto kódy přidávají ke zprávám zabezpečující informace, což zvyšuje objem dat, ale chrání příjemce před špatnou interpretací zprávy. Díky své funkci můžeme tyto kódy nazývat bezpečnostní kódy.

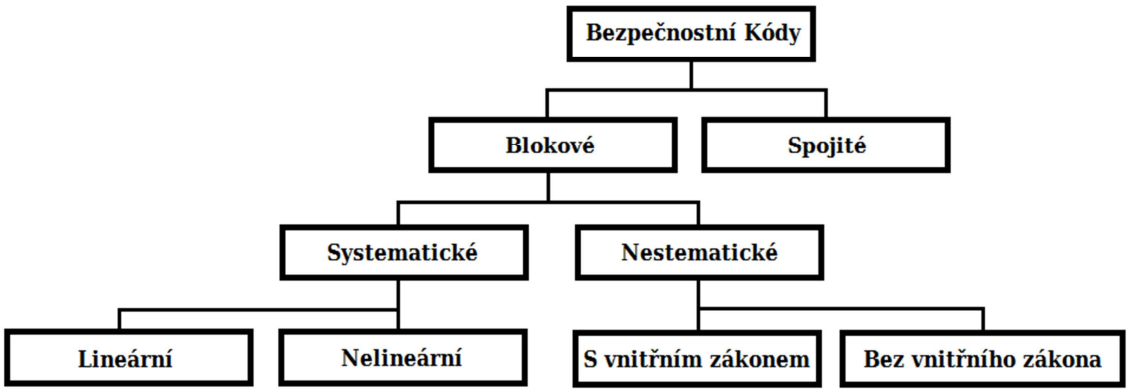
Na obrázku (Obr. 2) vidíme zdrojové a kanálové kódování/dekódování v komunikačním řetězci.



Obrázek 2 - Komunikační řetězec

1.3 Bezpečnostní kódy

Při přenosu dat dochází vlivem rušení, přeslechů, šumu a jiným vlivům ke vzniku chyb v přenosovém kanálu. Z těchto důvodů je nutné používat bezpečnostní kódy, které jsou schopny detekovat nebo i opravit chyby. Kódy můžeme dělit na detekční, které dokážou chybu najit a kódy samo opravné (korekční), které dokážou chybu najít a opravit. Typy bezpečnostních kódu jsou uvedeny na obrázku (Obr.3).



Obrázek 3 - Rozděleni bezpečnostních kódů

Nesystematické kódy na rozdíl od systematických nerozlišují informační a zabezpečující část.

- Nesystematické kódy s vnitřním zákonem sestavují kódová slova pole předem definovaných pravidel. Příkladem těchto kódů jsou Izokódy, Bergerovy kódy a Reed-Müllerovy kódy. Další informace k těmto kódům jsou v literatuře [2][3][8].
- Nesystematické kódy bez vnitřního zákona jsou kódy sestaveny bez předem definovaných pravidel a je možné je dekódovat pouze s pomocí dekódovací tabulky. Více o těchto kódech v literatuře [2][3].

1.3.1 Druhy chyb

Při přenosu může dojít k více druhům chyb. Každý druh vzniká jiným způsobem a používáme na ně jiné metody kódovaní/dekódování.

Základní druhy chyb:

- Nezávislé chyby
- Shlukové chyby

Nezávislé chyby jsou chyby, které se vyskytují v daném úseku dat nezávisle na sobě. Nezávislé chyby můžeme také ještě rozdělit na chyby jednoduché, kde na daném úseku dat vznikne jedna chyba a chyby n-násobné, kde na úseku dat vznikne n chyb.

Shlukové chyby se v úsecích dat vyskytují jako shluky po sobě jdoucích chyb.

2 Teoretické podklady

2.1 Úvod

Před popisem cyklických kódů je třeba uvést některé nutné teoretické základy, které budou popsány v následujících kapitolách.

2.2 Grupy, Okruhy, Tělesa

Grupou nazýváme množinu G spolu s binární operací na ní, která se nazývá grupová operace. Tato operace libovolným dvěma prvkům grupy a,b přiřazuje prvek c téže grupy. Dále se v definici grupy požaduje, aby grupová operace splňovala určité vlastnosti, které se nazývají axiomy[1] grupy. [4][5]

- *Uzavřenost* - Pro všechny prvky a, b v G platí, že i prvek $c = a \cdot b$ je prvkem G
- *Asociativita* - Pro všechny prvky a, b v G platí: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- *Existence neutrálního prvku* - Pro všechny a z G existuje takový neutrální prvek n z G, pro který platí: $a \cdot n = n \cdot a = a$
- *Existence inverzního prvku* - Pro každý prvek grupy a existuje prvek a^{-1} takový, že $a \cdot a^{-1} = a^{-1} \cdot a = n$, tj. jejich složení v libovolném pořadí je rovno neutrálnímu prvku.
- *Komutativita* - Pro každý prvek a, b v G platí $a \cdot b = b \cdot a$.

Příkladem komutativní grupy je množina $\{0,1\}^n$ všech binárních slov délky n .

- Neutrálním prvkem je $000\ldots0$.
- Každé slovo $v = v_1 v_2 v_3 ... v_n$ je inverzní samo sobě, protože $v + v = 000\ldots0$
- V grupě můžeme dělit pomocí inverzního prvku $\frac{a}{b} = a \cdot b^{-1}$

Okruh je v matematice algebraická struktura se dvěma binárními operacemi běžně nazývanými sčítání a násobení. Strukturu R s dvěma binárními operacemi $+$ a $\cdot$ na R nazýváme okruh, platí-li pro každé x, y, z prvky R krom všech axiomů pro grupy navíc následující axiom: [4][5]

- *Distributivita* – $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$, $(x+y) \cdot z = (x \cdot z) + (y \cdot z)$

Těleso je algebraická struktura, na které jsou definovány dvě binární operace. Je rozšířením okruhu, oproti kterému navíc přináší existenci inverzního prvku pro obě binární operace. Existence inverzního prvku umožnuje definovat operaci dělení jako násobení

[1] Axiom (z řec. axióma, to co se uznává) je tvrzení, které se předem pokládá za platné, a tudíž se nedokazuje. Podobný význam má slovo postulát.[6]

inverzním prvkem. V běžném tělese se předpokládá komutativita pouze sčítání. Pokud platí komutativita pro obě operace, těleso se nazývá komutativní. [4][5]

Příklady těles:

Množina racionálních čísel Q , množina reálných čísel R

Množina celých čísel Z není těleso, protože například k prvku 2 neexistuje v Z inverzní prvek, tedy není splněn axiom o inverzním prvku.

2.3 Kódová vzdálenost

Kódová vzdálenost (též Hammigova vzdálenost) je jednou z nejdůležitějších vlastností kódu, která udává jeho detekční a korekční schopnosti.

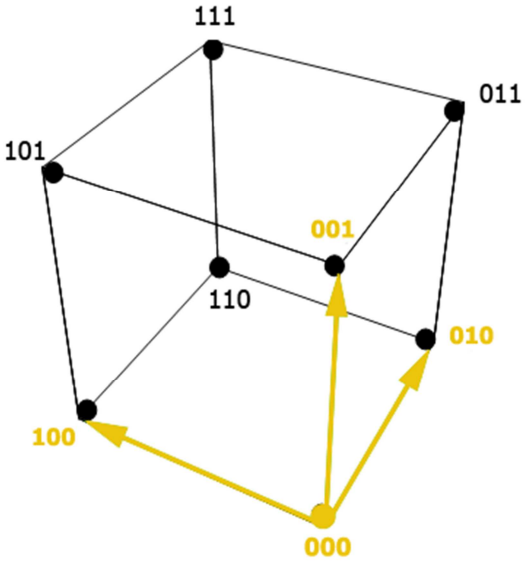
Mějme dvě slova c_1 a c_2, pak kódovou vzdáleností mezi nimi rozumíme počet změn nutných, aby slovo c_1 přešlo na slovo c_2 (a naopak). Dále definujeme d_{min} jako minimální kódovou vzdálenost, kterou mají všechna slova a d_{max} jako maximální kódovou vzdálenost, která však nemusí být mezi všemi slovy stejná.

Pro ukázku kódových vzdáleností mezi slovy můžeme uvažovat slova $c_1 = 000$, $c_2 = 001$, $c_3 = 010$, $c_4 = 011$, $c_5 = 100$, $c_6 = 101$, $c_7 = 110$ a $c_8 = 111$. Kódové vzdálenosti máme pak zobrazené v tabulce (tab. 1).

Tabulka 1 - Kódové vzdálenosti

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
c_1	-	1	1	2	1	2	2	3
c_2	1	-	2	1	2	1	3	2
c_3	1	2	-	1	2	3	1	2
c_4	2	1	1	-	3	2	2	1
c_5	1	2	2	3	-	1	1	2
c_6	2	1	3	2	1	-	2	1
c_7	2	3	1	2	1	2	-	1
c_8	3	2	2	1	2	1	1	-

Z tabulky vidíme že $d_{max} = 3$ a $d_{min} = 1$. Kódovou vzdálenost můžeme také znázornit různými grafickými formami. Příklad grafického znázorněni kódové vzdálenosti na obrázku (Obr. 4).



