VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KNIHOVNA PRO GENEROVÁNÍ ČÁROVÝCH KÓDŮ

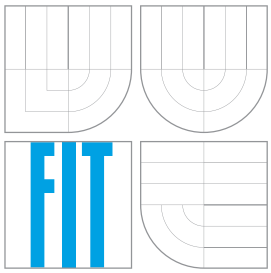
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ HORÁČEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KNIHOVNA PRO GENEROVÁNÍ ČÁROVÝCH KÓDŮ

BARCODES GENERATION LIBRARY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ HORÁČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADOVAN JOŠTH
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se v teoretické části zabývá popisem lineárních, skládaných a dvoudimenzionálních čárových kódů. Důraz je kladen na popis QR kódu. V praktické části práce je popsána implementace knihovny pro generování čárových kódů. Knihovna je pak použita v několika demonstračních aplikacích.

Abstract

In the theoretical part of this thesis are described linear, stacked and 2-D Matrix barcodes. Practical part describes the barcode generation library and shows some example which demonstrate the usage of this library.

Klíčová slova

Čárové kódy, Industrial 2/5, Interleaved 2/5, EAN, UPC, Code 128, PDF417, QR kód

Keywords

Barcodes, Industrial 2/5, Interleaved 2/5, EAN, UPC, Code 128, PDF417, QR Code

Citace

Tomáš Horáček: Knihovna pro generování čárových kódů, bakalářská práce, Brno, FIT VUT v Brně, 2011

Knihovna pro generování čárových kódů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Radovana Joštha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Horáček
18. května 2011

Poděkování

Rád bych na tomto místě poděkoval Ing. Radovanu Jošthovi za vedení a odbornou pomoc při psaní této práce.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Čárové kódy jsou dnes jednou z nejrozšířenějších forem automatické identifikace. Vznikly z potřeby mnohem rychlejšího zjišťování informací a eliminaci chyb při čtení člověkem. Využití postupně nalézaly jako forma pro identifikaci zboží, přes průmyslové aplikace, až po marketingové nasazení. Dnes se tak s nimi setkáme ve většině lidských činností a oborů.

Tato bakalářská práce si klade za cíl seznámit čtenáře s nejpoužívanějšími lineárními a dvoudimenzionálními čárovými kódy, vysvětlit mu jejich konstrukci a nasazení. Na základě teoretických znalostí je v praktických částech této práce popisována knihovna pro generování čárových kódu a ukázky jejího použití na demonstračních aplikacích.

Bakalářská práce je rozdělena na osm kapitol. Druhá kapitola popisuje historický vývoj čárových kódů od prvních nezdařilých krůčků až po rozsáhlé nasazení v dnešní době. Následuje kapitola s obecným úvodem do lineárních čárových kódů a s teoretickým popisem těch nejvýznamnějších.

Ve čtvrté kapitole se seznámíme se skládanými dvoudimenzionálními kódy, jež jsou dalším stupněm ve výjoji čárových kódů. Tvoří je sekvence po sobě jdoucích klasických kódů. Vznikly z potřeby ukládat větší množství informací na malém prostoru.

Pátá kapitola obsahuje popis dvoudimenzionálních kódů se zaměřením na QR kód[1], kterému je v této práci věnováno nejvíce prostoru a je jejím stěžejním bodem.

Následující dvě kapitoly tvoří praktickou část této práce. Zabývají se implementací knihovny pro generování čárových kódů, popisem její struktury a řešení. Na základě této knihovny jsou v sedmé kapitole uvedeny praktické ukázky jejího nasazení.

[1] Quick Response Code (Kód rychlé odpovědi)

Kapitola 2

Vývoj čárových kódů

2.1 Historie

V roce 1932 byl na Harvardově univerzitě prováděn ambiciózní projekt, který měl usnadnit nákup zboží v obchodech. Malou skupinu studentů vedl *Wallace Flint*. V návrhu projektu si měli zákazníci vybrat požadované zboží z katalogu. Ke každé položce měl být přiložen děrný štítek, který byl poté předán obsluze a ta ho vložila do čtečky. Systém zboží, podle přečteného děrného štítku, automaticky vyhledá ve skladu, dopraví na výdejní místo, přičte na účet a aktualizuje skladové zásoby. Tento systém však zůstal pouze ve fázi teoretického návrhu. [3, 10, 13].

První moderní čárový kód se začal vyvíjet až v roce 1948, kdy *Bernard Silver*, bývalý student Drexelovy univerzity ve Filadelfii, vyslyšel ředitele místního potravinářského řetězce, který požadoval vytvořit systém pro automatické čtení informací o zboží během jeho kontroly před placením. Silver o tom řekl svému příteli *Normanu Woodlandu*, který působil na Drexelu jako učitel. Problém ho natolik zaujal, že na systému začal ihned pracovat. [3, 10, 13].

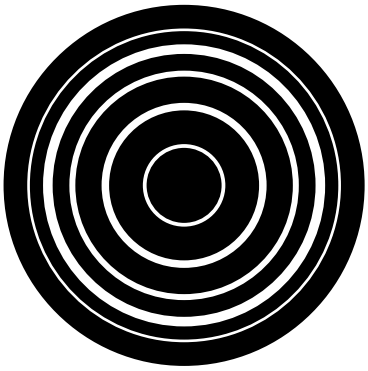
První Woodlanduv nápad bylo použít vzor z inkoustu, který by mohl být přečten pod ultrafialovým světlem. Woodland se Silverem sestrojili zařízení, které fungovalo, ale měli problémy s inkoustem, který byl nestabilní a drahý. Woodland však věřil, že jde správným směrem. Ukončil práci na Drexelu a přestěhoval se na Floridu, kde mohl v klidu pracovat. [3, 10, 13].

20. října 1949 podali Woodland a Silver žádost na patentový úřad s názvem "Classifying Apparatus and Method" (Klasifikační přístroje a metody). Woodlanduv a Silveruv kód je popisován jako "Bull's eye" (býčí oko). Sestává ze série soustředných kruhů, které tvoří kód viz. obrázek 2.1. Popis jejich kódu je velmi podobný pozdějším jednodimenzionálním kódům. Patent byl vydán 7. října 1952 pod označením "US Patent 2,612,994". Ve stejný rok byl prodán společnosti za velmi nízkou sumu, což bylo dlouho před komerčním využitím čárových kódů. [3, 10, 13].

V roce 1967 instalovala firma RCA jedno z prvních skenovacích zařízení v obchodě Kroger v Cincinnati. Kódy byly reprezentovány již zmíněným "Bull's eye" kódem. Těmi se zboží označovalo přímo v obchodě. Tento kód se však ukázal jako nevhodný pro širší využití například v průmyslu a začalo se hledat lepší řešení. [3, 10, 13].

V roce 1969 požádala NAFC[1] firmu Logicon, Inc. k vytvoření návrhu široce využitelného

[1] The National Association of Food Chains (Americká asociace obchodních řetězců)



Obrázek 2.1: Ilustrační příklad možné podoby kódu “Bull’s eye”.

systému čárových kódů. Výsledkem byl kód UGPIC[2]. V roce 1973 na jeho základě vznikl kód UPC[3], který byl doporučen pro širší využití. Kód UPC se tak brzy stal nejrozšířenějším kódem pro označování výrobků v severní americe. První čtečka UPC kódů byla instalována v roce 1974 v supermarketu Marsh ve státě Ohio a první naskenovanou položkou byl balíček žvýkaček Wrigley’s. [3, 10, 13].

2.2 Současnost

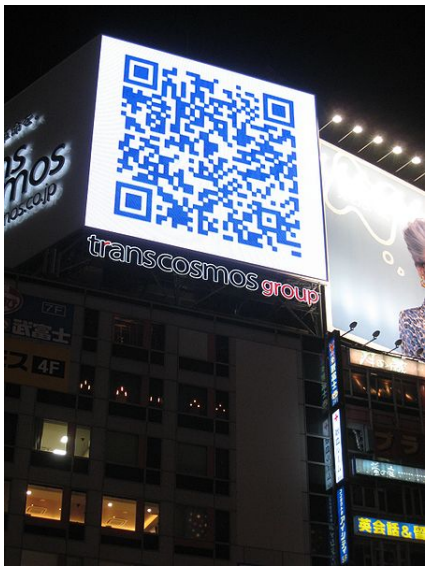
V současné době jsou čárové kódy nepostradatelným pomocníkem všude tam, kde je potřeba rychle a efektivně získávat informace o různých typech zboží a dalších věcech. Jsou široce uplatňovány nejen v obchodě, ale i ve všech průmyslových i neprůmyslových odvětvích.

Obrovský nástup dnes začínají zaznamenávat dvoudimenzionální čárové kódy, například pro označování balíků[4]. QR kód jako další zástupce dvoudimenzionálních kódů se v hojné míře využívá například v tisku pro rychlý přístup na stránky inzerenta. Na obrázku 2.3 je tento kód použit k reklamním účelům jako velký billboard.

Dalším stupněm ve vývoji čárových kódů resp. automatické identifikace jsou RFID[5] čipy, které se na rozdíl od klasických čárových kódů nečtou opticky, ale pomocí rádiové čtečky. Data jsou zde uložena na miniaturním čipu, který je vybaven malou anténou. V praxi se s RFID čipy můžeme nejčastěji setkat v podobě štítků, které se lepí na zboží, nebo se používají pro vyhledávání skladových položek [21]. Často na sobě mají natištěn i normální čárový kód pro manuální čtení.

Využití RFID čipů a dvoudimenzionálních kódů se dle mého názoru během pár let značně zvýší, obzvláště v supermarketech by se tímto dalo ušetřit velké množství času při skenování položek. Každý výrobek by byl označen RFID čipem a zákazník by pak projel jen kolem speciálního rámu s čtečkou a prodavač by okamžitě viděl obsah košíku. Nemusel by tak skenovat každou položku zvlášť. Všechno ovšem záleží na ochotě obchodních řetězců investovat do nových technologií. [24, 25]

[2] Universal Grocery Products Identification Code (Univerzální identifikační produktový kód)
[3] Universl Product Code (Univerzální produktový kód)
[4] Kód Maxicode společnosti UPS (United Parcel Service)
[5] Radio Frequency IDentification (Rádiová frekvenční identifikace)



Obrázek 2.2: QR kód na billboardu v Japonsku. [23]

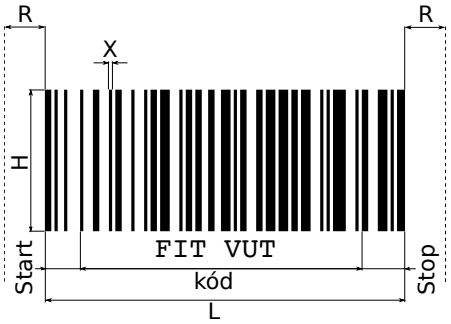
Obrázek 2.3: RFID čip na etiketě. [20]

Kapitola 3

Lineární čárové kódy

Lineární čárové kódy, někdy označované jako 1D kódy, jsou kódy sestávající pouze z tmavých čar a světlých mezer. Můžeme je rozdělit do několika skupin, podle různých parametrů. Nejčastěji se dělí podle způsobu využití a to v obchodu (označování zboží) a v průmyslu. V obchodu se nejčastěji používají kódy UPC (Severní Amerika) a EAN[1] (Evropa a další části světa). Oba tyto kódy mají pevnou délku, a proto se hodí právě pro označování výrobků. Dalším kritériem je počet znaků, které dokáže kód zakódovat, mluvíme pak o kódech numerických, numerických se speciálními znaky a kódech, které dokáží kódovat část nebo celou ASCII[2] tabulku. [4]

U většiny typů lineárních čárových kódů nesou informaci jak čáry, tak mezery, které mají proměnlivou šířku. Různou kombinací čar a mezer se dosáhne zakódování jednotlivých znaků. Každý typ kódu navíc začíná speciálním START znakem a končí STOP znakem. START a STOP znaky se u jednotlivých typů kódů liší, podle toho je čtecí zařízení může rozpoznat. Některé kódy můžou obsahovat také dělicí znak, který rozděluje kódovaný řetězec na více částí[3]. Na levé a pravé straně od kódu musí být tzv. světlé pásmo, které je zcela prázdné. Základní charakteristika lineárního čárového kódu je znázorněna na obrázku 3.1. [4]



Obrázek 3.1: Základní charakteristika lineárního čárového kódu.

[1] European Article Number (Evropské číslo zboží)
[2] American Standard Code for Information Interchange (Americký standardní kód pro výměnu informací)
[3] Např. kódy EAN a UPC.

X	Šířka modulu je nejužší element kódu, je jím čára, nebo mezera.
L	Délka kódu.
H	Výška kódu. Výška musí být dostatečně velká pro přečtení ruční čtečkou.
R	Světlé pásmo. Šířka pásma je pro každý kód jiná.
kód	Kódovaný řetězec.
Start	Start znak.
Stop	Stop znak.

Aby mohl být čárový kód přečten, musí splňovat určitou hodnotu kontrastu C. Ta je definována podle následujícího vztahu.

$$C = \frac{O_p - O_c}{O_p} \geq 0.7 \tag{3.1}$$

O_p je odraz pozadí a O_c je odraz čáry. Při dodržení této podmínky téměř nehrozí selhání při čtení kódu. Kód by musel být poškozen, nebo jinak deformován, aby přečtení selhalo. Aby se zamezilo nesprávné interpretaci informací v kódu, který je třeba nějak deformován, používá se kontrolní součet, ten je u některých kódu povinný, u některých pouze volitelný. [4]

Každý druh kódu umožňuje do jednoho kódového znaku zakódovat různé množství kombinací znaků. Obecně se u kódů uvádí označení *(n, k)*, například Code 93 (9, 3). Počet kombinací záleží na počtu modulů k, které tvoří jeden kódový znak a počtu čar, resp. mezer n. Počet nejčastějších kombinací je uveden v tabulce 3.1. Obecný vzorec pro počet možných kombinací P má následující tvar:

$$P = \frac{(n-1)!}{(2k-1)!(n-2k)!} \tag{3.2}$$

Příklad některých kódů s uvedením jejich (n, k) čísel: UPC/EAN (7, 2), Kód 93 (9, 3), Code 128 (11, 3), Code 49 (16, 4), Code 16K (11, 3), PDF417[4] (17, 4). [13]

Lineární čárové kódy se dělí také podle toho, na jak velké ploše jsou vytištěné (zobrazené). Tato vlastnost se nazývá *hustota zápisu* a záleží na šířce základního modulu X. Čím je šířka modulu menší, tím přesnější musí být čtecí zařízení. Hustota zápisu se dělí na následující tři skupiny.