Obrázek 4 - Kódové vzdálenosti

2.4 Korekční a detekční schopnost kódů

Detekční a korekční schopnost kódu udává kolik chyb je kód schopen najít nebo opravit. Jak bylo zmíněno v předchozí kapitole, korekční a detekční schopnosti udává kódová vzdálenost. Počet detekovatelných chyb α pak můžeme definovat jako

$$\alpha = d_{min} - 1 \tag{1}$$

Počet opravitelných chyb β pak bude

$$\beta = \frac{d_{min} - 1}{2} \tag{2}$$

Mějme kód s kódovou vzdálenosti $d_{min} = 3$. Tento kód bude schopen detekovat $\alpha = 3 - 1 = 2$ chyby a opravovat $\beta = \frac{3-1}{2} = 1$ chybu. Vyšleme-li pak slovo $c_0 = 1001$, které vlivem rušení při přenosu přejde v slovo $c_1 = 1111$. Kód bude schopen detekovat, že nastala chyba při přenosu, ale už nebude schopen chybu opravit. Chyba totiž mohla vzniknout z více kódových slov a nebude schopen správě přiradit, které slovo je správné.

2.5 Opakovací kódy

Jelikož cyklické kódy, které mají $k = 1$ se chovají jako opakovací kódy, bude zde popsán princip těchto kódů.

Opakovací kód je velmi jednouchý kód, kde opakujeme znak do požadované délky n. Takovýto kód má pak kódovou vzdálenost $d_{min} = n$ tzn. je schopen detekovat $n-1$ chyb a opravit $\frac{n-1}{2}$ chyb. Mějme binární kód se znaky $\{0,1\}$ a $n = 4$ kódová slova pak budou 1111 a 0000 s minimální vzdáleností $d_{min} = 4$. Kód bude schopen najít 3 chyby a 2 opravit.

2.6 Lineární blokové kódy

Cyklické kódy patří do třídy lineárních kódů. Proto zde budou popsány vlastnosti těchto kódů.

Základní myšlenkou teorie bezpečnostních kódů je zavedení algebraických operací sčítání a násobení na abecedě T. Lineární kód je kód, kde lineární kombinace dvou nebo více kódových slov je opět kódové slovo.

Lineární blokový kód (nebo také $kód(n,k)$), kde n je celková délka kódu, k je počet informačních bitů a $n-k$ pak bude počet ochranných bitů. Kód tedy kóduje k vstupních bitů na celkový počet n bitů a $n-k$ bitů je vypočteno ze vstupních bitů. Lineární blokové kódy patří k systematickým kódům, takže po kódování lze oddělit část informační a zabezpečující.

Jak již bylo zmíněno při kódovaní/dekódovaní používáme operace sčítaní a násobení na abecedě T. Jelikož se v digitální technice používají převážně binární kódy, budou další operace ukázány na abecedě $T=\{0,1\}$. Operace sčítaní a odčítaní na této abecedě pak označujeme jako sčítaní/násobení modulo 2, značíme $\oplus$ pro sčítání mod2 a $\odot$ pro násobení mod2. [4] [3]

Tabulka 2 - Sčítání modulo 2

a	0	1	0	1
b	0	0	1	1
$a \oplus b$	0	1	1	0

Tabulka 3 - Násobení modulo 2

a	0	1	0	1
b	0	0	1	1
$a \odot b$	0	0	0	1

2.6.1 Kódovaní

Mějme kód (n,k) s informačními slovy $\mathbf{i} = i_0, i_1, ..., i_{n-k-1}$ a kódovými slovy $\mathbf{v} = v_0, v_1, \ldots, v_n$. Proces kódovaní pak můžeme popsat v maticové formě násobením

$$\mathbf{v} = \mathbf{i} \cdot \mathbf{G}, \tag{3}$$

kde $\mathbf{G}$ je generující matice (n,k) kódu.

Generující matice má rozměr $n \times k$ a musí splňovat 3 základní pravidla:

1. každý řádek je kódovým slovem
2. řádky jsou lineárně nezávisle, takže hodnost matice $\mathbf{G}$ je rovna k
3. každé kódové slovo je lineární kombinaci řádků matice

Generující matici pak můžeme sestavit jako jednotkovou matici $\mathbf{E}$ (informační slova), která má k řádků. K ní pak přidáme matici $\mathbf{R}$, která obsahuje zabezpečující prvky a má rozměr $k \times (n-k)$. Matici $\mathbf{R}$ vytváříme tak, aby byla dodržena požadovaná kódová vzdálenost d_{min} mezi všemi řádky.

$$\mathbf{G} = [\mathbf{E} \,|\, \mathbf{R}] \tag{4}$$

Jako příklad postupu vytváření generující matice uvažujme kód $(n,k) = (5,3)$ s $d_{min} = 2$.

- Marice $\mathbf{E}$ má rozměr $k \times k = 3 \times 3$
- Matice $\mathbf{R}$ má rozměr $k \times (n-k) = 3 \times 2$
- Matice $\mathbf{G}$ má rozměr $k \times n = 3 \times 5$

$$\mathbf{E} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

2.6.2 Dekódování

Mějme kód (n,k) s přijatými kódovými slovy $\mathbf{w} = w_0, w_1, ..., w_n$ a generující maticí $\mathbf{G}$. Chceme-li zjistit, zda přijatá slova jsou správně, můžeme opět využít maticové operace pomocí kontrolní matice $\mathbf{H}$. Kontrolní matici $\mathbf{H}$ o rozměrech $(n-k) \times n$ vypočteme:

$$\mathbf{H} = [\mathbf{R}^T \,|\, \mathbf{E}] \tag{5}$$

Pomocí matice $\mathbf{H}$ pak ověříme správnost přijatého slova výpočtem

$$\mathbf{w} \cdot \mathbf{H}^T = \mathbf{s}, \tag{6}$$

kde $\mathbf{s}$ je syndrom a je nulový pokud je slovo bez chyby. Pokud bude tedy syndrom nenulový, můžeme říct, že jsme detekovali chybu. Bude-li to opravitelná chyba, můžeme pomocí syndromu určit její polohu, jelikož platí, že vektor chyby $\mathbf{e}$ bude mít stejný syndrom $\mathbf{s}_e$ jako chybné slovo.

Jako ukázku kontrolního postupu můžeme opět uvažovat kód $(n,k) = (5,3)$, jehož matici **R** máme z přechozího přikladu a můžeme tak rovnou sestavit kontrolní matici **H**. Přijatá slova ke kontrole budou $w_1 = 01111$, $w_2 = 01101$.

$$\mathbf{w}_1 = [0 \quad 1 \quad 1 \quad 1 \quad 1],\ \mathbf{w}_2 = [0 \quad 1 \quad 1 \quad 0 \quad 1]$$

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$\mathbf{s}_{w1} = [0 \quad 1 \quad 1 \quad 1 \quad 1].\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ Syndrom $\mathbf{s}_{w1}$ je nulový, takže slovo je bez chyby

$\mathbf{s}_{w2} = [0 \quad 1 \quad 1 \quad 0 \quad 1].\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ Syndrom $\mathbf{s}_{w2}$ je nenulový, takže ve slově je chyba.

Chyba je na pozici 2 ($\mathbf{e} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$), protože platí $\mathbf{e} \cdot \mathbf{H}^T = \mathbf{s}_{w2}$

2.7 Polynomy

2.7.1 Definice

Polynom (též mnohočlen) je výraz ve tvaru

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_n x^n \ , \tag{7}$$

kde $a_1, ..., a_n$ jsou koeficienty polynomu a $n-tý$ exponent proměnné x s nenulovým koeficientem se nazývá stupněm polynomu $p(x)$. [2]

Příklady polynomů:

- $p(x) = 0$ nulový polynom
- $p(x) = 1$ polynom nultého stupně (konstanta)
- $p(x) = x + 1$ polynom 1. stupně (lineární polynom)
- $p(x) = x^2 + x + 1$ polynom 2. stupně (kvadratický polynom)
- $p(x) = x^3 + x$ polynom 3. stupně (kubický polynom)

2.7.2 Operace s polynomy

Pro účely kódovaní budou v následujících kapitolách popsány základní operace s polynomy. Pro všechny operace si nadefinujme dva polynomy $g(x)$ a $f(x)$, na kterých se budou ukazovat všechny operace.

$$g(x) = \sum_{i=0}^{n} a_i x^i, a_n \neq 0 \tag{8}$$

$$f(x) = \sum_{i=0}^{m} b_i x^i, b_m \neq 0 \tag{9}$$

1. Rovnost polynomů

Rovnost polynomů definujeme jako, $f(x) = g(x)$ když $n = m$ a pro každé i platí $a_i = b_i$.

2. Součet polynomů

Součet polynomu, jehož výsledkem bude polynom $h(x)$ je definován jako:

$$h(x) = f(x) + g(x) = \sum_{i=0}^{k} (a_i + b_i) x^i \tag{10}$$

3. Součin polynomů

Součin polynomu, kterým dostaneme polynom $h(x)$, můžeme popsat následujícím postupem. Polynom $h(x) = f(x) \cdot g(x)$ získáme vzájemným vynásobením všech členů obou polynomů, přičemž stupeň polynomu $h(x)$ je $n + m$.

$$h(x) = f(x) \cdot g(x) = \sum_{i=0}^{n} a_i x^i \cdot \sum_{i=0}^{m} b_i x^i = \sum_{i=0}^{n+m} \left(\sum_{j=0}^{i} a_j \cdot b_{i-j} \right) x^i \tag{11}$$

4. Podíl polynomů

Uvažujme rovnici (11) z předchozího příkladu, pak platí: je-li $n > m$,pak existují právě dva polynomy $r(x)$, $s(x)$ takové že platí: $f(x) = g(x) * r(x) + s(x)$, kde $s(x)$ má stupeň menší než m nebo je nulový. Pokud je nulový pak říkáme, že polynom $f(x)$ je dělitelný polynomem $g(x)$.

5. Kořeny polynomu

Číslo a je kořenem polynomu $g(x)$ pokud platí: $g(a) = 0$.

6. Ireducibilní polynom

Polynom $f(x)$ se nazývá ireducibilní, jestliže se polynom $f(x)$ nedá napsat jako součin dvou nebo více polynomů menšího stupně než je stupeň $f(x)$.

7. Primitivní polynom

Polynom, jehož koeficienty nemají společného dělitele většího než 1, se nazývá primitivní polynom. Když $f(x)$ a $g(x)$ budou primitivní polynomy pak $h(x) = f(x) \cdot g(x)$ je také primitivní polynom.

2.8 Konečná tělesa

Je-li těleso T tvořeno konečnou množinou prvků pak T je konečné těleso. Velký význam v oblasti kódovaní má konečné těleso značené jako Z_p. Konečné těleso Z_p je algebraická struktura, která je tvořená zbytky po dělení celých kladných čísel prvočíslem p.

Rozšířením konečného tělesa Z_p stupněm r vznikají tělesa označená jako Galoisova tělesa $GF(p^r)$. Pravidla pro aritmetické operace mezi prvky tělesa $GF(p^r)$ jsou určena operacemi modulo *p*.

Ukázku operací na konečném tělese vidíme v tabulkách (Tab. 4, Tab. 5) pro těleso $GF(4)$. [7]

Tabulka 4 - Sčítaní v $GF(4)$

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

Tabulka 5 - Násobení v $GF(4)$

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

2.8.1 Primitivní prvek tělesa

Primitivní prvek je takový prvek tělesa, pro který platí, že všechny nenulové prvky tělesa mohou být vyjádřeny pomoci mocnin tohoto prvku. Například pro $GF(7)$ je primitivní prvek číslo 3, protože pro něj platí:

Tabulka 6 - Primitivní prvek pro $GF(7)$

exponent	0	1	2	3	4	5
hodnota	$3^0 = 1$	$3^1 = 3$	$3^2 = 9$	$3^3 = 27$	$3^4 = 81$	$3^5 = 243$
hodnota v $GF(7)$	1	3	2	6	4	5

Hodnoty pro vyšší mocniny se dále opakují. Pomocí neutrálního prvku můžeme provádět násobení pomocí mocnin čísla 3. Například $3 \cdot 5 = 3^1 \cdot 3^5 = 3^6 = 1$. Pomocí primitívního prvku můžeme taky najít inverzní prvek. Inverzní prvek k prvku 3^i pak bude 3^{-i}.

2.9 Konečná tělesa tvořená polynomy

Stejně jako v přechozím případě byly všechny nenulové prvky tělesa vyjádřeny prostřednictvím primitivního prvku, tak i pro tělesa tvořená polynomy lze najít primitivní prvek neboli ireducibilní polynom.

Mějme ireducibilní polynom $f(x)$ stupně n nad $GF(p)$. Pak množina mnohočlenů stupně menšího než n, s operacemi sčítaní, odčítaní, násobení dělení $mod(f(x))$ tvoří konečné těleso T, které má p^n prvků. Ukázku konečného tělesa tvořeného polynomy vidíme v tabulce (Tab. 7) pro těleso $GF(2^2) = GF(4)$. Těleso je tvořeno čtyřmi binárními čísly o délce 2 bity, které můžeme vyjádřit pomoci polynomů. [8]

Tabulka 7 - Prvky $GF(4)$ vyjádřené polynomy

00	0
01	1
10	x
11	$x + 1$

3 Cyklické kódy

3.1 Úvod

Síla algebraického popisu se v teorii kódování plně projevuje u třídy cyklických kódů. Jejich definice je překvapivě jednoduchá: lineární kód je cyklický, jestliže s každým slovem $v_0, v_1, \ldots, v_n$ obsahuje i jeho cyklický posun $v_{n-1}, v_0, \ldots, v_{n-2}$. Cyklické kódy jsou rozsáhlou třídou lineárních kódu a patří mezi nejpoužívanější bezpečnostní kódy. Hlavní výhodou cyklických kódu je jednoduchá konstrukce kodéru a dekodéru, kdy se využívá zpětnovazebních posuvných registrů a logických obvodů pro sčítání. Proto jsou tyto kódy velmi využívané v místech, kde potřebujeme sestavit jednorázové kódové zařízení.[4][3]

3.2 Konstrukce cyklických kódů

Cyklické kódy vznikají cyklickými posuny kódových slov, kde pro zjednodušení matematického zápisu se slova vyjadřují pomoci polynomů. Koeficienty polynomu pro slovo $a = a_1 a_2 a_3 \ldots a_n$ budou vyjadřovat hodnotu bitu slova, pro slovo a bude polynom $a(x)$ vypadat podle následujícího vzorce:

$$a(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$$

Číslo, které přestavuje polynom $a(x)$ můžeme číst od nevyšší mocniny k nejnižší či naopak, což se velmi liší v zápisu čísel, které představují. Polynom $x^2 + x$ čtený od nevyšší mocniny k nejnižší představuje binární číslo 110, zatím co čtený naopak vyjadřuje číslo 011. Pro tuto práci uvažujme čtení od nevyšší mocniny k nejnižší, které je lepší s ohledem na praktickou část práce.

Cyklický posun o n bitů pak lze napsat jako vynásobení slova v polynomickém tvaru polynomem x^n, který odpovídá posunu. Operace násobení se provádí jako násobení $modulo(x^n - 1)$. Slovo a_2, které vznikne posunutím slova a o n bitu pak odpovídá:

$$a_2(x) = a(x) \cdot x^n \bmod (x^n - 1) \tag{12}$$

Pro operace s polynomy platí pravidla pro operace modulo2.

Příklad posunu můžeme ukázat na slově $a = 1001$ délky $n = 4$, které chceme posunout o 2 bity ($s = 100$).

$$a(x) = x^3 + 1 \qquad s(x) = x^2$$

$$a_2(x) = a(x) \cdot s(x) \bmod (x^4 - 1) = (x^3 + 1) \cdot x^2 \bmod (x^4 - 1)$$

$$a_2(x) = x^2 + x = 0110$$

Operace $modulo(x^4 - 1)$ je znázorněna v následující tabulce (Tab. 8):

Tabulka 8 - Tabulka pro $modulo\left(x^4-1\right)$

člen	1	x	x^2	x^3	x^4	x^5	x^6	x^7
modulo(x^4-1)	1	x	x^2	x^3	1	x	x^2	x^3

3.2.1 Generující polynom

Pro každý cyklicky kód (n,k) vždy existuje unikátní kódové slovo ve tvaru

$$g(x) = g_0x^0 + g_1x^1 + g_2x^2 + ... + g_{n-k}x^{n-k} \ , \tag{13}$$

pro které platí:

- Všechna kódová slova jsou násobky polynomu $g(x)$
- Polynom $g(x)$ dělí beze zbytku polynom $x^n - 1$
- Cyklickým posunem $g(x)$ o $x^0, x^1, \ldots, x^{n-k-1}$ dostaneme generující matici **G**
- g_0 a g_{n-k} nesmí být 0, jinak by všechna kódová slova začínala nebo končila číslicí 0 a ta by pak nenesla žádnou informaci.