- Hight Density (vysoká hustota)
- Medium Density (střední hustota)
- Low Density (nízká hustota)

Příklad jednotlivých skupin je znázorněn na obrázku 3.2. [4, 13]

Některé lineární čárové kódy se dále označují jako *diskrétní* a některé jako *navazující*. *Diskrétní* typy kódů mají jednotlivé kódové znaky odděleny mezerou, která nenese žádnou informaci. Každý kódový znak v tomto případě začíná čárou a končí také čárou (obr. 3.3). U *navazujících* kódů nejsou kódové znaky oddělené zvláštní mezerou a přiléhají těsně na sebe. Kódový znak zde začíná čarou a končí mezerou (obr. 3.4). [13]

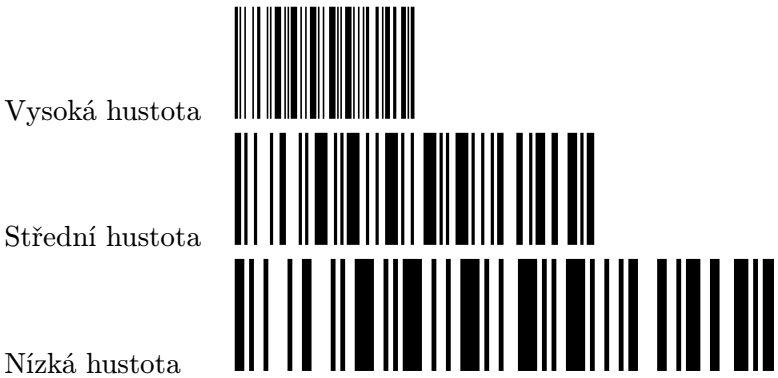
3.1 Skupina kódů 2/5

Skupina kódů 2/5 patří k historicky nejtarším, jedná se o jedny z nejjednodušších numerických kódů s proměnlivou délkou, které umožňují kódovat pouze číslice 0–9. Každý znak tvoří

[4] Portable Document File

Tabulka 3.1: Možné kombinace jednotlivých kódových znaků (n, k).

n	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
2	1	0	0	0	0
3	2	0	0	0	0
4	3	0	0	0	0
5	4	4	0	0	0
6	5	10	1	0	0
7	6	20	6	0	0
8	7	35	21	0	0
9	8	56	56	8	0
10	9	84	126	36	1
11	10	120	252	120	10
12	11	165	462	330	55
13	12	220	792	792	220
14	13	286	1287	1716	715
15	14	364	2002	3432	2002
16	15	455	3003	6435	5005
17	16	560	4368	11440	11440

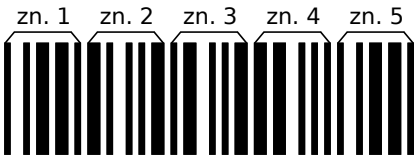


Obrázek 3.2: Příklad různých hustot zápisu pro kód kódující slovo “hustota”.

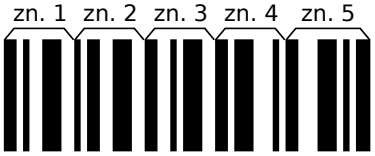
5 čar (případně mezer), z nichž dvě jsou tlusté a tři úzké. Kód se skládá ze START znaku, posloupností znaků 0-9, které tvoří data a STOP znakem, který kód ukončuje. [13, 4, 22]

3.1.1 Kód 2/5 Industrial

Kód 2/5 Industrial, někdy označovaný také jako Standard 2/5, byl vytvořen společností Identicon Corporation v roce 1968. Ve velké míře byl používán ve foto-průmyslu, logistice nebo při označování letenek. 2/5 Industrial je nejjednodušší čárový numerický kód s variabilní délkou vůbec. K zakódování vstupu se používají pouze čáry a to vždy dvě široké a tři úzké. Mezery nenesou žádnou informační hodnotu a mohou se svoji šířkou lišit. Poměr šířek široké a úzké čáry by měl být v rozmezí od 2:1 do 3:1. Šířka mezery by měla být rovna šířce základního elementu X. Výhodou tohoto typu kódu je jeho velmi snadná implementace a široké toleranční pásmo ± (15%–20%), které umožnuje tisk i na nekvalitní povrch. Nevýhodou je naopak jeho malá informační hustota, což způsobuje dlouhou délku tohoto



Obrázek 3.3: Příklad diskrétního kódu 39, kódující číslo 123.



Obrázek 3.4: Příklad navazujícího kódu 128, kódující číslo 123.

kódu. Délka kódu se spočítá podle následujícího vzorce:

$$L = 2R + 2(2PX + 3X) + N(2PX + 7X) + (N + 1)M \tag{3.3}$$

kde:
N — počet kódovaných znaků,
L — délka kódu včetně světlého pásma [mm],
X — šířka modulu [mm],
P — poměr širokých a úzkých čar,
M — šířka mezery mezi čarami [mm] (standardně M=X),
R — šířka světlého pásma.

Kód může volitelně obsahovat kontrolní číslo, které má za úkol odhalit chybu při čtení. Postup při výpočtu kontrolní číslice je následující:

1. Číslici úplně vpravo označíme jako sudou a ostatní číslice potom zprava označují střídavě sudá-lichá.
2. Sečteme čísla na všech lichých pozicích.
3. Sečteme čísla na všech sudých pozicích a výsledek vynásobíme číslem 3.
4. Sečteme výsledky z předchozích dvou kroků.
5. Kontrolní číslice je číslo, které musíme přičíst k číslu z kroku 4, aby jsme dostali číslo dělitelné bezezbytku 10-ti.
6. Pokud je číslo z kroku 4 dělitelné bezezbytku 10 je kontrolní číslicí 0.

[4, 13, 22]

Příklad 3.1: Výpočet kontrolní číslice budeme demonstrovat pro řetězec “1987”. Část postupu je uvedena v tabulce 3.2,

Po sečtení jednotlivých dílčích výsledků dostaneme číslo *57*. Podle kroku 5 musíme přičíst 3, aby bylo výsledné číslo dělitelné bezezbytku číslem 10. Kontrolní čislice pro řetězec “1987” je tedy *3*.

Tabulka 3.2: Výpočet kontrolní číslice.

Řetězec	1	9	8	7
Pozice	lichá	sudá	lichá	sudá
Váha	1	3	1	3
Výpočet	1*1	3*9	1*8	3*7
Výsledek	1	27	8	21

Tabulka 3.3: Kódovací tabulka pro kód 2/5 Industrial. L1–L5 jsou čáry 1–5, N představuje úzkou čáru a W širokou.

Znak	**L1**	**L2**	**L3**	**L4**	**L5**
0	N	N	W	W	N
1	W	N	N	N	W
2	N	W	N	N	W
3	W	W	N	N	N
4	N	N	W	N	W
5	W	N	W	N	N
6	N	W	W	N	N
7	N	N	N	W	W
8	W	N	N	W	N
9	N	W	N	W	N
START	W	W	N		
STOP	W	N	W		

V následujícím textu budeme předpokládat, že `1` znamená čáru a `0` mezeru. Sekvence `111011101` potom znamená širokou čáru, mezeru, širokou čáru, mezeru a úzkou čáru. Tato sekvence je znázorněna na obrázku 3.5.

Obrázek 3.5: Zakódování sekvence 111011101.

Pravidla pro řazení čar diskrétního kódu definuje kódovací tabulka 3.3. Za diskrétní kód se považuje sekvence čar a mezer konkrétního znaku. Oddělovací znaky nejsou mezi diskrétními kódy jeho součástí. [4, 13, 22]

Příklad 3.2: V tomto příkladu si ukážeme zakódování řetězce “19873” z příkladu 3.1, kde číslo “3” je námi vypočtená kontrolní číslice. Nyní zakódujeme každou číslici pomocí kódovací tabulky. Na začátek vložíme START znak a na konec STOP znak.

1. START znak tvoří sekvence široká čára, široká čára a úzká čára. Ekvivalentní kód zapsaný pomocí jedniček a nul je `1110111010`.

2. Číslo “1” — `11101010101110`.
3. Číslo “9” — `10111010111010`.
4. Číslo “8” — `11101010111010`.
5. Číslo “7” — `10101011101110`.
6. Číslo “3” — `11101010111010`.
7. STOP znak — `111010111`.

Výsledný kód i s grafickým znázorněním celého postupu je uveden na obrázku 3.6.

Obrázek 3.6: Příklad kódu Industrial 2/5 se zakódovaným řetězcem “19873”.

3.1.2 Kód 2/5 Interleaved

Kód Interleaved 2/5, někdy též označovaný jako překrývaný, popřípadě prokládaný kód 2/5, je numerický kód s variabilní délkou spadající do skupiny kódů 2/5 vyvinutém v roce 1972. Narozdíl od kódu Industrial 2/5 využívá k ukládání informací i mezery, z tohoto důvudu je jeho informační hustota podstatně vyšší. Kódové znaky se vždy vyskytují v párech — lichý znak je zakódován do čar a jeho párový znak do mezer. Z tohoto faktu vyplývá, že kód musí mít sudý počet znaků. Pokud máme zakódovat lichý počet znaků, vkládá se na začátek kódu 0, nebo se na konec přidá kontrolní čislice, jejíž výpočet je stejný jako v případě kódu Industrial 2/5. [2, 4, 13, 22]

Kódovací tabulka je až na START a STOP znak totožná s kódovací tabulkou 3.3. L1–L5 však mohou představovat i mezery. START znak je kódovan jako úzká čára, úzká mezera a úzká čára. STOP znak je kódován jako široká čára, úzká mezera, úzká čára. Poměry mezi širokou a úzkou čárou/mezerou je stejný jako u kódu Industrial 2/5. Délka kódu Interleaved 2/5 se spočítá podle níže uvedeného vzorce. [2, 4, 13, 22]

$$L = (N(2P + 3) + 6 + P)X \tag{3.4}$$

N — počet kódovaných znaků,
L — délka kódu,
X — šířka modulu,
P — poměr širokých a úzkých čar.

Příklad 3.3: Zakódování řetězce “1987” pomocí kódu Iterleaved 4/5.

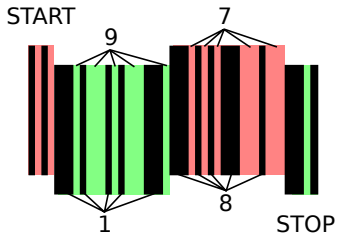
1. Podmínka sudosti délky řetězce je splněna, a proto můžeme začít s kódováním.
2. START znak tvoří sekvence úzká čára, úzká mezera a úzká čára následovaná jedním úzkým oddělovacím pruhem. Ekvivalentní kód zapsaný pomocí jedniček a nul je `1010`.

3. Následuje dvojice znaků “19”, kde se znak “1” kóduje do čar znak “9” do mezer. Dle kódovací tabulky 3.3 bude kód vypadat následovně: WnNwNnNwWn — velkými znaky je označeno kódované číslo “1” a malými “9”. Ekvivalentní kód je `111010001010001110`.

4. Druhou dvojici znaků “87” zakódujeme obdobně jako tu první, — WnNnNnWwNw, ekvivalentní kód: `111010101110001000`.

5. STOP znak — `11101`.

Výsledný kód i s grafickým znázorněním celého postupu je na obrázku 3.7.



Obrázek 3.7: Příklad kódu Interleaved 2/5 se zakódovaným řetězcem “1987”.

3.2 Kód EAN

Čárový kód EAN vznikl v roce 1978, jako rozšíření kódu UPC pro označování výrobků. Postupem času se stal standardem pro označování různých druhů zboží ve většině zemí světa. Nejčastěji používanou formou kódu EAN je EAN-13 definovaný standardizační organizací GS1 sídlící v Bruselu. [2, 4, 13, 22]

EAN-13 je navazující kód s pevně danou délkou, umožňuje kódovat 13 číslic 0–9. 13. číslice slouží jako kontrolní součet a ve vetšině případů se dopočítává automaticky. Na začátku obsahuje kód START znak, uprostřed speciální oddělovací znak, který rozděluje kód na dvě části a ukončovací STOP znak, který je stejný jako START znak. Pro kód byly definovány tři kódovací tabulky. Kódovací tabulka A pro lichou paritu, tabulka B a C pro sudou paritu. Paritu určuje číslice na 1. místě. Parita jednotlivých číslic je uvedena v tabulce 3.4. Kódovací tabulka jednotlivých znaků je v tabulce 3.5, kde číslo 1 symbolizuje čáru a 0 mezeru o šířce X. [2, 4, 13, 22]

Samotná čísla EAN kódu mají svůj vlastní význam. První 2–3 číslice určují kód země, kde je zaregitrován výrobce, další 4–5 číslice znamenají kód výrobce. 5 znaků je kód výrobku a poslední číslo je kontrolní znak. Poslední kontrolní číslice se spočítá jako součet číslic na lichých pozicích sečtená s trojnásobkem součtu číslic na sudých pozicích. Tento výsledek zaokrouhlíme nahoru na celé desítky. Kontrolní číslice se pak rovná rozdílu zaokrouhlené hodnoty a původní hodnoty. Příklad kódu EAN je na obrázku 3.8. [2, 4, 13, 22]

3.3 Kód UPC

Kód UPC byl jako standard pro označování výrobků schválen 3.května 1973 v USA a Kanadě, tedy několik let před kódem EAN. Kód UPC je podmnožinou kódu EAN a je s ním tak plně kompatibilní. Stejné jsou i všechny vlastnosti obou kódů, kromě počtu kódovaných

Tabulka 3.4: Určení parity.

1. číslice	Parita číslic 2–7	Parita číslic 8–13
0	AAAAAA	CCCCCC
1	AABABB	CCCCCC
2	AABBAB	CCCCCC
3	AABBBA	CCCCCC
4	ABAABB	CCCCCC
5	ABBAAB	CCCCCC
6	ABBBAA	CCCCCC
7	ABABAB	CCCCCC
8	ABABBA	CCCCCC
9	ABBABA	CCCCCC

Tabulka 3.5: Kódovací tabulka pro kód EAN

Znak	A	B	C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100
START	101		
STOP	101		
Děl. zn.	01010		

znaků. UPC kód jich nese 12 včetně kontrolní číslice. Kompatibilita obou kódů je zajištěna tím, že UPC kód využívá základní paritu, která je určená číslem *0*. Pokud se tedy jako první znak v EAN kódu uvede 0, vytvoří se vlastně UPC kód. [2, 13, 22]

3.4 Code 128

Čárový kód Code 128 představil v roce 1981 Ted Williams, široce se rozšířil po roce 1990. Je to alfanumerický kód s vysokou hustotou zápisu a variabilní délkou. Dokáže kódovat celou ASCII tabulku, od toho má také v názvu číslo 128, což je počet znaků v ACII tabulce. Každý zakódovaný znak je složen z 11 modulů, které mohou být jak tmavé, tak světlé. Znak je tvořen třemi čarami a třemi mezerami, vždy začíná čarou a končí mezerou. [2, 4, 13, 22]

Kód Code 128 má k dispozici 106 vzorů pro zakódování jednotlivých znaků a jeden STOP znak, který má dvojnásobnou délku. Každý kódový znak může být interpretován až třemi způsoby. Způsob interpretace závisí na použité kódové sadě. Jeden ze tří různých START znaků říká, jaká kódovací sada se použije pro první znak kódovaného řetězce. Přepínání kódovacích sad uvnitř kódu je umožněno pomocí tří tzv. SHIFT znaků. [2, 4, 13, 22]

Obrázek 3.8: Příklad kódu EAN-13 se zakódovaným řetězcem “5449000028921”.

Kódovací sady jsou označeny velkými písmeny A, B a C. Znaky jsou v těchto sadách číslovány od 0. První znak má tedy pořadí 0 a poslední 106. Sada A obsahuje ASCII znaky číslo 32–95 tj. “ ”–“_” (mezera–podtržítko), následují ASCII znaky 0–31 (NUL–US), 2 funkční znaky, znak SHIFT pro posun, SHIFT znak pro přepnutí na sadu C, SHIFT na sadu B, další dva funkční znaky, startovací znaky jednotlivých sad a STOP znak. Startovací znaky a STOP znak jsou společné pro všechny tři sady. Sada B obsahuje ASCII znaky 32–127 (mezera–DEL), dva funkční znaky, znak SHIFT, SHIFT znak pro přepnutí na sadu C, funkční znak, SHIFT znak na sadu A a další funkční znak. Sada C umožňuje kódovat čísla 00-99, dále obsahuje SHIFT znak na sadu B a A a jeden funkční znak. [2, 4, 13, 22]

Kód Code 128 obsahuje povinný kontrolní znak. Kontrolní znak je roven zbytku po dělení váženého součtu číslem 103. Vážený součet je součet hodnoty START znaku a hodnot jednotlivých datových znaků vynásobených jejich pořadím. Konkrétní výpočet je demonstrován na následujícím příkladu. [2, 4, 13, 22]

Příklad 3.4: Výpočet kontrolního znaku pro řetězec “FIT-1987”.

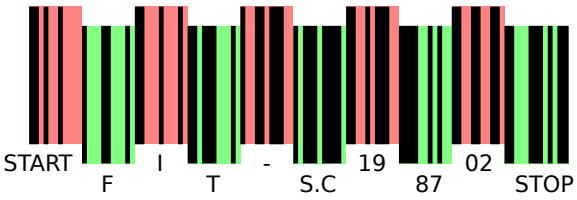
Nejdříve podle prvního znaku “F” určíme START znak. V kódovací tabulce 3.7 odpovídá písmeno “F” sadě A i B. Jako výchozí se v tomto případě vybírá nějčastěji sada B, protože obsahuje i znaky malé abecedy. Samotný postup nalezení kontrolního znaku je uveden v tabulce 3.6, kde nám vyšel zbytek 2, což dle kódovací tabulky a aktuální sadě odpovídá dvojčíslí 02. Kontrolním znakem je tedy dvojčíslí 02.

Tabulka 3.6: Výpočet kontrolního znaku kódu Code 128.

Znak	**Hodnota**	**Váha**	**Váha x Hodnota**
START B	104	1	104
F	38	1	38
I	41	2	82
T	52	3	156
-	13	4	52
SHIFT C	99	5	495
19	19	6	114
87	87	7	609
Vážený součet			1650
			1650 / 103 = 16
Zbytek			2

Příklad kódu Code 128 je uveden na obrázku 3.9. Znaku “F” odpovídá v kódovací tabulce kód 132311, ten se ve formě čar a mezer zapíše jako čára o šířce jednoho modulu,

mezera o šířce 3 moduly atd. Vždy se tedy strčídají čáry a mezery.



Obrázek 3.9: Příklad kódu Code 128 obsahujícího řetězec “FIT-1987”.

3.5 Další lineární čárové kódy

Code 39 Kód Code 39 byl prvním alfanumerickým kódem, který v roce 1974 vyvinul Dr. David C. Allais z Intermec Corporation. Tento kód je široce rozšířen všude, kromě označování obchodních položek, kde dominují kódy UPC a EAN. Kód Code 39 je diskrétní, samokontrolní[5] a má variabilní délku. Čáry a mezery mohou být dvou různých šířek. Kódu Code 39 se někdy přezdívá také “Kód 3 z 9”, protože 3 elementy jsou vždy široké a 6 je úzkých. Kažký kódovaný znak obsahuje 5 čar a 4 mezery. Kód Code 39 umožňuje kódovat čísla 0–9, velké znaky anglické abecedy A–Z, symboly “-.$/+%*” a mezeru. Znak “*” je rezervovaný pro START a STOP znak. Jednotlivé znaky jsou od sebe odděleny jednou úzkou mezerou, která nenese žádnou informaci. [13, 4, 2, 22]

Přestože kód Code 39 umožňuje kódovat pouze 43 znaků, je možné pomocí speciálního módu zakódovat celou ASCII tabulku, tedy 128 znaků. Tuto vlastnost musí podporovat i použitá čtečka. Znaky, které nejsou obsaženy ve standardní kódovací tabulce kódu Code 39 jsou v tomto případě zdvojeny tak, že se před kódovaný znak přidá některý ze symbolů. Protože je kód Code 39 samokontrolní, většinou se již nepřidává kontrolní součet. Pokud je však vyžadována větší bezpečnost dat, je možné přidat kontrolní součet, který je zbytkem po dělení číslem 43. Příklad kódu Code 39 je na obrázku 3.10. [13, 4, 2, 22]

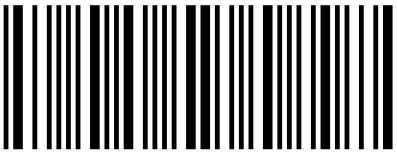
Obrázek 3.10: Příklad kódu 39.

Kód Codabar Kód Codabar byl vyvinut v roce 1972 firmou Monarch Marking Systems. Dnes se využívá například v knihovnách, ve zdravotnictní pro označování krevních transfůzí nebo při označování poštovních zásilek. Codabar je samokontrolní, diskrétní kód s variabilní délkou, který umožňuje kódovat znaky 0–9 a symboly “-$/:.+”. Dále obsahuje čtyři různé START/STOP znaky, které mohou nést další informace. Zajímavostí je, že kódované znaky mají dvě různé délky. Příklad kódu Codabar je znázorněn na obrázku 3.11. [13, 4, 2, 22]

[5] Při tiskové vadě, nebo deformaci nemůže dojít k zaměně jednoho znaku za druhý.

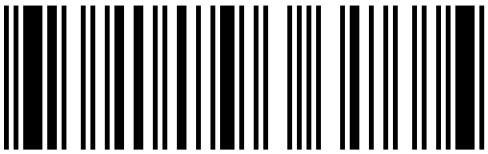
Tabulka 3.7: Kódovací tabulka pro kód Code 128.

Č.	**Kód**	**Sada**			**Č.**	**Kód**	**Sada**		
		A	B	C			A	B	C
0	212222	mezera	mezera	00	54	311123	V	V	54
1	222122	!	!	01	55	311321	W	W	55
2	222221	”	”	02	56	331121	X	X	56
3	121223	#	#	03	57	312113	Y	Y	57
4	121322	$	$	04	58	312311	Z	Z	58
5	131222	%	%	05	59	332111	[	[	59
6	122213	&	&	06	60	314111	\	\	60
7	122312	’	’	07	61	221411	]	]	61
8	132212	(	(	08	62	431111	^	^	62
9	221213	)	)	09	63	111224	_	_	63
10	221312	*	*	10	64	111422	NUL	‘	64
11	231212	+	+	11	65	121124	SOH	a	65
12	112232	,	,	12	66	121421	STX	b	66
13	122132	-	-	13	67	141122	ETX	c	67
14	122231	.	.	14	68	141221	EOT	d	68
15	113222	/	/	15	69	112214	ENQ	e	69
16	123122	0	0	16	70	112412	ACK	f	70
17	123221	1	1	17	71	122114	BEL	g	71
18	223211	2	2	18	72	122411	BS	h	72
19	221132	3	3	19	73	142112	HT	i	73
20	221231	4	4	20	74	142211	LF	j	74
21	213212	5	5	21	75	241211	VT	k	75
22	223112	6	6	22	76	221114	FF	l	76
23	312131	7	7	23	77	413111	CR	m	77
24	311222	8	8	24	78	241112	SO	n	78
25	321122	9	9	25	79	134111	SI	o	79
26	321221	:	:	26	80	111242	DLE	p	80
27	312212	;	;	27	81	121142	DC1	q	81
28	322112	<	<	28	82	121241	DC2	r	82
29	322211	=	=	29	83	114212	DC3	s	83
30	212123	>	>	30	84	124112	DC4	t	84
31	212321	?	?	31	85	124211	NAK	u	85
32	232121	@	@	32	86	411212	SYN	v	86
33	111323	A	A	33	87	421112	ETB	w	87
34	131123	B	B	34	88	421211	CAN	x	88
35	131321	C	C	35	89	212141	EM	y	89
36	112313	D	D	36	90	214121	SUB	z	90
37	132113	E	E	37	91	412121	ESC	{	91
38	132311	F	F	38	92	111143	FS	|	92
39	211313	G	G	39	93	111341	GS	}	93
40	231113	H	H	40	94	131141	RS	~	94
41	231311	I	I	41	95	114113	US	DEL	95
42	112133	J	J	42	96	114311	FNC 3	FNC 3	96
43	112331	K	K	43	97	411113	FNC 2	FNC 2	97
44	132131	L	L	44	98	411311	Shift B	Shift A	98
45	113123	M	M	45	99	113141	Sada C	Sada C	99
46	113321	N	N	46	100	114131	Sada B	FNC4	Sada B
47	133121	O	O	47	101	311141	FNC 4	Sada A	Sada A
48	313121	P	P	48	102	411131	FNC 1	FNC 1	FNC 1
49	211331	Q	Q	49	103	211412	Start A		
50	231131	R	R	50	104	211214	Start B		
51	213113	S	S	51	105	211232	Start C		
52	213311	T	T	52	106	2331112	Stop (7 čar a mezer)		
53	213131	U	U	53					



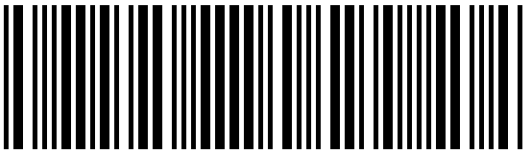
Obrázek 3.11: Příklad kódu Codabar obsahující data “010987”, START znak “A” a STOP znak “B”.

Code 93 Kód Code 93 byl představen v roce 1982 jako konkurent pro kód Code 39 s vysokou hustotou zápisu. Kód má variabilní délku a je navazující. Každý kódovaný znak je tvořen třemi čarami a třemi mezerami, složené z devíti modulů — jedná se tedy o kód (9, 3). Odtud vznikl i jeho název Code 93. Kód umožňuje kódovat 48 různých znaků, čtyři speciální SHIFT znaky však umožnují zakódovat celou ASCII tabulku. START a STOP znaky jsou stejné, za STOP znak se ale přidává ještě tzv. ukončovací znak, který tvoří čára široká jeden modul. Kód nachází uplatnění například pri inventarizaci majetku. Ukázka kódu je na obrázku 3.12. [13, 4, 22]



Obrázek 3.12: Příklad kódu Code 93 obsahující data “CODE 93”.

Code 11 Jako reakce na pořadavek o diskrétní kód s vysokou hustotou zápisu byl v roce 1977 vytvořen kód Code 11. Označení Code 11 si získal díky možnosti zakódovat 11 znaků (čísla 0–9 a pomlčka). Jednotlivé kódové znaky tvoří vždy tři čáry a dvě mezery. Čáry mohou mít tři různé rozměry a mezery dva. Kód Code 11 není samokontrolní, a proto obsahuje jeden nebo dva kontrolní znaky. Využití si Code 11 našel při označování telekomunikačních komponent. Na obrázku 3.13 je uveden příklad tohoto kódu. [13, 22]



Obrázek 3.13: Příklad kódu Code 11 obsahující data “0123456789”.

Channel kód Idea Channel kódu se zrodila v hlavách pracocvníků AIM[6] v roce 1992, když pluli v trajektu přes kanál La Manche[7]. Rozhodli se použít vyšší řády kódu — až $n = 30$, který umožňuje v jednom kódovém znaku zakódovat větší množství kombinací. Tento způsob může velmi efektivně zredukovat potřebné místo pro čárový kód. Příklad Channel kódu je uveden na obrázku 3.14. [13]

[6] Association for Automatic Identification and Mobility (Asociace pro automatickou identifikaci a mobilitu)

[7] V angličtině má označení “English Channel”, od kterého vznikl i název kódu.

Obrázek 3.14: Příklad Channel kódu obsahující data “1987”.

Postnet Dle požadavků poštovního průmyslu se zrodil mírně modifikovaný typ kódu, jenž má všechny čáry a mezery stejně široké, ale různě vysoké. Tento typ kódů má obrovskou výhodu v rychlosti tisku. Obzvlášť na dopisech musí být tato rychlost vysoká a při použití klasických čárových kódů by se jen velmi těžko dala dodržet kvalita. Velmi rozšířeným zástupcem je kód *Postnet*, který využívá americká pošta. Postnet je numerický kód využívající 2 dlouhé čary a 3 krátké pro zakódování jednoho znaku. Přklad naleznete na obrázku 3.15. [13]

Obrázek 3.15: Příklad kódu Postnet obsahující data “12345”.

Kapitola 4

Skládané dvoudimenzionální kódy

Skládané dvoudimenzionální kódy byly vyvinuty kvůli potřebě redukovat prostor nutný pro zápis lineárního čárového kódu. U klasického lineárního kódu se při větší délce úměrně zvětšuje i jeho výška, která by měla být minimálně 15%. Při velkém počtu znaků tak kód zabírá i velkou plochu. Dr. David Allais navrhl rozdělit dlouhý lineární kód na několik kratších a poskládat je pod sebe. Jednotlivé řádky tak jsou kratší a i výška jednotlivých řádků může být menší. Výsledný kód má ve výsledku měnší plochu. [13]

S tímto konceptem ale přichází některé problémy, které musí daný typ kódu umět řešit:

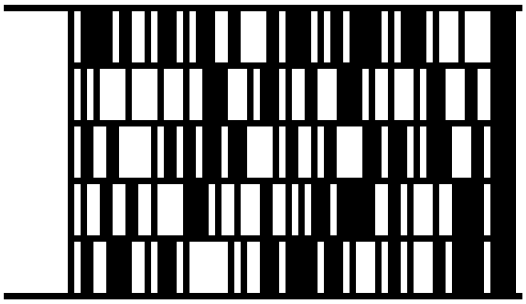
- Jak sledovat jednotlivé řádky.
- Jak zpracovat čáry, které procházejí více řádky.
- Jak najít a opravit případné chyby.