Generující matice **G** kódu s generujícím polynomem $g(x)$:

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix} \tag{14}$$

3.2.2 Kontrolní polynom

Kontrolní polynom $h(x)$ dostaneme podílem:

$$h(x) = \frac{x^n - 1}{g(x)} \tag{15}$$

Podobně jako z $g(x)$ jsme sestavili **G** tak z $h(x)$ můžeme cyklickými posuny sestavit kontrolní matici **H,** která bude mít tvar

$$\mathbf{H} = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_k & 0 & \cdots & 0 & 0 \\ 0 & h_0 & h_1 & h_2 & \cdots & h_k & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & h_0 & h_1 & h_2 & \cdots & h_k \end{bmatrix} \tag{16}$$

3.2.3 Detekční a korekční schopnosti

Kódová vzdálenost cyklických kódů (n,k) je daná řádem generujícího polynomu. Generující polynom má řád $n-k$, což znamená, že kódová vzdálenost $d_{min}=n-k$. Výjimku tvoří kódy, kde $k=1$. Kódová vzdálenost je pak rovna $d_{min}=n$. Tyto kódy mají vlastnosti jako kódy opakovací. Z toho vyplývá, že kód (n,k) bude schopen detekovat $\alpha=n-k-1$ chyb a opravit $\beta=\frac{n-k-1}{2}$ chyb. Pro $k=1$ bude $\alpha=n-1$ a $\beta=\frac{n-1}{2}$.

Mějme kód $(n,k)=(7,4)$. Tento kód bude schopen najít $\alpha=n-k-1=2$ chyb a bude schopen opravit $\alpha=\frac{n-k-1}{2}=1$ chybu.

3.3 Kódování

Pro kódování cyklických kódů máme dvě základní metody kódování. První je založena na násobení informačního slova (polynomu) generujícím polynomem. Výsledkem je však nesystematické kódové slovo, kde nemůžeme rozlišit informační a zabezpečující část. Druhá metoda kódování je založena přičtení zabezpečující části k rozšířenému informačnímu slovu o $n-k$ bitů. Zabezpečující část získáme jako zbytek po dělení rozšířeného slova generujícím polynomem. Tímto postupem dostaneme systematické kódové slovo.

3.3.1 Nesystematické kódování

Uvažujme informační slovo $i=i_0 i_1 i_2 \ldots i_{k-1}$, které chceme zakódovat kódem (n,k) s generujícím polynomem $g(x)$. Kódovaní můžeme provést dvěma způsoby, buď vytvořit z generujícího polynomu generující matici $\mathbf{G}$ a tou vynásobit slovo i, nebo vytvořit slovo i ve tvaru polynomu a vynásobit ho s $g(x)$.

Vynásobení $\mathbf{v}=\mathbf{i}\cdot\mathbf{G}$ bude vypadat následovně:

$$\mathbf{v}=\begin{bmatrix} i_0 & i_1 & i_2 & \cdots & i_{k-1} \end{bmatrix}\cdot\begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} \end{bmatrix}, \qquad (17)$$

kde $\mathbf{v}$ je vektor kódového slova.

Postup vynásobení $v(x) = i(x) \cdot g(x)$ bude následující:

$$i(x) = i_0 x^0 + i_1 x^1 + i_2 x^2 + ... + i_{k-1} x^{k-1}$$

$$v(x) = i(x) \cdot g(x)$$

$$v(x) = (i_0 x^0 + i_1 x^1 + i_2 x^2 + ... + i_{k-1} x^{k-1}) \cdot (g_0 x^0 + g_1 x^1 + g_2 x^2 + ... + g_{k-1} x^{k-1}) \,, \qquad (18)$$

kde $v(x)$ je polynom stupně $n - 1$ který vyjadřuje kódové slovo.

Příklad postupu kódování můžeme ukázat pro kód $(n,k) = (7,3)$ s generujícím polynomem $g(x) = x^4 + x^2 + x + 1$, což odpovídá vektoru $\mathbf{v} = [1 \quad 0 \quad 1 \quad 1 \quad 1]$. Zdrojové slovo je $i = 101$ ($\mathbf{i} = [1 \quad 0 \quad 1]$).

Postup s vytvořením generující matice $\mathbf{G}$:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{v} = [1 \quad 0 \quad 1] \cdot \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} = [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1]$$

Postup s vytvořením $i(x)$:

$$i(x) = x^2 + 1$$

$$v(x) = i(x) \cdot g(x) = (x^2 + 1) \cdot (x^4 + x^2 + x + 1)$$

$$v(x) = x^6 + x^4 + x^4 + x^3 + x^2 + x^2 + x + 1$$

$$v(x) = x^6 + x^3 + x + 1 = 1001011$$

3.3.2 Systematické kódování

Pro systematické kódování kódu (n,k) využíváme operace dělení polynomů. Prvním krokem tedy bude převod informačního slova i na polynom $i(x)$. Poté musíme polynom $i(x)$ rozšířit na n bitů, abychom mohli přidat zabezpečující část. Rozšíření provedeme vynásobením polynomu $i(x)$ členem x^{n-k}. Takto upravený polynom $i(x)$ pak vydělíme generujícím polynomem $g(x)$, čímž dostaneme výsledek ve tvaru:

$$\frac{x^{n-k} \cdot i(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)} \tag{19}$$

Výsledkem podílu je $q(x)$, který není nijak použitý pro další zpracování, a zbytek po dělení $r(x)$, který přestavuje zabezpečující část. Polynom $r(x)$ bude nejvýše stupně $n-k-1$, protože dělitel $g(x)$ je stupně $n-k$. Kódové slovo pak vznikne sečtením vypočtené zabezpečující části $r(x)$ a rozšířeného slova $x^{n-k} \cdot i(x)$ tedy:

$$v(x) = x^{n-k} \cdot i(x) + r(x) \tag{20}$$

Zabezpečující část $r(x)$ bývá také označována jako CRC – *Cyclic Redundancy Check.*

Ukázku kódovaní vidíme na následujícím postupu pro kód $(n,k) = (7,3)$, kde máme informační slovo $i = 101$ k zakódovaní. Generující polynom kódu (7,3) je $g(x) = x^4 + x^2 + x + 1$. Slovo i vyjádřené polynomem bude $i(x) = x^2 + 1$. Nejprve musíme rozšířit $i(x)$ členem x^{n-k} tedy:

$$i_2(x) = x^{n-k} \cdot i(x) = x^4 \cdot (x^2 + 1) = x^6 + x^4$$

Pak musíme získat zabezpečující část $r(x)$ vydělením $i_2(x)$ polynomem $g(x)$:

$$(x^6 + x^4) : (x^4 + x^2 + x + 1) = x2 + \frac{x^3 + x^2}{x^4 + x^2 + x + 1}$$
$$-(x^6 + x^4 + x^3 + x^2)$$
$$x^3 + x^2$$

Z výsledku dělení vidíme že $r(x) = x^3 + x^2$ kódové slovo $v(x)$ pak tedy bude:

$$v(x) = i_2(x) + r(x) = (x^6 + x^4) + (x^3 + x^2) = x^6 + x^4 + x^3 + x^2$$

Neboli $v = 1011100$.

3.4 Dekódování

Po přijetí slova $w(x)$ potřebujeme ověřit, jestli je přijaté slovo kódové. Chyby odhalujeme postupem, kde přijaté slovo $w(x)$ vydělíme polynomem $g(x)$ a dostaneme výsledek ve tvaru

$$\frac{w(x)}{g(x)} = q(x) + \frac{s(x)}{g(x)} \quad , \tag{21}$$

kde $q(x)$ je dále nevyužívaný podíl a $s(x)$ je syndrom přijatého slova. Je-li syndrom $s(x)$ roven nule, tzn. slovo $w(x)$ je beze zbytku dělitelné generujícím polynomem $g(x)$, pak je přijaté slovo bez chyby. Bude-li $s(x)$ nenulový pak při přenosu nastala chyba, kterou jsme detekovali.

Pokud budeme chtít chybu opravit, budeme muset najít chybový vektor $e(x)$, který binárně vyjadřuje chyby tak, že je roven 1 na pozici, kde nastala chyba. Pro opravené kódové slovo $v(x)$ pak platí

$$v(x) = w(x) + e(x) \tag{22}$$

Pro nalezení chybového vektoru $e(x)$ může využít toho, že syndrom chybného slova $s(x)$ bude stejný jako syndrom samotného chybového vektoru $s_e(x)$, který získáme stejně jako u $w(x)$ zbytkem po dělení generujícím polynomem $g(x)$

$$\frac{e(x)}{g(x)} = q(x) + \frac{s_e(x)}{g(x)} \tag{23}$$

Pro nalezení chyby tedy musíme sestavit tabulku syndromů možných chyb, které mohou nastat a které jsme schopni najít. Poté, najdeme chybu, u které máme stejný syndrom jako u chybného slova a můžeme pak chybu odstranit použitím vztahu (22).

Příklad zjištění chyb je ukázán na kódu $(n,k) = (7,3)$ s generujícím polynomem $g(x) = x^4 + x^2 + x + 1$. Systematickým kódováním byly vytvořeny a odeslány slova $v_1 = 1011100$ a $v_2 = 0101110$. Na přijímací straně byla přijata slova $w_1 = 1011100$ a $w_2 = 1101110$. Následující postup ukazuje proces dekódování.