4.1 Code 49

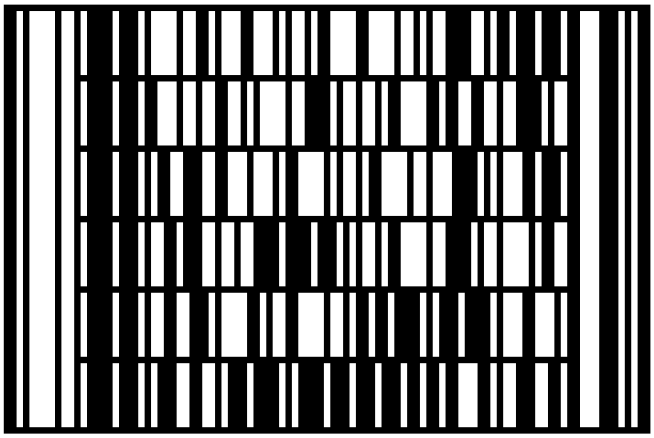
Kód Code 49 byl představen firmou Intermec v roce 1987 jako unikátní kód vhodný pro označování malých předmětů a byl vůbec prvním skládaným kódem. Může obsahovat 2–8 řádků oddělených linkou širokou jeden modul. Počet řádků závisí na počtu kódovaných dat a na komprimační metodě, která je případně použita. Každý řádek obsahuje START znak, 4 tzv. "slova", která jsou kódována ve formátu (16, 4) a STOP znak. Jednotlivé řádky jsou složeny z 18 čar a 70 modulů. Tichá zóna musí být nalevo široká minimálně 10 modulů a napravo 1 modul. Každý řádek obsahuje informaci o svém čísle, resp. pořadí. Poslední řádek pak navíc obsahuje informaci o celkovém počtu řádků a kontrolní znaky. Díky těmto informacím nezáleží na pořadí, ve kterém jsou řádky naskenovány. Čtečka data posílá až v momentě, kdy naskenuje všechny řádky. Kód Code 49 umožňuje kódovat 49 různých znaků, díky speciálním přepínacím (SHIFT) znakům, může pak kódovat celou ASCII tabulku. Příklad kódu je uveden na obrázku 4.1. [13, 2]

4.2 Codablock F

Codablock F je založen na lineárním čárovém kódu Code 128. Umožnuje tedy kódovat stejně jako on celou ASCII tabulku. Codablock F může obsahovat 2–44 řádků po 4–62 znacích. Jednotlivé řádky jsou odděleny linkou o šířce jednoho modulu. Kód Codablock F umožňuje vysokou hustotu zápisu, není ale tak efektivní jako ostatní skládané kódy. Ukázka tohoto kódu je na obrázku 4.2. [6, 13]



Obrázek 4.1: Příklad kódu 49 obsahující řetězec “TOMAS HORACEK”.



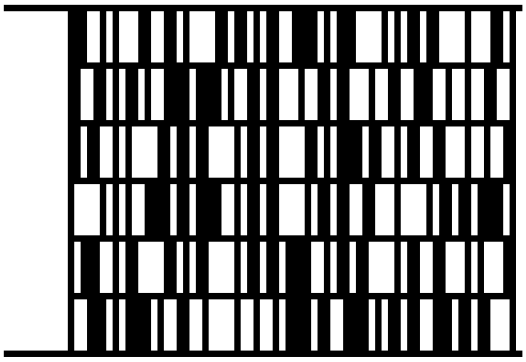
Obrázek 4.2: Příklad kódu Codablock F obsahující řetězec “Fakulta Informatiky”.

4.3 Code 16K

Code 16K je z fyzického hlediska velmi podobný jako Code 49, narozdíl od něj však umožňuje kódovat data až do 16 řádků. Ke kódování využívá, stejně jako Codablock F, lineární čárový kód Code 128, ten je ale invertovaný, tj. čáry jsou bílé a mezery černé. Každý řádek obsahuje speciální START a STOP znak, ten umožnuje rozlišit jednotlivé řádky a směr skenování. Na konci kódu jsou zakódovány dva znaky s kontrolním součtem, jednotlivé řádky ale kontrolní součet nemají. První znak na prvním řádku udává celkový počet řádků. Kód 16K umožňuje, oproti kódu 49, zakódovat větší množství dat díky vetšímu počtu řádků. Pokud bychom ale využili maximálně 8 řádků, je lepší použít kód Code 49, který má v tomto případě větší kapacitu. Příklad je uveden v obrázku 4.3. [6, 13]

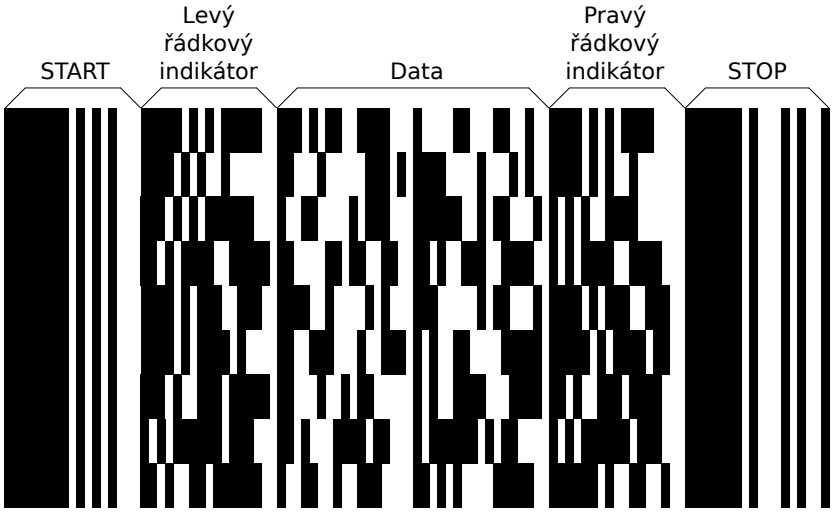
4.4 PDF417

Kód PDF417 byl představen v roce 1990 společností Symbol Technologies. Číslo 417 vyjadřuje způsob kódování jednotlivých znaků — (17, 4), kdy jeden kódovaný znak tvoří 17 modulů a 4 čáry resp. mezery. Kód PDF417 se z části podobá ryze dvoudimenzionálnímu kódu Data Matrix, protože na rozdíl od předešlých skládaných kódů nejsou jednotlivé řádky odděleny separační linkou. Tento kód využívá také menší velikosti modulů, což umožňuje na menší plochu zakódovat víc informací. Kód je navazující, má variabilní délku a počet řádků může být 3–90. Každý řádek obsahuje START znak, levý řádkový indikátor, 1–30 datových kódových znaků, pravý řádkový indikátor a STOP znak. Počet datových znaků



Obrázek 4.3: Příklad kódu 16K obsahující řetězec “xhorac00@stud.fit.vutbr.cz”.

na jeden řádek se volí libovolně podle konkrétního způsobu použití kódu. Struktura kódu je znázorněna na obrázku 4.4.



Obrázek 4.4: Struktura kódu PDF417. Kód obsahuje řetězec “Vysoke Uceni Technicke”.

Při kódování jednotlivých řádků se používají tří vzájemně se vylučující kódové sady, kterým se říká “clustery”. Každý cluster umožňuje kódovat 929 zůzných hodnot. Sousedící řádky jsou kódovány vždy odlišným clusterem, díky tomu tyto řádky může čtečka jednoduše odlišit a není potřeba separační linky mezi řádky. Data mohou být uložena ve třech různých módech — textovém, binárním a numerickém. Mezi těmito módy je možné se pomocí speciálních znaků přepínat a tím optimalizovat celkovou velikost kódu.

Kód PDF417 obsahuje povinná kódová slova pro korekci chyb. Počet těchto kódových slov je dána úrovní chybové korekce, která je 0–8. Počet korekčních kódových slov N v závislosti na úrovni L je dán následujícím vztahem.

$$N = 2^{L+1} \tag{4.1}$$

Chybová korekční kódová slova se používají k řešení dvou typů problému: pokud se znak nepodaří dekódovat, nebo pokud je pozice a hodnota kódovaného znaku neznáma. K výpočtu korekčních kódových slov se využívá Reed-Solomon algoritmus, který je blíže popsán v kapitole věnující se QR kódu. Doporučený level pro korekci chyb v závislosti na počtu datových kódových slov je uveden v tabulce 4.1.

Tabulka 4.1: Doporučená úroveň chybové korekce v závislosti na počtu datových kódových slov.

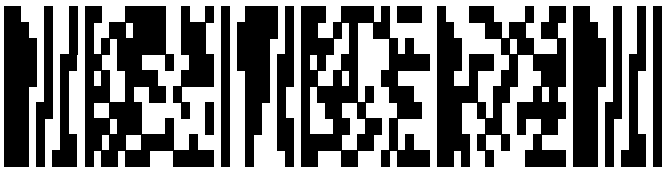
Počet datových kódových slov	Úroveň
1–40	2
41–160	3
161–320	4
321–863	5

4.4.1 Varianty kódu PDF417

Díky velkému rozšíření kódu PDF417 postupně vzniklo pár jeho variant.

Zkrácený kód PDF417 Všude tam, kde nehrozí poškození štítku s kódem můžeme použít zkrácený kód PDF417. Ten se od klasického kódu liší absencí pravého řádkového indikátoru a zkrácením STOP znaku pouze na jeden modul. Tato změna umožní velkou úsporu místa pro samotný kód.

MicroPDF417 Koncem devadesátých let dvacátého století byl firmou Symbol Technologies představen kód MicroPDF417, jež vychází ze standardního kódu PDF417 a je mu velice podobný. Tento kód je optimalizovaný pro označování malých položek, na které by se klasický kód PDF417 nevešel. MicroPDF417 umožňuje zakódovat až 150B, 250 alfanumerických znaků, nebo 366 jednociferných čísel.



Obrázek 4.5: Příklad kódu MicroPDF417 obsahující řetězec “Vysoke Uceni Technicke”.

4.5 Další skládané čářové kódy

Datastrip Datastrip je kód vyvinutý stejnojmennou společností, který se používá hlavně na různých osobních kartách (ID), v armádě, školství, lékařství a dalších odvětvích. Umožnuje kódovat informace o pacientovi, biometrické informace atd. Kód poskytuje vysokou podporu zabezpečení. Přesná specifikace kódu není bohužel volně dostupná. [13, 5]

UltraCode Je kód vytvořený společností Zebra Technologies, který měl vyřešit problémy s označováním zásilek. Kód má velký poměr šířky k výšce. Může být jako ostatní černobílý, nebo barevný. Kód využívá Reed-Solomonův algoritmus pro výpočet chybové korekce. Umožnuje zakódovat až 16-ti bitové Unicode znaky. [13, 17]

Kapitola 5

Dvoudimenzionální kódy

Dvoudimenzionální kódy označované zkráceně 2-D kódy, nebo plošné kódy, vznikly z potřeby ukládat větší množství informací na stejné, nebo menší ploše jako lineární kódy, popřípadě skládané kódy. Ve výsledku je tak oproti klasickým čárovým kódům informační hustota obrovská. Daní za to je větší komplikovanost kódu, takže už není možné je jednoduše dekódovat, čtecí zařízení musí být daleko sofistikovanější. Jestliže lineární kódy nesly nejčastěji klíč, podle kterého se daly dohledat konkrétní informace, tak dvoudimenzionální kódy nesou nejčastěji informace samotné bez potřeby je jakkoli dohledávat.

V následujících podkapitolách si podrobně popíšeme v současné době nejrozšířenější QR kód a představíme další užívané druhy dvoudimenzionálních kódů.

5.1 QR kód

Tato podkapitola je zpracovaná převážně podle [1] a [13].

QR kód je jeden z nejrozšířenějších dvoudimenzionálních kódu. V roce 1994 jej představila japonská firma Denso Wave. Data jsou uložena ve čtvercové matici, která obsahuje tři zaměřovací vzory. Tmavý modul reprezentuje binární 1 a světlý binární 0. Mezinárodní standard pro tento kód nese označení *ISO/IEC 18004* a byl publikován v roce 2000.

5.1.1 Základní charakteristika

Pozice modulu

Jednotlivé moduly jsou umístěny ve čtvercové mřížce a jejich pozice se označuje pomocí souřadnic (y, x). Souřadnice x udává umístění ve vodorovném směru a souřadnice y v horizontálním. Číslování souřadníc začíná nulou. Modul s označením $(0, 0)$ leží úplně vpravo nahoře.

Verze

Verze kódu se zapisuje ve formátu *V-E*, kde *V* označuje samotnou verzi, nabývající hodnot 1–40 a *E* označuje úroveň použité chybové korekce. Verze je přímo spjatá s velikostí kódu. Nejmenší verze 1 má velikost 21x21 modulů, verze 40 pak 177x177. S každou následující verzí roste velikost kódu o 4 moduly.

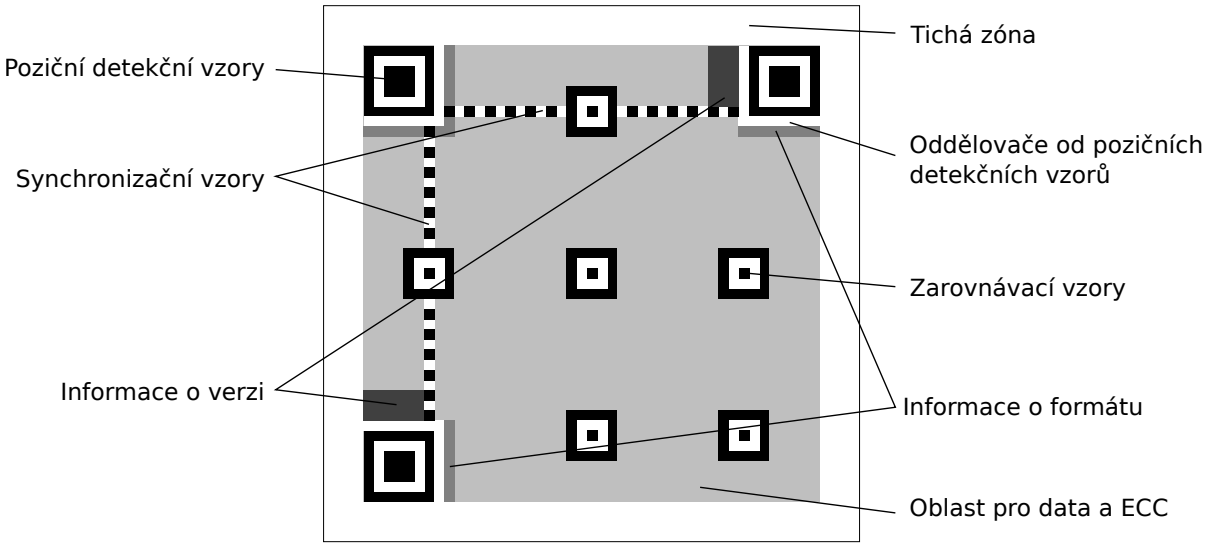
Módy

Data je možné uložit pomocí různých módů. V této práci se budu věnovat těm nejdůležitějším:

- *Numerický* — umožňuje kódovat číslice 0–9.
- *Alfanumerický* — kóduje číslice 0–9, velká písmena A–Z a znaky "mezera,$%*+-./:);".
- *8 bitový* — kóduje data jako sekvenci 8 bitů.

5.1.2 Struktura kódu

Každý kód je složen z několika typů oblastí, které mají přesně definovaný význam a použití. *Kódovací oblast* obsahuje data, ECC[1] a informace o verzi formátu. Dále kód obsahuje pevně dané funkční vzory, mezi které patří *poziční detekční vzory*, *zarovnávací vzory*, *synchronizační vzory* a *tichou zónu*. Struktura kódu je znázorněna na obrázku 5.1.



Obrázek 5.1: Struktura QR kódu verze 7.

Poziční detekční vzory jsou vždy tři, umístěné v levém spodním, levém horním a pravém horním rohu. Každý detekční vzor tvoří tři soustředné čtverce — největší tmavý o velikosti 7x7 modulů, menší světlý 5x5 modulů a nejmenší tmavý 3x3 moduly. Tyto vzory musí být odděleny od oblasti pro data a ECC z každé strany pásem o šířce jeden modul.

Každý symbol obsahuje horizontální a vertikální *synchronizační vzor*. Tvoří jej přerušované čáry o šířce 1 modul umístěné na 6. řádku, resp. sloupci.

Od verze 2 obsahuje symbol *zarovnávací vzory*. Tvoří je opět 3 soustředné čtverce, kde největší tmavý má velikost 5x5 modulů. Menší světlý čtverec je velký 3x3 moduly a nejmenší tmavý 1x1 modul. Rozmístění a počet zarovnávacích vzorů záleží na verzi kódu. Konkrétní hodnoty jsou uvedeny v tabulce E.1 z [1].

Kolem symbolu by měla být tzv. *tichá zóna* o šírce 4 modulů, která je světlá a neobsahuje žádnou jinou grafiku apod.

[1] Error Correction Codewords (korekční chybová kódová slova).

5.1.3 Vytvoření kódu

V této části si popíšeme, jaké kroky je nutné provést pro převedení vstupních dat do finálního QR kódu.

1. Analýza vstupních dat
2. Kódování dat
3. Výpočet ECC
4. Sestavení výsledné datové části
5. Umístění datových modulů do matice
6. Maskování
7. Vložení informací o formátu a verzi.

Analýza vstupních dat

Při analýze vstupních dat určujeme, jaký mód pro kódování se použije. Na základě vybraného módu a zvoleného ECC levelu najdeme dle tabulky 7–11 z [1] verzi, se kterou budeme dále pracovat.

Kódování dat

Po analýze vstupních dat, vybrání módu a verze se tato data musí převést na tok bitů. Převod se řídí určitými pravidly, která jsou pro každý mód mírně odlišná. Bitový tok je složen z následujících částí:

- indikátor módu (4 bity),
- indikátor počtu znaků,
- datový bitový tok
- ukončovací 0.

Indikátor módu má vždy 4 bity, hodnoty jsou uvedeny v tabulce 5.1. Indikátor počtu znaků udává počet znaků vstupního textu, který budeme kódovat. Jeho délka závisí na módu a verzi. Příslušné hodnoty jsou uvedeny v tabulce 5.2. Protože se bitový datový kód vytváří pro každý mód mírně odlišně, popíšeme si jeho tvorbu pro jednotlivé módy zvlášť v následujících odstavcích.

Ukončovací nula se ukládá za datovou část na čtyřech bitech. Délka bitového datového toku je poté zarovnána na 8 bitů, přidáním případných nul (délka musí být dělitelná 8-mi). Protože se data v QR kódu ukládají do 8 bitových kódových slov, musíme celý bitový řetězec rozdělit na kódová slova po 8 bitech. Datová oblast QR kódu musí být plně zaplněná, zbývající místo se doplní přidáním střídavě čísly 236 a 17. Kapacita dat je pro jednotlivé verze QR kódu uvedena v tabulkách 7–11 z [1].

Tabulka 5.1: Indikátory módu.

Mód	Indikátor
Numerický	0001
Alfanumerický	0010
8-mi bitový	0100

Tabulka 5.2: Počet bitů pro indikátor počtu znaků.

Verze	Numerický mód	Alfanumerický mód	8-mi bitový mód
1–9	10	9	8
10–26	12	11	16
27–40	14	13	16

Numerický mód Vstupní řetězec je v tomto případě rozdělen do skupin po třech číslicích. Každá skupina je překonvertována na její desetibitový ekvivalent. Pokud je počet číslic ve skupině 2 nebo 1, konvertuje se tato skupina na binární ekvivalent o čtyřech, resp. sedmi bitech. Délka bitového toku pro numerický kód je dána následujícím vztahem.

$$B = 4 + C + 10(D/3) + R \tag{5.1}$$

Kde B je celkový počet bytů, C — počet bitů indikátoru počtu znaků, D — počet vstupních znaků. $R = 0$ pokud $(D\%3) = 0$, $R = 4$ pokud $(D\%3) = 1$ a $R = 7$ pokud $(D\%3) = 2$. Symbol / označuje operaci celočíselné dělení a symbol % operaci modulo.

Příklad 5.1 Zakódování vstupního řetězce “0123456789” na tok bitů při použité verzi kódu 1-L.

1. Vstupní řetězec rozdělíme na skupinu po 3 znacích: `012 345 678 9`
2. Každou skupinu převedeme na její binární ekvivalent:
   `012` → `0000001100`
   `345` → `0101011001`
   `678` → `1010100110`
   `9` → `1001`
3. Spojení do sekvence: `0000001100 0101011001 1010100110 1001`
4. Indikátor počtu znaků: `10` → `0000001010`
5. Přidání indikátoru módu `0001` a indikátoru počtu znaků před datové bity:
   `0001 0000001010 0000001100 0101011001 1010100110 1001`

Alfanumerický mód Alfanumerický mód je schopen kódovat 45 různých znaků, které jsou číslovány od 0, výčet je uveden v tabulce 5.3. Vstupní řetězec je rozdělen na skupiny po dvou znacích, které jsou zapsány ve svých číselných ekvivalentech. První číslo z každé skupiny je vynásobeno číslem 45 a k tomu je přičteno druhé číslo. Výsledek je pak převeden na 11 bitové číslo. Pokud je znak osamocen (netvoří dvojici), je převeden na svůj 6-ti bitový ekvivalent. Celý bitový tok je pak tvořen opět indikátorem módu, indikátorem počtu znaků a daty.

Tabulka 5.3: Kódovací tabulka pro Alfanumerický mód.

Znak	Č.	Znak	Č.	Znak	Č.	Znak	Č.
0	0	C	12	O	24	mezera	36
1	1	D	13	P	25	$	37
2	2	E	14	Q	26	%	38
3	3	F	15	R	27	*	39
4	4	G	16	S	28	+	40
5	5	H	17	T	29	-	41
6	6	I	18	U	30	.	42
7	7	J	19	V	31	/	43
8	8	K	20	W	32	:	44
9	9	L	21	X	33		
A	10	M	22	Y	24		
B	11	N	23	Z	35		

Příklad 5.2 Zakódování vstupního řetězce "TH-87" na tok bitů při použité verzi kódu 1-L.

1. Rozdělení na skupiny po dvou znacích a převedení na číselné ekvivalenty:
   TH-87 → (T,H) (-,8) (7) → (29,17) (41,8) (7)

2. Každou skupinu převedeme na její binární ekvivalent:
   (29,17) → 29*45+17 → 1322 → `10100101010`
   (41,8) → 41*45+8 → 1853 → `11100111101`
   (7) → 7 → `000111`

3. Spojení do sekvence: `10100101010 11100111101 000111`

4. Indikátor počtu znaků: `5` → `000000101`

5. Přidání indikátoru módu `0010` a indikátoru počtu znaků před datové bity:
   `0010 000000101 10100101010 11100111101 000111`

8 bitový mód V tomto módu jsou jednotlivé znaky kódovány přímo do 8 bitových kódových slov a nemusí již probíhat žádná konverze, tak jako u numerického a alfanumerického módu.

Výpočet ECC

QR kód povinně obsahuje tzv. chybová korekční slova, pomocí kterých je možné rekonstruovat data v poškozeném, nebo jinak nečitelném kódu. Byly definovány 4 úrovně, jež jsou schpné opravit různý objem poškozených dat. Označení úrovní a přibližná velikost opravitelných dat jsou uvedeny v tabulce 5.4.

K samotnému výpočtu se používá Reed-Solomonův algoritmus. Volba úrovně pro výpočet chybové korekce je velmi důležitá, nízká úroveň nemusí v některých aplikacích stačit, vysoká naopak výrazně snižuje kapacitu pro data.

Tabulka 5.4: Úrovně chybové korekce.

Úroveň	**Možná oprava dat [%]**
L	7
M	15
Q	25
H	30

Sestavení výsledné datové části

Tok datových bitů se musí rozdělit do několika bloků, jejichž velikost a počet jsou definovány v tabulkách 13–22 z [1]. Po rozdělení se ke každému bloku spočítají příslušná ECC.

Výsledná posloupnost kódových slov se sestavuje nejdříve z datových bloků a za ně se umístí bloky ECC. Kódová slova se za sebe umisťují střídavě a to takto: 1. kódové slovo z 1. bloku, 1. kódové slovo z druhého bloku atd. Chybová kódová slova se umisťují obdobně jako datová. Příklad rozdělení do bloků je uveden v tabulce 5.5.

Tabulka 5.5: Rozdzdělení kódových slov do bloků.

	Datová kódová slova					**Datová kódová slova**			
Blok 1	D1	D2	...	D11		E1	E2	...	E22
Blok 2	D12	D13	...	D12		E23	E24	...	E44
Blok 3	D23	D24	...	D33	D34	E45	E46	...	E66
Blok 4	D35	D36	...	D45	D46	E67	E68	...	E88

Pro datová kódová slova a ECC z tabulky 5.5 bude výsledná sekvence následující: D1, D12, D23, D35, D2, D13, D24, D36, ... D11, D22, D33, D45, D34, D46, E1, E23, E45, E67, E2, E24, E46, E68, ... E22, E44, E66, E88 (rozdělení je převzato z [1]).

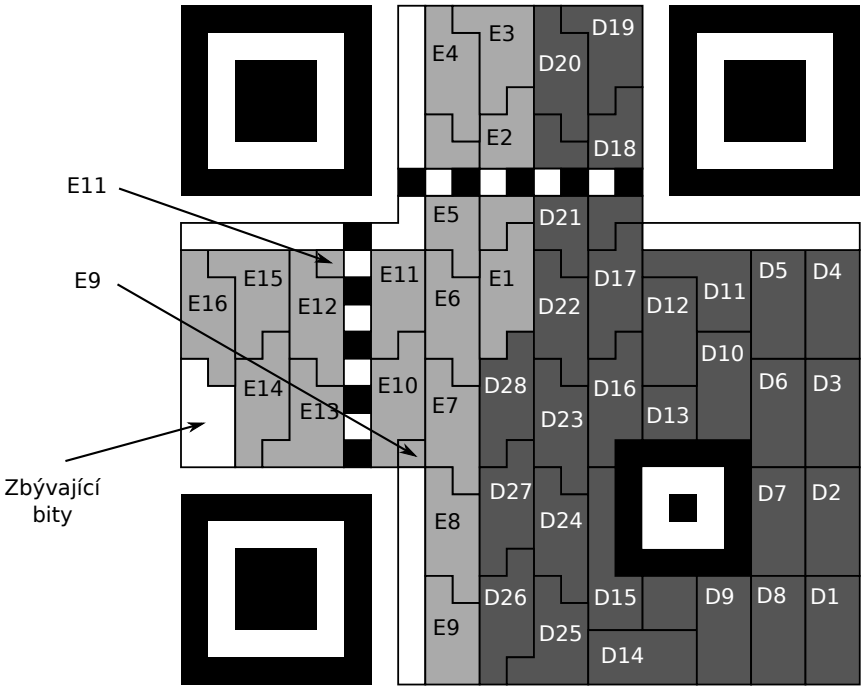
Umístění datových modulů do matice

Jednotlivá kódová slova jsou do matice umisťována nejčastěji do obdelníkového prostoru 2x4 moduly — horizontálně, nebo vertikálně. V případě, že narazíme na poziční detekční vzory nebo zarovnávací vzory, může se tvar prostoru pro umístění kódového slova změnit. Kódová slova se do matice začínají vkládat v pravém dolním rohu směrem nahoru. Jakmile už se nedá jít dál nahoru, otočí se směr a kódová slova se vkládají směrem dolů. Obdobně se postupuje až do úplného zaplnění celé matice. Příklad umístění kódových slov pro QR kód verze 2-M je na obrázku 5.2.

Maskování

Maskování se provádí z důvodů vyvážení počtu a rozmístění tmavých a světlých modulů. To vede k lepší čitelností kódu. Maska se aplikuje jen na oblast pro data a ECC.

Proces probíhá tak, že se matice vyplněná kódovými slovy postupně překryje každou z 8 různých dostupných masek. Nad každým bodem matice se provede operace XOR. Výsledky po všech maskováních se mezi sebou porovnají a na základě hodnotících kritérií se vybere nejvhodnější maska.



Obrázek 5.2: Umístění kódových slov v datové oblasti QR kódu.

Vložení informací o formátu a verzi

Informace o formátu a verzi se do kódu vkládají až po aplikaci masky. Informace o formátu je 15 bitová sekvence obsahující 5 datových bitů a 10 chybových korekčních bitů. Datová část je složena z dvoubitové informace o použité chybové korekční úrovni a tříbitové informace o použité masce.

Informace o verzi je 18 bitová sekvence obsahující 6 datových bitů a 12 bitů pro korekci chyb. Protože se informcace o verzi používá pro verze kódu 7-40, obsahuje datová část číslo 7–40.

5.1.4 Příklad tvorby QR kódu

V této podkapitole si ukážeme krok po kroku sestavení QR kódu s textem "Tomáš Horáček" a úrovní chybové korekce L. Budeme postupovat podle návodu z předchozí podkapitoly.

1. **Analýza vstupních dat**

   Vstupní text budeme kódovat v 8 bitovém módu. Počet znaků je *17* — znaky "š" a "č" mají délku 2, jelikož nejsou v kódování UTF-8 součástí základní ASCII tabulky, ale jsou "zdvojené". Pro počet znaků 17 a úroveň chybové korekce L je nejmenší možná verze kódu *1*.

2. **Kódování dat**

   Indikátor 8 bitového módu je `0100`. Pro tento mód ve verzi 1 je indikátor počtu znaků uložen na 8 bitech — `00010001`.