Slovo w_1:

$$w_1(x) = x^6 + x^4 + x^3 + x^2$$

$$\frac{w_1(x)}{g(x)} = \frac{x^6 + x^4 + x^3 + x^2}{x^4 + x^2 + x + 1} = x^2 + \frac{0}{x^4 + x^2 + x + 1}$$

$$s(x) = 0$$

Vidíme, že syndrom slova w_1 je nulový slovo tedy bylo přijato bezchybně.

Slovo w_2:

$$w_2(x) = x^6 + x^5 + x^3 + x^2 + x$$

$$\frac{w_2(x)}{g(x)} = \frac{x^6 + x^5 + x^3 + x^2 + x}{x^4 + x^2 + x + 1} = (x^2 + x + 1) + \frac{x^3 + x + 1}{x^4 + x^2 + x + 1}$$

$$s(x) = x^3 + x + 1$$

Vidíme, že syndrom je nenulový proto musíme sestavit tabulku možných chyb a najít shodný syndrom. Kód $(7,3)$ dokáže opravit podle $\beta = \frac{n-k-1}{2} = 1$chybu, tabulka syndromu pak bude obsahovat 7 možných chyb a jejich syndromy, neboť na 7 bitech může nastat 7 jednonásobných chyb.

Tabulka 9 - Tabulka syndromů

pozice chyby	$e(x)$	$s_e(x)$
0	1	1
1	x	x
2	x^2	x^2
3	x^3	x^3
4	x^4	$x^2 + x + 1$
5	x^5	$x^3 + x^2 + x$
6	x^6	$x^3 + x + 1$

Z tabulky je vidět ze syndrom $s(x)$ odpovídá chybě na pozici 6, tedy $e(x) = x^6$ Opravené slovo $v(x)$ pak dostaneme sečtením $e(x)$ a $w_2(x)$*:*

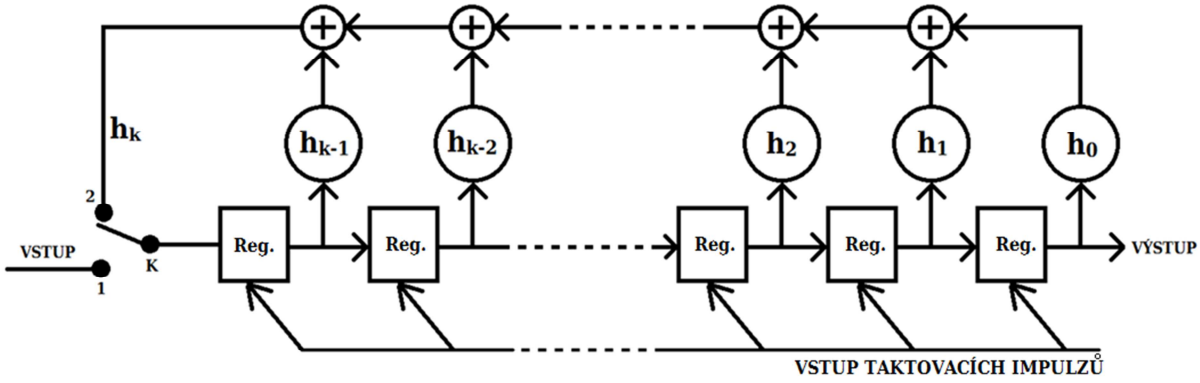
$$v(x) = w_2(x) + e(x) = (x^6 + x^5 + x^3 + x^2 + x) + x^6 = x^5 + x^3 + x^2 + x$$

Slovo v pak bude $v = 0101110$, což odpovídá vyslanému slovu v_2 a oprava tedy proběhla správně.

3.5 Kodér a dekodér

Za pomoci vhodně zapojených posuvných registrů a sčítaček lze jednoduše realizovat principy kódování. Z výstupů jednotlivých registrů se zpracovávají signály v sčítačkách s aritmetikou modulo 2. Zpracované signály ze sčítaček jsou přiváděny na vstup registru. Posuvný registr je ovládán taktovacími impulzy. Při příchodu taktovacího impulzu se do klopných obvodů zapíše informace připravená na vstupu klopného obvodu.

3.5.1 Kodér pomocí kontrolního polynomu *h(x)*



Obrázek 5 - Kodér pomocí kontrolního polynomu

K realizaci kodéru potřebujeme kontrolní polynom $h(x)$, jehož koeficienty nám udávají, jak bude kodér vypadat. Pokud bude koeficient nulový, bude na jeho místě cesta rozpojená, a proto se nekreslí do zapojení. Do zapojení se pak nekreslí ani sčítačka, do které by vedl, protože ta pak je v obvodu nadbytečná (viz obrázek (Obr. 8) s reálným kodérem s nulovým koeficientem h_2).

Popis funkce kodéru na obrázku (obr. 5) lze popsat v několika krocích:

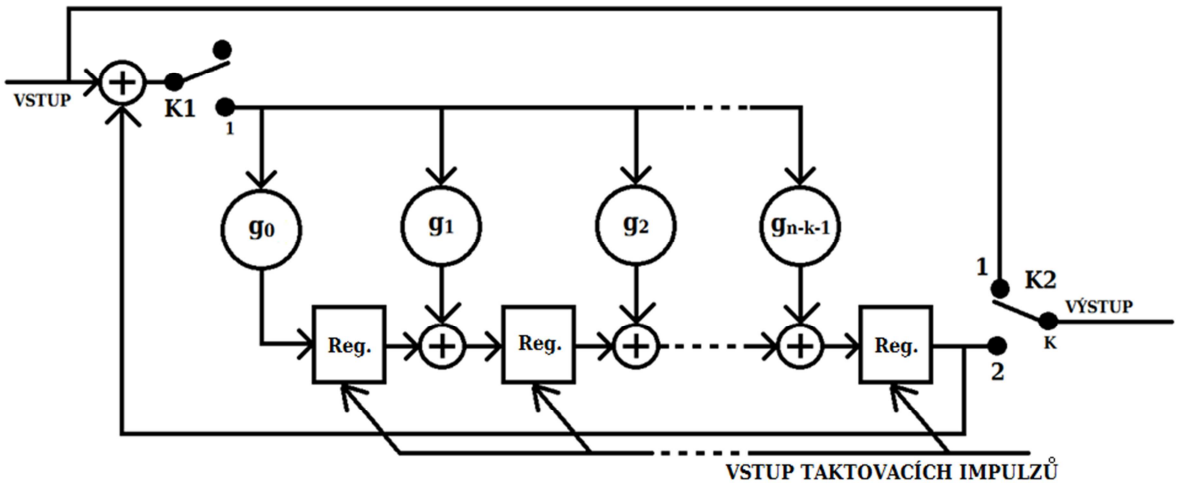
1. Přepínač K je v poloze 1 a do registrů nasuneme ze vstupu k informačních bytů. Zpětnovazební obvody zatím nemají vliv.
2. Přepínač K přepne do polohy 2, čímž zapojí zpětnovazební obvody.
3. Na příchod taktovacího impulzu se obvod posouvá, přičemž zleva se zpětnou vazbou do registrů nasouvají vypočtené zabezpečující znaky a zprava se posouvají znaky na výstup.
4. Krok 3 se opakuje $n-k$ krát a poté se spínač K opět přepne o polohy 1.
5. Do registrů se začne nasouvat další slovo a zároveň se vysouvá zbytek přechozího. Na výstupu pak máme zakódované slovo, které čteme pozpátku.

3.5.2 Kodér pomocí generujícího polynomu *g(x)*

K realizaci toho kodéru potřebujeme generující polynom s koeficienty, pro které platí pravidla jako v předchozím případě.

Proces kódovaní kodéru z obrázku (obr. 6) si opět můžeme popsat v několika krocích:

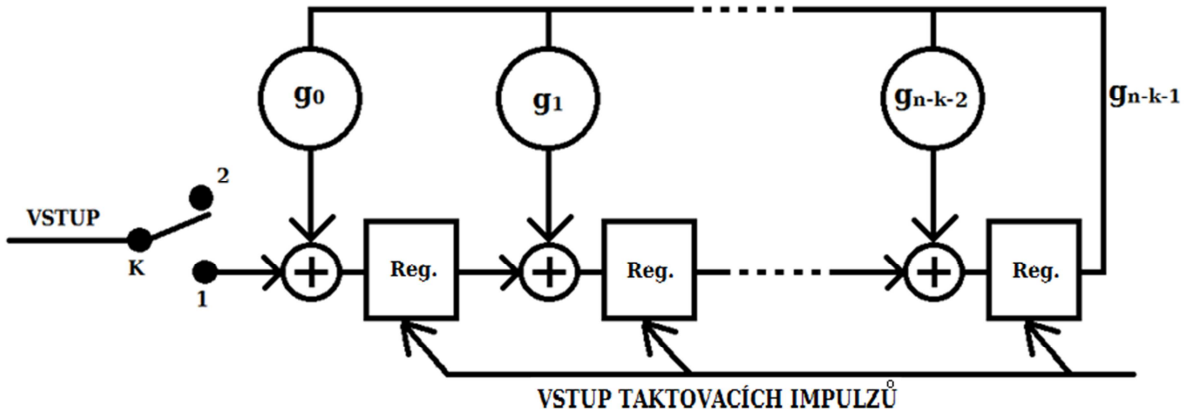
1. Oba spínače jsou v poloze 1 do registrů a zároveň na výstup jsou informační znaky.
2. Po odeslání k znaku se přepínače přepnou do polohy 2.
3. Na výstupu nyní máme informační znaky a po přepnutí se k nim přidají vypočtené bezpečnostní znaky a na výstupu tedy bude kódové slovo.