   Samotná data se ukládají jako ASCII hodnoty jednotlivých znaků na 8 bitů.
   T $\rightarrow$ 84 $\rightarrow$ `01010100`

o → 111 → `01101111`
⋮
k → 107 → `01101011`

Nakonec se za data vloží na čtyř bitech ukončovací 0. Celý bitový tok má v tuto chvíli následující podobu.

```
0100 00010001 01010100 01101111 01101101 11000011 10100001 11000101
10100001 00100000 01001000 01101111 01110010 11000011 10100001
11000100 10001101 01100101 01101011 0000
```

Nyní řetězec bitů přeskupíme na kódová slova a doplníme případné nuly.

```
01000001 00010101 01000110 11110110 11011100 00111010 00011100
01011010 00010010 00000100 10000110 11110111 00101100 00111010
00011100 01001000 11010110 01010110 10110000
```

Protože bitový tok právě využívá 19 kódových slov, což je maximální kapacita pro data kódu verze 1-L, nepřidáváme žádná vyplňovací kódová slova.

3. **Výpočet ECC**

   Ve verzi 1-L se data nerozdělují na žádné bloky, resp. obsahují pouze jeden blok, a proto můžeme ECC spočítat přímo. Pro výpočet se používá Reed-Solomonův algoritmus, který je však značně náročný a proto ho tu nebudu uvádět.

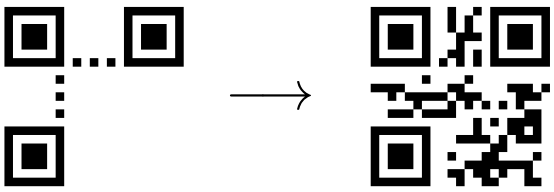
   ```
   11100101 00001000 01001110 11001100 00111001 11000111 11000000
   ```

   ECC přidáme nakonec datové části a dostaneme tak 26 kódových slov, což je kapacita QR kódu verze 1.

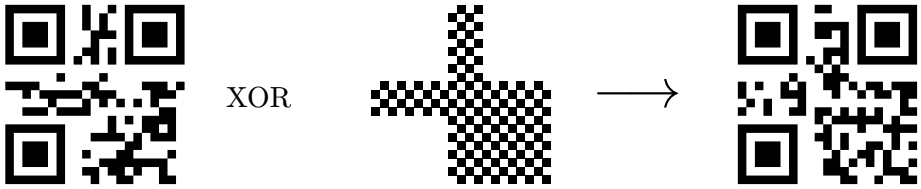
   ```
   01000001 00010101 01000110 11110110 11011100 00111010 00011100
   01011010 00010010 00000100 10000110 11110111 00101100 00111010
   00011100 01001000 11010110 01010110 10110000 11100101 00001000
   01001110 11001100 00111001 11000111 11000000
   ```

4. **Umístění datových modulů do matice**

   Nyní budeme vkládat kódová slova do matice o velikosti 21x21 modulů. Bit s hodnotou 1 bude tvořit tmavý modul a bit s hodnotou 0 světlý. Vložení kódových slov do prázdného symbolu (obsahujícího pouze funkční vzory) je zobrazen na obrázku 5.3.



Obrázek 5.3: Umístění datových modulů.



Obrázek 5.4: Proces maskování.

5. **Maskování**

   Proces maskování maskou s referencí `000` je znázorněn na obrázku 5.4.

6. **Vložení informací o formátu a verzi**

   Nakonec celého procesu vložíme do symbolu informace o formátu. Protože se jedná o verzi kódu 1, informace o verzi se do něj nevkládá. Výsledek je zobrazen na obrázku 5.5.

Obrázek 5.5: Výsledný QR kód obsahující data “Tomáš Horáček”.

5.1.5 Ukládání strukturovaných dat

S technickým rozvojem mobilních telefonů v první dekádě 21. století se QR kód začal stále více prosazovat na poli mobilního marketinku a při výměně dat. Bylo ale potřeba stanovit nějaký standardizovaný formát, jak tyto data do kódu ukládat. Společnost NTT DoCoMo standardizovala několik formátů pro formátování URL adresy, kontaktních informací, a dalších. Těchto různých formátů existuje velké množství a záleží jen na čtečce, jestli je dokáže správně interpretovat. Kontaktní informace mohou být zapsány například takto:

```
MECARD:
    N:     Tomáš Horáček;
    TEL:   +420732xxxxxx;
    EMAIL: xhorac00@stud.fit.vutbr.cz;
;
```

Čtečka v mobilním telefonu tento formát rozezná a nabídne uložit kontakt do adresáře. Jednoduše se dá vytvořit i odkaz na webovou stránku, což je nejčastejší způsob použití QR kódu v reklamě:

```
MEBKM:
    TITLE: Fakulta informačních technologií VUT v Brně;
    URL:   http://www.fit.vutbr.cz;
;
```

V tomto případě může čtečka otevřít odkaz rovnou v prohlížeči. [12, 27]

5.2 Další dvoudimenzionální kódy

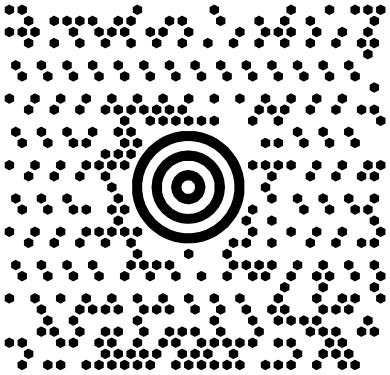
5.2.1 Data Matrix

Kód Data Matrix byl představen v roce 1994 společností Rvsi Acuity Cimatrix Inc., nyní má na něj práva firma Siemens. Kód má maximální kapacitu 3116 numerických, nebo 2335 alfanumerických znaků. Poznávacím znakem jsou dvě plné čáry na levé a spodní straně a přerušované čáry nahoře a vpravo. Stejně jako QR kód využívá Reed-Solomonovo kódování pro korekci chyb. Použití nalézá hlavně při označování elektronických součástek, ale také v armádě, logistice, letecké přepravě nebo v mobilním marketingu. Ukázka kódu Data Matrix je na obrázku 5.6. [9, 19, 13]

Obrázek 5.6: Ukázka kódu Data Matrix obsahujícího odkaz na stránku http://www.barcodepack.com.

5.2.2 Maxicode

Kód Maxicode vyvinula v roce 1993 americká logistická společností UPS pro automatické třídění a sledování zásilek. Maxicode má pevně dané rozměry celého symbolu i modulu, což umožňuje velmi rychlé čtení na dopravníkových pásech. Tento kód můžeme snadno rozeznat díky třem soustředným kružnicím ve středu symbolu, které slouží k detekci kódu. Moduly mají tvar 6 úhelníku, což umožňuje mít hustější strukturu. Kapacita kódu je 93 znaků. Maxicode využívá Reed-Solomonův algoritmus pro detekci a opravu chyb. Příklad kódu Maxicode je na obrázku 5.7. [13, 2]



Obrázek 5.7: Ukázka kódu Maxicode obsahujícího text “Fakulta Informatiky”

Kapitola 6

Návrh a implementace

V následující části bakalářské práce se zabývám popisem implementace knihovny pro generování čárových kódů. Snahou bylo vytvořit knihovnu vhodnou pro nasazení v prostředí webových informačních systémů. Při návrhu byl kladen důraz na její snadné použití a instalaci, uživatel by neměl být zbytečně obtěžován zjišťováním její vnitřní implementace.

Knihovna implementuje několik lineárních (EAN/UPC, Code 128, kód Industrial 2/5 a kód Interleaved 2/5) čárových kódů a QR kód jako zástupce dvoudimenzionálních kódů. Každý z kódů lze použít samostatně díky rozdělení na příslušné soubory, ve kterých jsou třídy daného kódu.

Jako implementační jazyk jsem zvolil na webu dnes nejrozšířenější PHP[1]. Díky tomu může být knihovna integrována do velkého množství již hotových informačních systému, jež běží právě na PHP.

Knihovna využívá plně objektový návrh PHP verze 5.2., který je nutný pro běh knihovny. Dále je vyžadováno rozšíření PHP v podobě GD[2] knihovny.

6.1 Struktura knihovny

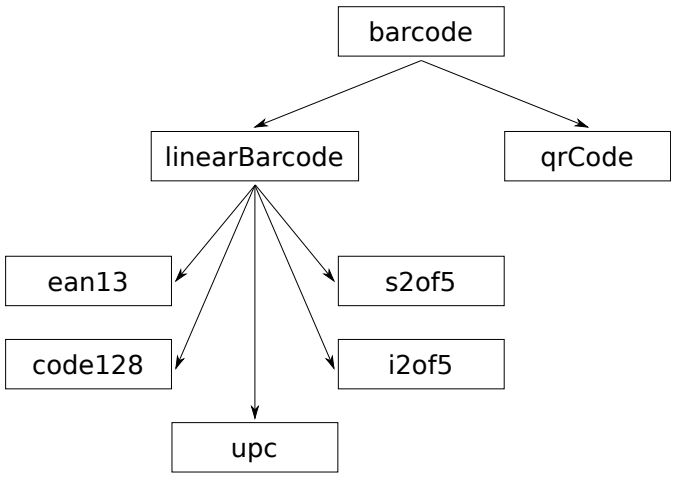
Každý typ čárového kódu je tvořen samostatnou třídou pojmenovanou po tomto kódu. Třídy jsou pak v samostatných souborech pojmenovaných ve tvaru `class.nazevKodu.php`. Seznam souborů a jim odpovídající třídy je uveden v tabulce 6.1. Součástí tříd jsou i kódovací tabulka a další potřebné informace nutné k sestavení kódu.

Tabulka 6.1: Seznam souborů knihovny a odpovídajících tříd.

Název souboru	**Název třídy**	**Popis**
`class.barcode.php`	`barcode`	Společná třída pro všechny ostatní třídy.
`class.linarBarcode.php`	`linearBarcode`	Společná třída pro lineární čárové kódy.
`class.qrCode.php`	`qrCode`	Třída generující QR kód.
`class.s2of5.php`	`s2of5`	Třída generující kód Industrial 2/5.
`class.i2of5.php`	`i2of5`	Třída generující kód Interleaved 2/5.
`class.code128.php`	`code128`	Třída generující kód Code 128.
`class.ean13.php`	`ean13`	Třída generující kód EAN 13.
`class.upc.php`	`upc`	Třída generující kód UPC.

[1] Hypertext Preprocessor
[2] Graphics Draw

Schéma závislostí tříd je znázorněn na obrázku 6.1. Nejvýše stojí třída `barcode` a pod ní jsou její potomci. Podrobný popis všech tříd je uveden v následující kapitole.



Obrázek 6.1: Schéma závislostí tříd.

6.2 Popis tříd

6.2.1 Třída barcode

Tato třída je rodičem všech tříd. Ostatní třídy jsou jejím přímým nebo nepřímým potomkem. V konstruktoru třídy se ověřují vstupní parametry společné pro všechny třídy. V případě, že jsou vstupní parametry špatné, vyvolá se příslušná výjimka.

6.2.2 Třída linearBarcode

Třída *linearBarcode* obsahuje metody a proměnné společné pro všechny typy lineárních čárových kódů které jsem implementoval. V konstruktoru se volá metoda `checkAllowedChars()`, která ověřuje, jestli vstupní řetězec obsahuje pouze povolené znaky.

Metoda `draw()` vytvoří grafickou reprezentaci čárového kódu. Metoda postupně projde pole `$this->biteCode` a podle indexu vykreslí buď delší funkční čáry nebo kratší datové čáry. Pod samotným symbolem s kódem vykresluje příslušný text kódu.

Metoda `rawData()` vrací čárový kód jako řetězec jedniček a nul, kde číslo jedna reprezentuje tmavý modul a nula světlý modul.

6.2.3 Třída code128

V konstruktoru třídy `code128` se nejprve naplní privátní proměnné kódových sad příslušnými znaky a poté je zavolána metoda `createBiteCode()`, která provede konverzi vstupního řetezce na daný kód. Metoda podle úvodního znaku nejdříve najde počáteční kódovou sadu a poté začíná kódovat znak po znaku. Po každém zakódovaném znaku se kontroluje kódová sada pro další znak. Pokud se kódová sada liší od předešlé, přepne se pomocí speciálního znaku SHIFT.

Během kódování je průběžně počítán vážený součet daných znaků a po zakódování celého vstupního řetězce je vypočítán kontrolní součet. Ten je spolu se STOP znakem a ukončovacím znakem vložen na konec zakódovaného řetězce.

6.2.4 Třída s2of5

Třída `s2of5` generuje kód typu Standard 2/5 (Industrial 2/5). V metodě `createBiteCode()` se v cyklu prochází vstupní řetězec a každý znak je podle kódovací tabulky převeden na odpovídající kód.

6.2.5 Třída i2of5

Tato třída implementuje čárový kód Interleaved 2/5. Kódování znaků probíhá po párech a to tak, že se střídavě kódují čáry z prvního znaku páru a mezery z druhého znaku. Lichý a sudý znak páru si uchovává čítač, u které čáry resp. mezery se právě nachází.

6.2.6 Třída ean13

Třída `ean13` umožňuje vytvářet kód typu EAN-13. Zajímavá může být přetížená metoda `draw()`, která doplňuje původní metodu o vložení rozděleného textu do symbolu. První číslo je předsazené před kódem, poté je řetězec rozdělen na 6 a 6 znaků, které jsou odděleny oddělovacím znakem.

6.2.7 Třída upc

Jelikož je kód UPC prakticky totožný s EAN kódem, liší se obě třídy jen nepatrně. Před text, který se má zakódovat, je jen vložena úvodní 0 tak, aby mohla třída využít kódovací tabulku EAN kódu. Tato 0 se pak ve výstupu nevypisuje.

6.2.8 Třída qrCode

Třída qrCode zajišťuje generování QR kódu. Kromě samotného získání obrazových dat symbolu je veškerá funkcionalita prováděna v konstruktoru, nebo v metodách z konstuktoru volaných. Schéma průběhu generování je na obrázku 6.2.

Nejprve se vstupní text převede na kódování *UTF-8*, aby se zamezilo případné špatné interpretaci některých speciálních znaků. Poté je metodou `getMode` na základě shody s regulárním výrazem vrácen indikátor módu.