Obrázek 6 - Kodér pomocí generujícího polynomu

3.5.3 Dekodér

Dekodér opět pracuje s generujícím polynomem $g(x)$ a jeho koeficienty podle obrázku (obr. 7). Na vstup zařízení se přivádí znaky přijatého slova a po předání celého kódového slova musí byt všechny registry nulové, jinak došlo ve slově k chybě.



Obrázek 7 - Dekodér

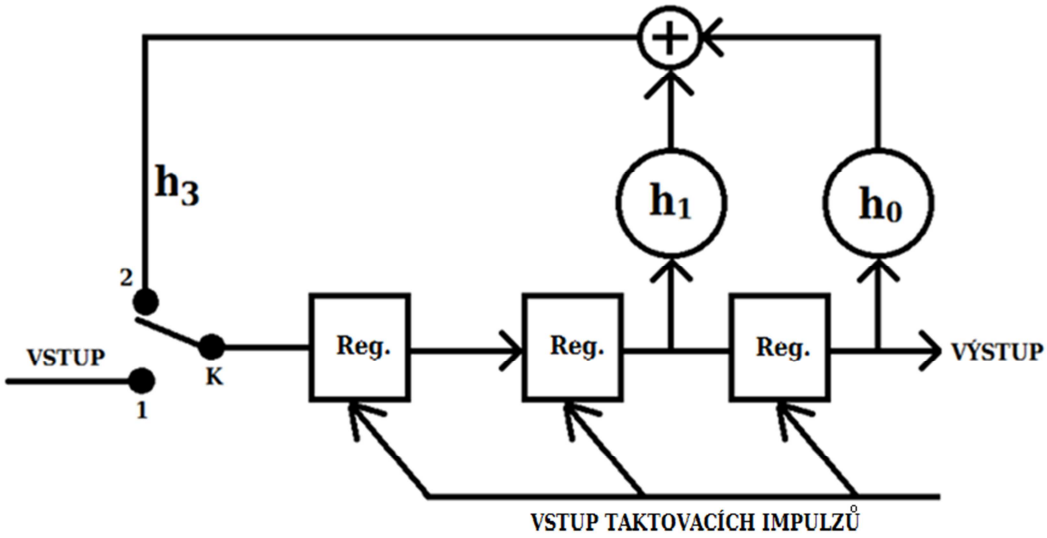
Uvažujme v kódovém slově jednonásobnou chybu. Její pozici pak zjistíme tak, že odpojíme vstup (přepnutí přepínače K do polohy 2) a sledujeme, kdy se v prvním registru objeví 1 a všechny ostatní budou nulové. Počet kroků nutných k tomuto stavu určuje pak pozici chyby v kódovém slově.

3.5.4 Ukázka reálného kodéru a dekodéru

Ukázku jednotlivých kroků v kodéru vidíme v tabulce (Tab. 10) pro kód $(n,k)=(7,3)$ s kontrolním polynomem $h(x)=x^3+x+1$ a slovem $i = 101$ k zakódovaní. Kodér tohoto kódu pak vidíme na obrázku (Obr. 8).

Tabulka 10 - Postup kódovaní

	Vložení inf. slova	1. krok	2. krok	3. krok	4. krok	Vložení dalšího slova
1. registr	1	1	1	0	0	x
2. registr	0	1	1	1	0	x
3. registr	1	0	1	1	1	x
Výstup	x	1	01	101	1101	0011101



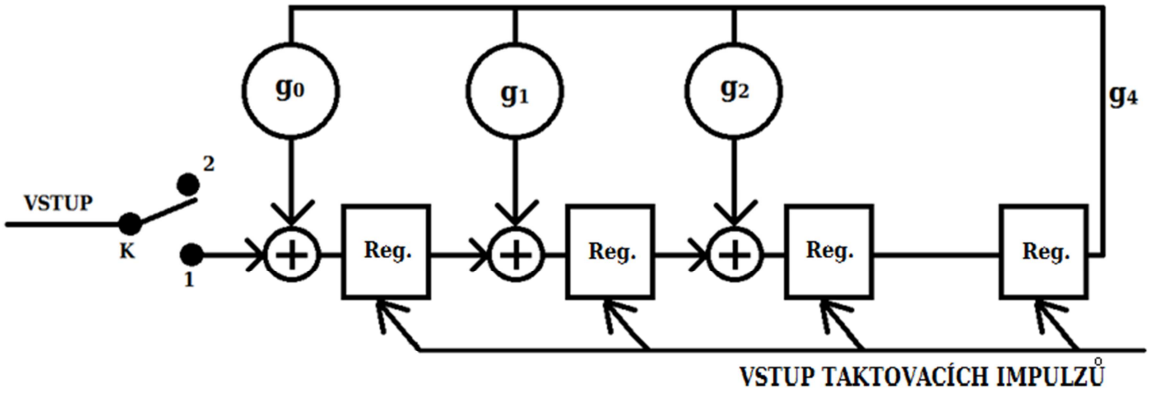
Obrázek 8 - Kodér pomocí kontrolního polynomu pro kód (7,3)

Pro ukázku dekodéru můžeme uvažovat přijaté slovo 1011101, které bylo zakódované systematickým kódem $(n,k)=(7,3)$ s generujícím polynomem $g(x)=x^4+x^2+x+1$. Proces dekódování dekodérem (obr. 9) bude následující:

Tabulka 11 - Průběh dekódovaní pomocí dekodéru

příchozí znak	1. registr	2. registr	3. registr	4. registr
počáteční stav	0	0	0	0
1	1	0	0	0
0	0	1	0	0
1	1	0	1	0
1	1	1	0	1
1	0	0	0	0
0	0	0	0	0
1	1	0	0	0

Z tabulky vidíme, že slovo je chybné zároveň také ale vidíme, že registry jsou ve stavu, kdy 1. registr má hodnotu 1 a ostatní 0 takže jsme zároveň našli i chybu, která je na pozici 0 a opravené slovo bude 1011100.



Obrázek 9 - Dekodér kódu (7,3)

3.6 Praktická ukázka postupů kódovaní a dekódování

V následující kapitole bude popsán postup kódování cyklickými kódy pro konkrétní příklad. Pro ukázku budeme uvažovat vyslané písmeno A zakódované systematickým kódem $(n,k)=(7,4)$. Příklad bude mít 2 části, 1. bude pro opravitelnou chybu a 2. pro náhodou chybu. Postup kódování bude následující:

- Převod textu do ASCII
- Rozdělení převedeného textu na informační slova
- Zakódování informačních slov daným kódem
- Zavedení opravitelné/náhodné chyby
- Dekódování přijatých slov
- Složení zprávy z dekódovaných slov

1. Převod textu do ASCII

Vyslaný text (písmeno) A má podle ASCII tabulky (Obr. 12) binární hodnotu 1000001.

2. Rozdělení převedeného textu na informační slova

Vybraný kód $(7,4)$ má délku informačních slov 4 bity a proto musíme binární hodnotu vyslaného písmena rozdělit na dvě informační slova. Původní informace je pouze 7 bitová, proto se přidá znak 0 na konec, aby bylo možné ji rozdělit na dvě 4 bitové části.

$$i_1(x)=1000=x^3$$
$$i_2(x)=0010=x$$

3. Zakódování informačních slov daným kódem

Kód $(7,4)$ má generující polynom $g(x)=x^3+x+1$ a postup kódování bude:

- Rozšířená informační slova

$$i_1(x)=i_1(x)\cdot x^{n-k}=x^3\cdot x^3=x^6$$
$$i_2(x)=i_2(x)\cdot x^{n-k}=x\cdot x^3=x^4$$

- Kódové slovo $v_1(x)$

$$x^6 : x^3+x+1 = x^3+x+1+\frac{x^2+1}{x^3+x+1}$$
$$-(x^6+x^4+x^3)$$
$$-(x^4+x^2+x)$$
$$-(x^3+x+1)$$

$$v_1(x)=i_1(x)+r_1(x)=x^6+x^2+1=1000101$$

- Kódové slovo $v_2(x)$

$$x^4 : x^3+x+1 = x+\frac{x^2+x}{x^3+x+1}$$
$$-(x^4+x^2+x)$$

$$v_2(x)=i_2(x)+r_2(x)=x^4+x^2+x=0010110$$

4. Zavedení chyby

a) Opravitelná chyba

Kód $(7,4)$ je schopen opravit $\alpha=\frac{7-4-1}{2}=1$ chybu, proto budeme uvažovat jednonásobnou chybu $e_1(x)=x^5=0100000$ pro slovo $v_1(x)$ a $e_2(x)=x^4=0010000$ pro slovo $v_2(x)$. Slova s chybou pak budou:

$$r_1(x)=v_1(x)+e_1(x)=x^6+x^5+x^2+1=1100101$$
$$r_2(x)=v_2(x)+e_2(x)=x^2+x=0000110$$

b) Náhodná chyba

Pro tento případ uvažujeme vznik náhodné chyby $e_1(x)=x^5+x+1=0100011$ pro slovo $v_1(x)$ a $e_2(x)=x^5+x^3+x^2+1=0101101$ pro slovo $v_2(x)$. Slova s chybou pak budou:

$$r_1(x)=v_1(x)+e_1(x)=x^6+x^5+x^2+x=1100110$$
$$r_2(x)=v_2(x)+e_2(x)=x^5+x^4+x^3+x+1=0111011$$

5. Dekódování přijatých slov

Tabulka syndromu:

Tabulka 12 - Tabulka syndromů pro kód $(7,4)$

pozice chyby	$e(x)$	$s_e(x)$
0	1	1
1	x	x
2	x^2	x^2
3	x^3	$x + 1$
4	x^4	$x^2 + x$
5	x^5	$x^2 + x + 1$
6	x^6	$x^2 + 1$

a) Dekódování slov s opravitelnou chybou

Slovo $r_1(x)$:

$$x^6 + x^5 + x^2 + 1 : x^3 + x + 1 = x^3 + x^2 + x + \frac{x^2 + x + 1}{x^3 + x + 1}$$

$$-(x^6 + x^4 + x^3)$$

$$-(x^5 + x^3 + x^2)$$

$$-(x^4 + x^2 + x)$$

Syndrom $s_1(x) = x^2 + x + 1$ a chyba tedy je podle tabulky syndromů (Tab. 12) $e(x) = x^5$. Opravené slovo $w_1(x)$ bude $w_1(x) = r_1(x) + e(x) = x^6 + x^2 + 1 = 1000101$

Slovo $r_2(x)$:

$$x^2 + x : x^3 + x + 1 = 0 + \frac{x^2 + x}{x^3 + x + 1}$$

Syndrom $s_2(x) = x^2 + 1$ a chyba tedy je podle tabulky syndromů (Tab. 12) $e(x) = x^4$. Opravené slovo $w_2(x)$ bude $w_2(x) = r_2(x) + e(x) = x^4 + x^2 + x = 0010110$

b) Dekódování slov s náhodnou chybou

Slovo $r_1(x)$:

$$x^6 + x^5 + x^2 + x : x^3 + x + 1 = x^3 + x^2 + x + \frac{x^2}{x^3 + x + 1}$$

$$-(x^6 + x^4 + x^3)$$

$$-(x^5 + x^3 + x^2)$$

$$-(x^4 + x^2 + x)$$

Syndrom $s_1(x) = x^2$ a chyba tedy je podle tabulky syndromů (Tab. 12) $e(x) = x^2$. Opravené slovo $w_1(x)$ bude $w_1(x) = r_1(x) + e(x) = x^6 + x^5 + x = 1100010$

Slovo $r_2(x)$:

$$x^5+x^4+x^3+x+1 : x^3+x+1 = x^2+x+\frac{1}{x^3+x+1}$$
$$-(x^5+x^3+x^2)$$
$$-(x^4+x^2+x)$$

Syndrom $s_2(x)=1$ a chyba tedy je podle tabulky syndromů (Tab. 12) $e(x)=1$. Opravené slovo $w_2(x)$ bude $w_2(x)=r_2(x)+e(x)=x^5+x^4+x^3+x=0111010$

6. Složení zprávy z dekódovaných slov
   a) Opravitelná chyba

Jelikož máme systematický kód, můžeme přímo z dekódovaných slov vyčíst informační slova a složit zprávu, která bude $i_1(x)+i_2(x)=10000010$. Po odebrání posledního přidaného znaku, budeme mít informaci 1000001 ,což odpovídá ASCII znaku A. Dekódování tedy proběhlo správně a přijali jsme správnou informaci.

   b) Náhodná chyba

Informaci složíme stejným způsobem jako v předchozím případě a bude 1100011. To odpovídá ASCII znaku c. Vidíme, že jsme přijaly špatnou informaci, jelikož jsme zavedli moc chyb a kód nebyl schopen je správně opravit.

4 Program

4.1 Vývojové prostředí

Pro tvorbu programu kodér/dekodér cyklických kódu bylo využito vývojového prostředí MATLAB, které je vhodné především kvůli jeho možnostem práce s maticemi vektory a polynomy.

4.2 Popis programu

Program *kodér/dekodér cyklických kódů* obsahuje algoritmy pro demonstraci kódování a dekódování cyklických kódů s grafickým uživatelským rozhraním. Program využívá pro kódování algoritmus pro systematické kódy celkové délky 4, 5, 7 a 15 s různými variantami délky informačních slov. Program je vytvořený pro demonstraci kódování, proto neobsahuje vyšší kombinace kódu (např. kódy délky 31). Delší kódy jsou výpočetně náročnější a při použitém algoritmu by doba kódování/dekódování nebyla v rozumných mezích. Program pracuje pouze se znaky, které jsou v ASCII tabulce (Obr. 12), jiná znaková sada není podporována.

4.3 Popis uživatelského rozhraní

Uživatelské rozhraní (dále jen GUI) se dá rozdělit na tři základní části, které vidíme na obrázku (Obr. 11).

Obrázek 10 - Uživatelské rozhraní

4.3.1 GUI 1. část

Část 1 je oblast pro výběr typu kódu. V horním panelu vybíráme celkovou délku kódu a ve spodních částech pak vybíráme mezi kombinacemi kódu dané délky. Pod panely se nám pak po výběru kódu zobrazí jeho generující polynom.

4.3.2 GUI 2. část

V 2. části jsou textová okna pro zobrazení jednotlivých kroků kódování a dekódovaní včetně editovatelného okna pro zadání zprávy a editovatelného okna pro vytvoření chyb v kódových slovech.

4.3.3 GUI 3. část

Třetí část GUI má dva menu panely s tlačítky pro ovládání programu. První panel je pro ovládání procesu kódování a dekódovaní. Tlačítko *instant* je pro provedení celého procesu najednou tzn., zakóduje zprávu vybraným kódem, přidá náhodnou opravitelnou chybu a dekóduje zprávu. Pod ním máme další tlačítka, pomocí kterých můžeme proces kódování udělat v krocích a vybrat jakou chybu zavedeme. Vedle tlačítka pro náhodnou chybu je textové okno, do kterého můžeme zadat maximální řád chyby, pokud necháme hodnotu 0, bude chyba maximálního řádu n. Druhý panel obsahuje tlačítka pro načtení zprávy z textového souboru a následné uložení vybraných výsledků do textového souboru. Dále je součástí GUI tlačítko reset, které vrátí program do původního stavu, ale nechá zprávu k zakódování a vybraný kód.

4.4 Struktura programu

4.4.1 Hlavní program

Hlavni program má celkem 2 soubory a to:

- *koderDekoder_cyklickych_kodu.m* obsahující zdrojové kódy.
- *koderDekoder_cyklickych_kodu.fig* s nastavením GUI

4.4.2 Funkce

Program využívá celkem 12 vlastních funkcí, pro zjednodušení práce se zdrojovým kódem a zpřehlednění hlavního zdrojového kódu programu.

1. Funkce *prevodZnaku* – tato funkce převede text zprávy na jednotlivé znaky a přiřadí jim jejich hodnoty z ASCII tabulky (Obr. 12). Vstupem je tedy text zprávy a výstupem matice, která obsahuje na každém řádku znak s jeho ascii kódem.
2. Funkce *kodovaSlova* – funkce vytvoří z textu zprávy informační slova vybraného typu kódu. Je-li informační část kódu menší než 7, binární hodnoty

znaků seřadí za sebe a rozdělí na požadované délky informačních slov. Pokud bude požadovaná délka větší než 7, doplní slova nulami na požadovanou délku. Vstupem této funkce je text zprávy a délka informačních slov. Výstupem je matice informačních slov a počet informačních slov v textové podobě.