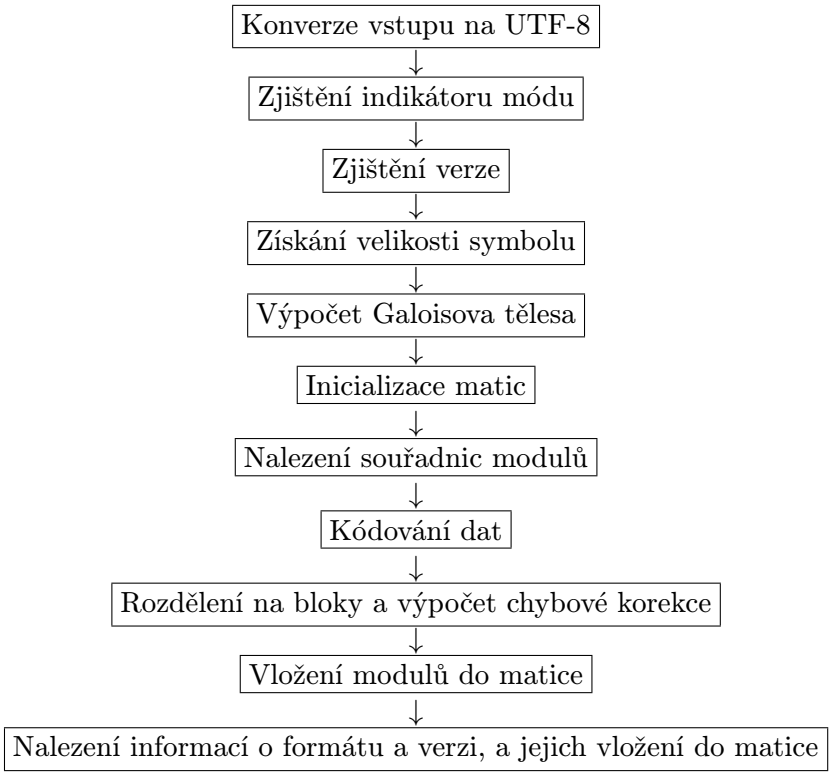
Metoda `getVersion` vrací na základě počtu vstupních znaků, úrovně chybové korekce a módu verzi kódu. Verze se zjistí podle tabulky s kapacitou pro jednotlivé verze uložené ve formě pole jako privátní proměnná.

Velikost symbolu podle verze vrací metoda `getMatrixSize` za využití jednoduchého bitového posunu.

```
private function getMatrixSize($version)
{
    return 17 + ($version << 2);
}
```

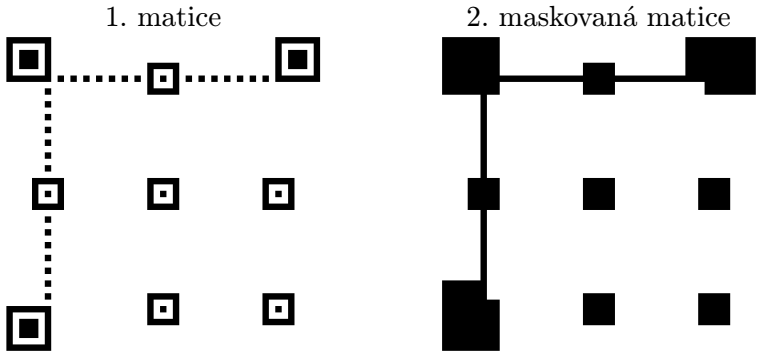
Pro výpočet Galoisova tělesa slouží metoda `countGaloisField`. Vypočtené hodnoty jsou ukládány do dvou polí z nichž druhé má prohozené hodnoty a indexy oproti tomu prvnímu. Při psaní funkce jsem se částečně inspiroval ukázkou zdrojového kódu na [14].

V metodě `init` probíhá inicializace matic. Jsou vytvořeny dvě matice. První slouží k pozdějšímu vložení modulů a druhá tzv. maskovaná matice, která se využívá při výpočtu souřadnic jednotlivých modulů. Do první matice se vkládají veškeré funkční prvky. Schéma pozičního detekčního vzoru a zarovnávacího vzoru jsou uloženy v privátních polích což



Obrázek 6.2: Schéma průběhu generování QR kódu.

usnadňuje práci při jejich vkládání do matice. Umístění pozičních vzorů je uloženo v podpůrném poli. Maskovaná matice je kromě oblasti pro data a korekční chybová kódová slova vyplněna jedničkami. Příklad obou matic je na obrázku 6.3.



Obrázek 6.3: Příklady inicializovaných matic.

Nalezení souřadnic modulů se provádí v metodě `getBitsCoordinates`. Souřadnice modulů jsou důležité pro pozdější umísťování do matice. Pro hledání se využívají nezamaskované body maskované matice. Konkrétní postup hledání je obdobný jako vkládání modulů, popsaném v podkapitole 5.1.3 a podrobněji pak na strnanách 46–49 v [1].

Stěžejním bodem celé třídy je funkce `convertData`, ve které se provádí většina hlavních důležitých operací. Na začátku této funkce je provedeno kódování dat popsané v podkapitole 5.1.3. Při kódování dat je nejzajímavější částí rozdělování na kódová slova, ve které se musí například 10 bitová čísla rozdělit na čísla 8 bitová. Významným pomocníkem tu jsou

bitové operace *AND*, *OR* a bitový posun.

Následuje část, která rozděluje kódová slova na bloky. Počet bloků je pro každou verzi uložen v poli jako privátní proměnná. Ke každému datovému bloku se zároveň vytváří blok s ECC. Tento výpočet obstarává metoda `reedSolomon`. Výpočet Reed-Solomonova kódu není z matematického hlediska triviální. Při programování mi pomohla názorná ukázka na [15]. Část algoritmu jsem pak přebral z knihovny *libqrencode* [8] a *PHP QR Code* [7].

V poslední části metody `convertData` se vkládají jednotlivé bity bloků do matice. Hodnota konkretniho bitu v kódovém slově se zjišťuje bitovým součinem kódového slova s číslem 128 bitově posunutým o pořadí zjišťovaného bitu. Jestliže je tato hodnota větší než 0, získáme 1, jinak 0. Operací *XOR* se na výslednou hodnotu aplikuje maska. Níže je ukázka kódu vkládající datový bit do matice.

```
$y = $this->bitsCoordinates[$bitCounter][0];
$x = $this->bitsCoordinates[$bitCounter][1];
$this->matrix[$y][$x] = (($dataBlocks[$j][$i] & (128>>$k)) ? 1 : 0) ^
    $this->mask($y,$x);
```

Nakonec se do matice vkládají informace o formátu a verzi. Metody `formatInformation`, `versionInformation` tyto informace naleznou a metody `addFormatInformation`, `addVersionInformation` je vloží do matice.

O příptavu obrazových dat se stará veřejná metoda `draw`. Ta po řádcích prochází vyplněnou maticí, z níž jedničkové bity zapisuje jako černé moduly a nulové bity jako bílé moduly.

Obdobně jako metoda `draw` pracuje i metoda `rawData`. Vrací kód v podobě textového výstupu, kde každý řádek představuje jeden řádek matice.

6.3 Přehled výjimek

V případě chyby knihovna vyvolá příslušnou výjimku. Číslování výjimek odpovídá třídám knihovny, třídě `barcode` tak odpovídají výjimky 100–199, třídě `linarBarcode` 200–299 atd. Výjimky č. 0–99 jsou vyhrazené pro API[3] popisované v další kapitole. Při používání knihovny je dobré tyto výjimky zachytávat a problém dále řešit. Přehled výjimek je uveden v tabulce 6.2.

6.4 Příklad použití knihovny

Použití knihovny je velmi jednoduché, stačí načíst třídu s požadovaným typem kódu, vytvořit instanci třídy s příslušnými parametry a následně zavolat veřejnou metodu `draw()`, nebo `rawdata()` pro získání obrázkových dat, resp. textový výstup daného kódu.

Následující příklad zobrazí čárový kód typu Code 128 se zakódovaným textem “Hello world!” jako obrázek ve formátu PNG.

```
<?php
include "class.code128.php";            // Načtení knihovny

$barcode = new code128('Hello world!'); // Vytvoření instance třídy
```

[3] Application Programming Interface

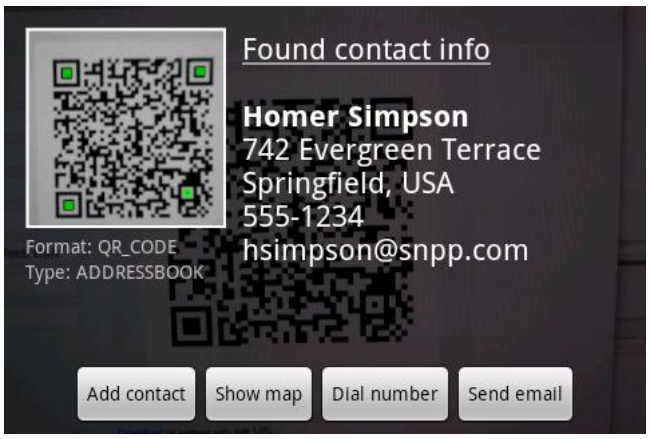
Tabulka 6.2: Kódy výjimek.

Č.	Kód	Popis
0	E_BAD_TYPE	Chyba: špatný typ kódu.
1	E_BAD_PARAMS	Chyba: špatné vstupní parametry.
2	E_BAD_OUTPUT	Chyba: špatný typ výstupu.
100	E_EMPTY_TEXT	Chyba: vstup nesmí být prázdný.
101	E_MODULE_SIZE	Chyba: velikost modulu musí být v rozmezí X–Y.
200	E_BAD_CHARS	Chyba: vstup vstup obsahuje nepovolené znaky.
500	E_ODD_LENGTH	Chyba: počet znaků musí být sudý.
600	E_BAD_EAN_LENGTH	Chyba: Délka musí být 12 znaků.
700	E_BAD_UPC_LENGTH	Chyba: Délka musí být 11 znaků.
800	E_BAD_QR_LENGTH	Chyba: delka vstupního řetězce je moc velká.
801	E_BAD_VERSION	Chyba: nelze zvolit vybranou verzi.
802	E_BAD_MASK	Chyba: špatná reference masky.

```
header('Content-type: image/png');      // Nastavení HTTP hlavičky
imagepng($barcode->draw());             // Vykreslení čárového kódu
?>
```

6.5 Čtečka čárových kódů

Pro průběžné testování knihovny jsem jako čtečku čárových kódů úspěšně využíval mobilní telefon HTC Magic se systémem Android 1.6 s aplikacemi *BarcodeScanner* [28] (ukázka na obrázku 6.4) a *QuickMark* [18]. Exituje také celá řada volně dostupných online dekodérů, které jsou schopné číst lineární i dvoudimenzionální čárové kódy. Například dekodér od RDK Software [16], nebo ZXing online dekodér [26].



Obrázek 6.4: Obrazovka programu BarcodeScanner. [28]

Kapitola 7

Demonstrační aplikace

Pro demonstraci knihovny, shrnující její schopnosti, jsem vytvořil několik ukázkových aplikací. Hlavním prvkem je webové online API, které se úspěšně využívá v dalších demo aplikacích.

7.1 Online API pro získávání čárových kódů

Webové API umožnuje dynamické generování čárových kódů pomocí parametrů v URL adrese. Uživatel tak může jednoduše vložit obrázek s čárovým kódem na své stránky, blog, či fórum a nemusí se starat o instalaci knihovny. Toto řešení tak dovoluje používat knihovnu i uživatelům, kteří programování nerozumí, ale přesto si potřebují vytvořit svůj vlastní čárový kód.

API vrací obrázek s čárovým kódem nebo jeho textovou reprezentaci jako odpověď na GET nebo POST požadavek. API je dostupné na následující URL adrese.

`http://www.barcodepack.com/api/`

Ovládání API probíhá výhradně přes parametry uvedené v tabulce 7.1 zaslané GET nebo POST požadavkem. Pro dlouhé vstupy například při vytváření QR kódu je lepší použít POST požadavek, který není narozdíl od GET požadavku limitován maximální délkou.

Tabulka 7.1: Přehled parametrů API.

Parametr	**Pov.**	**Možné hodnoty**	**Poznámka**
`type`	ano	`s2of5`, `i2of5`, `ean13`, `upc`, `code128`, `qrCode`	Typ čárového kódu.
`text`	ano		Vstupní řetězec, který se bude kódovat.
`output`	ne	`png`, `gif`, `jpg`, `rawdata`	Typ výstupu.
`moduleSize`	ne	1-10	Velikost modulu v pixelech.
`ecl`	ne	`L`, `M`, `Q`, `H`	Míra korekce chyb.
`version`	ne	1–40	Verze QR kódu.

V případě, že jsou některé parametry zadané nesprávně, vrací server HTTP kód 400 Bad Request. V hlavičce *Status* je pak předávaná chybová zpráva.

Například při volání API s parametry `type=qr&text=pokus` se pošle chybová hláška `Chyba: špatný typ kódu.`, protože kód typu "qr" není podporován.

Pro dosažení větší rychlosti celé aplikace se každý čárový kód nejprve vygeneruje do souboru a poté je poslán na výstup. Při dalším požadavku na získání stejného kódu nemusí server celý proces generování opakovat, ale odešle již vygenerovaný soubor. Název souboru je dán hashovací funkcí `md5` z předem sestaveného řetězce, který obsahuje všechny parametry daného čárového kódu a pro dané nastavení je tak vždy unikátní. Soubory se ukladají do adresáře `temp`, který musí mít nastaveny práva pro zápis. Příklady volání API jsou uvedené v příloze A.

7.1.1 Implementace API

Funkční kód je tvořen jedním souborem `index.php` umístěném v příslušném adresáři, v našem případě je to adresář `api`. Adresář dále obsahuje již zmíněnou složku `temp` a další adresář s knihovnou. Ta ale může být umístěná kdekoli.

Na začátku si skript načte všechny třídy naší knihovny, aby je pak mohl použít. API je složeno z jedné třídy `barcodePackAPI`, která obsahuje statickou metodu `getBarcode($type, $text, $params=null)`. Tato metoda vrací instanci třídy daného typu čárovéko kódu v závislosti na parametru `$type`.

Funkční kód zpracovává parametry URL a poté volá statickou metodu `getBarcode()`. Protože je společným parametrem všech tříd pouze `$text`, předává se jako třetí parametr metodě celý HTTP REQUEST v proměnné `$_REQUEST`. Parametry, které společné nejsou třídám předává výše zmíněná metoda.