3. Funkce *prevodTextu* – z textu vytvoří informační slova ve tvaru polynomů. Vstupem je text zprávy a výstupem matice polynomů kde každý řádek odpovídá jednomu informačnímu slovu.
4. Funkce *zakoduj* – funkce zakóduje předaná informační slova systematickým kódem vybraného typu. Vstupem jsou tedy informační slova ve tvaru polynomu a typ kódu, výstupem je matice polynomů kódových slov.
5. Funkce *vypisPolynomu* – tato funkce vytvoří z polynomu (matice polynomů) a požadovaného prefixu (např. ‘v(x) =‘) textovou podobu polynomu pro vypsání do textových oken. Vstupem je matice polynomů a požadovaný textový prefix, výstupem je matice slov pro výpis, kde na každém řádku je binárně vyjádřený polynom kódového slova se zadaným prefixem.
6. Funkce *generovaniChyb* – tato funkce generuje dva základní druhy chyb. Prvním typem je náhodná $n-$*násobná* chyba, kde funkce vygeneruje náhodný počet chyb na slově délky n. Druhým typem je náhodná opravitelná chyba, kde se chyba generuje podobně jako první druh ale počet generovaných chyb je omezen korekčními schopnostmi kódu. Vstupem je tedy typ kódu a chyby. Výstupem je matice polynomů odpovídající chybám.
7. Funkce *polynomZTextu* – tato funkce vezme text z textového okna a vybere z něj binární hodnoty odpovídající požadované informaci (např. kódová slova) a převede je na tvar polynomů. Vstupem je text z textového pole a typ kódu, výstupem je matice polynomů.
8. Funkce *dekoduj* – funkce pro kontrolu přijaté zprávy a opravy chyb. Vstupem je matice polynomů přijatých slov a typ kódu, výstupem je matice polynomů opravených slov.
9. Funkce *informacniSlova* – tato funkce z přijatých dekódovaných slov vybere informační část pro dekódování zprávy. Vstup je matice polynomů opravených (dekódovaných) slov a typ kódu, výstupem je matice polynomů informační slov.
10. Funkce *dekodovatZpravu* – funkce převede informační slova zpět do tvaru sedmi-bitových znaků a dekóduje text zprávy. Vstupem je matice polynomů informačních slov a typ kódů, výstupem je text dekódované zprávy.
11. Funkce *ulozitDoSouboru* – vyčte informace, které chceme uložit, a získá text z odpovídajících textových polí, který pak uloží do textového souboru. Vstupem je vektor binárních hodnot odpovídající výběru informací, které chceme uložit a struktura handles. Výstupem je soubor uložený na vybraném místě obsahující vybrané informace.
12. Funkce *puvodniStav* – tato funkce je pro tlačítko reset a uvede prvky programu do počátečního stavu. Funkce má pouze jeden vstup a to strukturu handles, která umožňuje práci s prvky programu.

Dec	Bin	ASCII	Dec	Bin	ASCII	Dec	Bin	ASCII	Dec	Bin	ASCII
0	00000000	NUL	32	00100000	SP	64	01000000	@	96	01100000	`
1	00000001	SOH	33	00100001	!	65	01000001	A	97	01100001	a
2	00000010	STX	34	00100010	"	66	01000010	B	98	01100010	b
3	00000011	ETX	35	00100011	#	67	01000011	C	99	01100011	c
4	00000100	EOT	36	00100100	$	68	01000100	D	100	01100100	d
5	00000101	ENQ	37	00100101	%	69	01000101	E	101	01100101	e
6	00000110	ACK	38	00100110	&	70	01000110	F	102	01100110	f
7	00000111	BEL	39	00100111	'	71	01000111	G	103	01100111	g
8	00001000	BS	40	00101000	(	72	01001000	H	104	01101000	h
9	00001001	HT	41	00101001	)	73	01001001	I	105	01101001	j
10	00001010	LF	42	00101010	*	74	01001010	J	106	01101010	j
11	00001011	VT	43	00101011	+	75	01001011	K	107	01101011	k
12	00001100	FF	44	00101100	,	76	01001100	L	108	01101100	l
13	00001101	CR	45	00101101	-	77	01001101	M	109	01101101	m
14	00001110	SO	46	00101110	.	78	01001110	N	110	01101110	n
15	00001111	SI	47	00101111	/	79	01001111	O	111	01101111	o
16	00010000	DLE	48	00110000	0	80	01010000	P	112	01110000	p
17	00010001	DC1	49	00110001	1	81	01010001	Q	113	01110001	q
18	00010010	DC2	50	00110010	2	82	01010010	R	114	01110010	r
19	00010011	DC3	51	00110011	3	83	01010011	S	115	01110011	s
20	00010100	DC4	52	00110100	4	84	01010100	T	116	01110100	t
21	00010101	NAK	53	00110101	5	85	01010101	U	117	01110101	u
22	00010110	SYN	54	00110110	6	86	01010110	V	118	01110110	v
23	00010111	ETB	55	00110111	7	87	01010111	W	119	01110111	w
24	00011000	CAN	56	00111000	8	88	01011000	X	120	01111000	x
25	00011001	EM	57	00111001	9	89	01011001	Y	121	01111001	y
26	00011010	SUB	58	00111010	:	90	01011010	Z	122	01111010	z
27	00011011	ESC	59	00111011	;	91	01011011	[	123	01111011	{
28	00011100	FS	60	00111100	<	92	01011100	\	124	01111100	|
29	00011101	GS	61	00111101	=	93	01011101	]	125	01111101	}
30	00011110	RS	62	00111110	>	94	01011110	^	126	01111110	~
31	00011111	US	63	00111111	?	95	01011111	_	127	01111111	DEL

Obrázek 11 - ASCII tabulka

Závěr:

Cílem práce bylo vysvětlení principů kódování a dekódování cyklických kódů a vytvoření aplikace pro demonstraci postupů kódování a dekódování.

Vysvětlení principů cyklických kódu je v kapitolách 1 až 3. Jsou zde vysvětleny základy kódování a jsou zde popsány zaklaní typy kódů a druhy bezpečnostních kódů s možnými druhy chyb, které mohou při přenosu vznikat. Dále jsou uvedeny základy pro principy cyklických kódů. Nakonec jsou popsány principy samotných cyklických kódu včetně kódovacích a dekódovacích zařízení s ukázkou reálného kodéru a dekodéru. Poslední částí cyklických kódu je pak praktická ukázka celého postupu kódování a dekódování systematickým kódem (7,4).

Aplikace pro demonstraci postupů kódování a dekódování je popsána ve čtvrté kapitole. Jsou zde vlastnosti programu s popisem grafického rozhraní, který slouží jako nápověda pro obsluhu. Dále jsou zde uvedeny funkce vytvořené pro program s jejich vstupy, výstupy a stručným popisem.

Program pracuje se znakovou sadou ASCII. Pro demonstraci kódování můžeme volit mezi celkem čtyřmi délkami kódových slov s různými kombinacemi délky informačních slov. Do programů můžeme zadávat zprávy ručně nebo importovat celé textové soubory. Importovaný text může být libovolně dlouhý, ale jsme omezení znakovou sadou ASCII. Zadaný text můžeme pak zakódovat v několika krocích, které se zobrazí v textových oknech. Všechny výsledky jednotlivých postupů se dají uložit do textového souboru. Dále je zde možnost okamžitého zakódování a dekódování zprávy pro rychlou ukázku postupu.

Jedno z možných budoucích vylepšení programu by mohlo být zavést do programu UNICODE, což by dovolilo používat všechny znaky. Dalším vylepšením by mohla být optimalizace algoritmů pro kódy s větší délkou kódových slov, které by byli schopné opravovat více chyb v reálném čase. Jako další by mohl být program přeložený do jiného programovacího jazyka (JAVA, C++). Program by pak fungoval samostatně bez potřeby vývojového prostředí MATLAB.

Seznam použité literatury:

[1] - Kódování. In: *Wikipedia: the free encyclopedia* [online]. 2013 [cit. 2013-05-03]. Dostupné z: http://cs.wikipedia.org/wiki/K%C3%B3dov%C3%A1n%C3%AD

[2] - HOFFMAN, D. *Coding theory: the essentials*. New York: M. Dekker, c1991, xi, 277 s. ISBN 08-247-8611-4.

[3] - LIN, Shu a Daniel J COSTELLO. *Error control coding: fundamentals and applications*. 2nd ed. Upper Saddle River: Pearson Prentice Hall, 2004, 1260 s. ISBN 01-304-2672-5

[4] - ADÁMEK, Jiří. *Kódování*. Praha 1: Nakladatelství technické literatury, 1989.

[5] - KOSMÁK, Ladislav - HORT, Daniel. *Algebra. Grupy, okruhy, tělesa.* 1. vyd. Masarykova univerzita, Pedagogická fakulta, 2001. 102 s. ISBN 80-210-2738-X.

[6] - Axiom. In: *Wikipedia: the free encyclopedia* [online]. 2013 [cit. 2013-05-03]. Dostupné z: http://cs.wikipedia.org/wiki/Axiom

[7] - BARTO, Libor a Jiří TŮMA. *KONEČNÁ TĚLESA*. Skripta. Dostupné z: http://www.karlin.mff.cuni.cz/~barto/student/SkriptaKonTel.pdf

[8] - VLČEK, Karel. *Komprese a kódová zabezpečení v multimediálních komunikacích*. Vyd. 1. Praha: BEN, 2000, 226 s. ISBN 80-860-5668-6