7.2 Online generátor čárových kódů

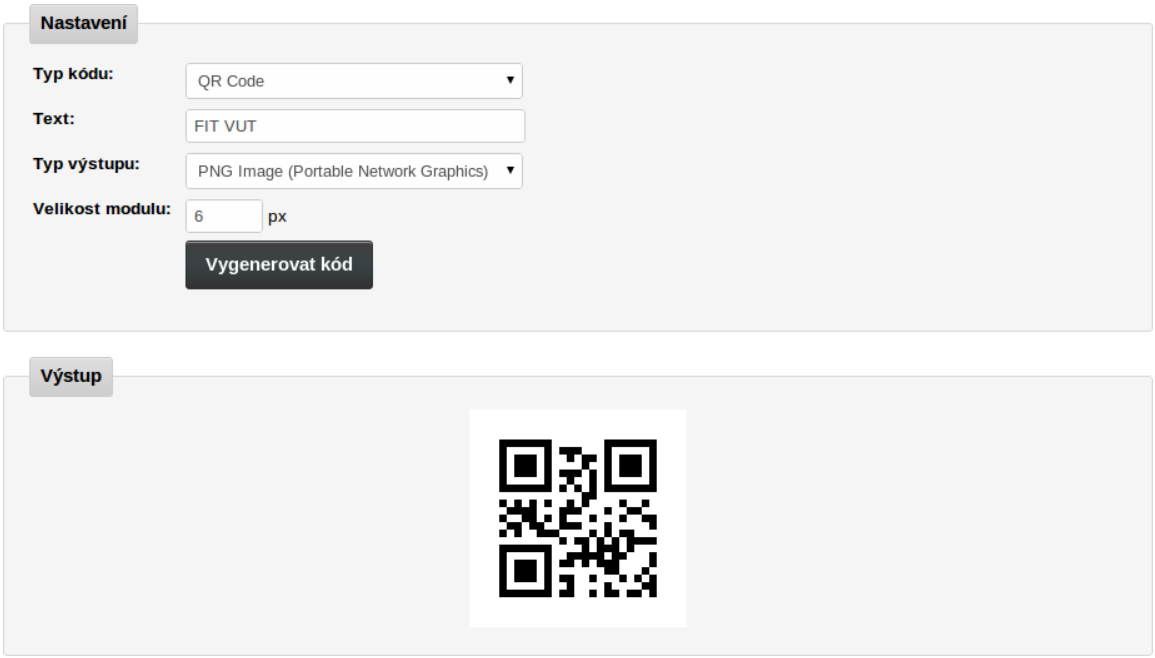
Online generátor čárových kódu je grafická nádstavba s uživatelským rozhraním nad API, popsaného v přechozí podkapitole. Generátor se skládá z formuláře, ve kterém je možné z roletkového menu vybrat typ kódu, zadat text, který se má zakódovat, vybrat typ výstupu (obrázkové formáty PNG, GIF, JPG a textová data) a zadat velikost modulu. Výstup se zobrazuje buď ve formě obrázku, nebo textu přímo pod formulářem. Správnost vstupních parametrů zajišťuje samotná knihovna. Případné chybové hlášky jsou zobrazeny na místě pro výstup kódu. Celé uživatelské rozhraní je napsáno v Nette Frameworku [11]. Ukázka uživatelského prostředí je na obrázku 7.1.

7.3 Aplikace pro inventarizaci majetku

Jako příklad komplexnější aplikace s využitím QR kódu jsem navrhl jednoduchý systém pro inventarizaci majetku VUT. Idea je taková, že veškerý inventář s požadovanými údaji bude uložen v databázi. Pomocí webového rozhraní se bude tento inventář spravovat. Každá položka bude mít svůj unikátní QR kód, ve kterém budou zakódovány potřebné informace. Tento kód bude vytištěn na štítku a nalepen na příslušné položce. Pomocí čtečky kódu (například v mobilním telefonu) si pak může kdokoli o této položce zjistit dané informace.

Aplikace je opět naprogramovaná v PHP s využitím Nette Frameworku [11]. Jako databázové uložiště se využívá MySQL. Základem je databázová tabulka `inventory`, jejíž schéma je uvedeno v tabulce 7.2. Ze struktury této tabulky vychází potom i struktura metadat uložených v QR kódu.

Pro generovaný QR kód jsem zvolil verzi 12, která je pevně daná. Všechny štítky s kódem tak budou mít stejnou velikost. Tato verze kódu je kompromisem mezi velikostí a svojí kapacitou. Kód je ještě dostatečně malý, ale vejde se do něj 287 znaků v 8 bitovém módu



Obrázek 7.1: Ukázka uživatelského rozhraní Online generátoru

při úrovni chybové korekce *M*. Tato úroveň byla zvolena za předpokladu, že k poškození štítku s kódem bude docházet jen zřídka.

Na základě kapacity QR kódu jsem volil i maximální možnou velikost jednotlivých databázových polí tak, aby se jejich součet vešel do 287 znaků. Přitom jsem bral v potaz i režiji, v podobě klíčových slov jednotlivých polí metadat. Struktura metadat je rozebraná v podkapitole 7.3.1.

Aplikace je dostupná online na adrese níže uvedené adrese. Uživatelský manuál je uveden v příloze B.

`http://www.barcodepack.com/admin/inventory/`

Tabulka 7.2: Struktura tabulky inventory.

Sloupec	**Typ**	**Popis**
id	int(11) Auto Increment	Identifikátor položky sloužící jako primární klíč
invNo	varchar(10)	Inventární číslo
name	varchar(20)	Název položky
faculty	varchar(6)	Zkratka názvu fakulty
department	varchar(10) NULL	Zkratka oddělení
room	varchar(10) NULL	Označení místnosti
address	varchar(60) NULL	Adresa
person	varchar(25) NULL	Odpovědná osoba za položku
type	varchar(15) NULL	Typ položky
note	varchar(100) NULL	Poznámka

7.3.1 Návrh struktury metadat

Pro potřeby ukládání metadat do QR kódu jsem vycházel z formátu *MECARD* zmíněného v podkapitole 5.1.5. Data jsou uloženy ve formátu `klíč: hodnota;`. Pro rozpoznání našeho "inventárního formátu" jsem použil klíč `INV`. Klíče jednotlivých položek jsou jednoznakové kvůli úspoře místa pro data. Délka řídících dat (klíče a oddělovací znaky) je 33 znaků. Na samotná data tak zbývá 254 znaků.

```
INV:
    I:  AB1234;                       // Inventární číslo
    N:  Židle;                        // Název položky
    F:  FIT;                          // Zkratka fakulty
    D:  UPGM;                         // Zkratka oddělení
    R:  L205;                         // Označení místnosti
    A:  Božetěchova 1/2, 612 66 Brno; // Adresa
    NT: Nepřenášet;                   // Poznámka
    P:  Petr Nejezchleb;              // Odpovědná osoba
    T:  furniture;                    // Typ
;
```

Aby byl náš formát správně přečten, je nutná softwarová úprava čtečky. Například ke čtečce BarcodeScanner [28] jsou volně dostupné zdrojové kódy. Její úprava pro naše účely by tak nebyla složitá.

Kapitola 8

Závěr

Cílem bakalářské práce bylo nastudovat problematiku lineárních a dvoudimenzionálních čárových kódů a na základě získaných teoretických znalostí vytvořit knihovnu pro generování vybraných čárových kódů.

Teoretickou část jsem se snažil koncipovat tak, aby čtenář získal náhled nad různými typy čárových kódů a na základě získaných informací dokázal některé z nich sestrojit. Dle zadání této práce jsem se měl hlouběji zabývat kromě QR kódu i dvoudimenzionálním kódem Data Matrix a skládaným kódem PDF417. Tyto kódy jsou však ještě stále velmi "mladé" a je o nich málo kvalitních informací. V češtině dokonce žádné. Jedinou možností by bylo zakoupit příslušné normy. Při ceně kolem 4 300 Kč za jednu normu jsme se s vedoucím práce dohodli, že bude stačit jen krátší popis z volně dostupných zdrojů.

V praktické části bakalářské práce se mi podařilo uspěšně implementovat knihovnu pro generování čárových kódů. Jejím stěžejním bodem je QR kód, jehož studium a pochopení mi zabralo nejvíce času. Na základě této knihovny jsem vytvořil několik praktických ukázek možného použití. Myslím, že knihovna je dostatečně kvalitní, aby mohla být nasazená i v praxi.

Díky této práci jsem hlouběji pronikl do velmi zajímavé oblasti automatické identifikace a obzvláště pak čárových kódů. Získané znalosti a mnou vytvořenou knihovnu budu určitě aplikovat na dalších projektech.

Literatura

[1] ISO/IEC 18004. Information technology—Automatic identification and data capture techniques—Bar code symbology—QR Code. Geneva : ISO, 2000. vi, 114 s.

[2] ADDAMS, Russ: BarCode 1. [online], 1995 [cit. 2010-12-27], <http://www.adams1.com/>.

[3] ADDAMS, Russ: A Short History Of Bar Code. [online], 1995 [cit. 2010-12-27], <http://www.adams1.com/history.html>.

[4] BENADIKOVÁ, Adriana; MADA, Štefan; WEINLICH, Stanislav: *Čárové kódy — automatická identifikace*. Grada, 1994, ISBN 80-85623-66-8, 272 s.

[5] Datastrip: Datastrip. [online], 2010, [cit. 2011-05-04], <http://www.datastrip.com>.

[6] DL Technology Ltd.: Barcodes. [online], 2010, [cit. 2011-03-17], <http://www.dlsoft.com/support/dlHelps/helps/barcodes/>.

[7] DZIENIA, Dominik: PHP Qr Code. [online], 2010, [cit. 2011-01-10], <http://phpqrcode.sourceforge.net/>.

[8] FUKUCHI, K.: libqrencode. [online], 2006, [cit. 2011-03-16], <http://fukuchi.org/works/qrencode/index.en.html>.

[9] KODYS, spol. s r.o.: Identifikační technologie. [online], 2009, [cit. 2011-04-15], <http://www.kodys.cz/identifikacni-technologie.html>.

[10] MOYA, Mason K.: Short History of Barcodes. [online], 2011, [cit. 2011-01-05], <http://www.moyak.com/papers/history-barcodes.html>.

[11] Nette Foundation: Nette Framework. [online], 2008 [cit. 2011-03-13], <http://nette.org>.

[12] NTT DoCoMo : Synchronization with Native Applications. [online], 1996 [cit. 2011-04-26], <http://www.nttdocomo.co.jp/english/service/imode/make/content/barcode/function/application/>.

[13] PALMER, Roger C.: *The Bar Code Book*. Trafford publishing, páté vydání, 2007, ISBN 978-1-4251-3374-0, 470 s.

[14] RedTitan Technology Ltd: Galois Field Arithmetic. [online], 2011, [cit. 2011-04-28], <http://eduramble.org/rs2/galois.html>.

[15] RedTitan Technology Ltd: REED SOLOMON calculator for QRCODE barcode. [online], 2011, [cit. 2011-04-28], <http://eduramble.org/rs2/calculator.html>.

[16] RKD Software: OnLine Barcode Decoder. [online], 1999 [cit. 2011-05-10], <http://www.datasymbol.com/barcode-recognition-sdk/barcode-reader/online-barcode-decoder.html>.

[17] Rosistem: Barcode Education. [online], [cit. 2010-04-22], <http://www.barcode.ro/tutorials/barcodes/ultracode.html>.

[18] SimpleAct, Inc.: QuickMark. [online], [cit. 2011-05-11], <http://quickmark.com.tw>.

[19] TEC-IT: Barcode Knowledge Base. [online], [cit. 2011-05-14], <http://www.tec-it.com/en/support/knowbase/symbologies/Default.aspx>.

[20] Userstar Information System co., Ltd.: User Star. [online], [cit. 2010-04-22], <http://www.userstar.com.tw/english/Productsynopsis01.html>.

[21] VANHARA, John: Shipito implementuje radiove cipy. [online], 2010-03-31 [cit. 2011-04-17], <http://www.podnikanivusa.com/2010/03/31/shipito-implementuje-radiove-cipy/>.

[22] WWW stránky: Barcode Island. [online], 2006 [cit. 2010-12-27], <http://www.barcodeisland.com>.

[23] WWW stránky: QR code. [online], 2011 [cit. 2010-04-22], <http://en.wikipedia.org/wiki/QR_code>.

[24] ZANDL, Patrick: RFID. Budoucnost. Realita (1.). [online], 2004-03-31 [cit. 2011-04-17], <http://www.lupa.cz/clanky/rfid-budoucnost-realita-1/>.

[25] ZANDL, Patrick: RFID. Budoucnost. Realita (2.). [online], 2004-07-29 [cit. 2011-04-17], <http://www.lupa.cz/clanky/rfid-budoucnost-realita-2/>.

[26] ZXing Team: ZXing Decoder Online. [online], 2008 [cit. 2011-05-14], <http://zxing.org/w/decode.jspx>.

[27] ZXing Team: ZXing ("Zebra Crossing"). [online], [cit. 2011-04-26], <http://code.google.com/p/zxing/>.

[28] ZXing Team: Barcode Scanner. [online], [cit. 2011-05-12], <https://market.android.com/details?id=com.google.zxing.client.android>.

Příloha A

Příklady volání API

Příklad A.1: Text "VUTBR" zakódovaný pomocí QR kódu s velikostí modulu 4px a mírou chybové korekce `H`.

`http://www.barcodepack.com/api/?type=qrCode&text=VUTBT&moduleSize=4&ecl=H`

Obrázek A.1: Příklad volání API.

Příklad A.2: Text "FIT VUT" zakódovaný pomocí QR kódu, defaultní chybovou korekcí a textovým výstupem.

`http://www.barcodepack.com/api/?type=qrCode&text=FIT VUT&output=rawdata`

Výstup:

```
11111110000000111111
10000101110001000001
10111010010110111101
10111010000100111101
10111010111010111101
10000100100101000001
11111110101010111111
00000000000100000000
10101010010100010010
01111000111000100100
10011010100010010011
01101100111000100001
10100111000010101000
00000000101101010111
11111110000101110100
10000100111110110001
10111010110101100001
10111010000000100010
10111010110010010101
10000100100001000011
11111110110010110110
```

Příklad A.3: Text "Ahoj svete!" zakódovaný pomocí kódu Code 128, výstupem do PNG a velikostí modulu 2.

```
http://www.barcodepack.com/api/?type=code128&text=Ahoj svete!&
                                                output=png&moduleSize=2
```

Obrázek A.2: Příklad volání API.

Příklad A.4: Řetězec "859473130328" zakódovaný EAN kódem s textovým výstupem.

```
http://www.barcodepack.com/api/api/?type=upc&text=85947313032&output=rawdata
```

Výstup:

```
10101101110110001000101101000110111011011101010
10110011010000101110010100001011011001100110101
```

Příklad A.5: Řetězec "355723" zakódovaný kódem Interleaved 2/5 s velikostí modulu 3 a výstupem ve formátu GIF.

```
http://www.barcodepack.com/api/api/?type=i2of5&text=355723&output=gif&moduleSize=3
```

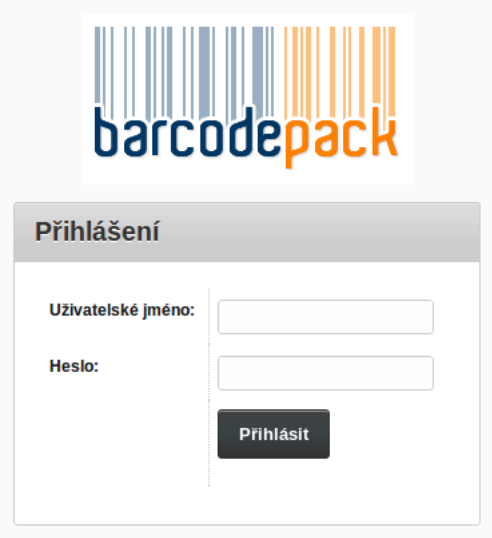
Obrázek A.3: Příklad volání API.

Příloha B

Uživatelský manuál

Uživatelský manuál popisuje ovládání demonstrační aplikace pro inventarizaci majetku. Pro přístup do aplikace je nutné projít přes přihlašovací obrazovku (obr. B.1). Byl vytvořen testovací uživatel s následujícími přihlašovacími údaji:

- Uživatelské jméno: `john`
- Heslo: `Xjohn1`

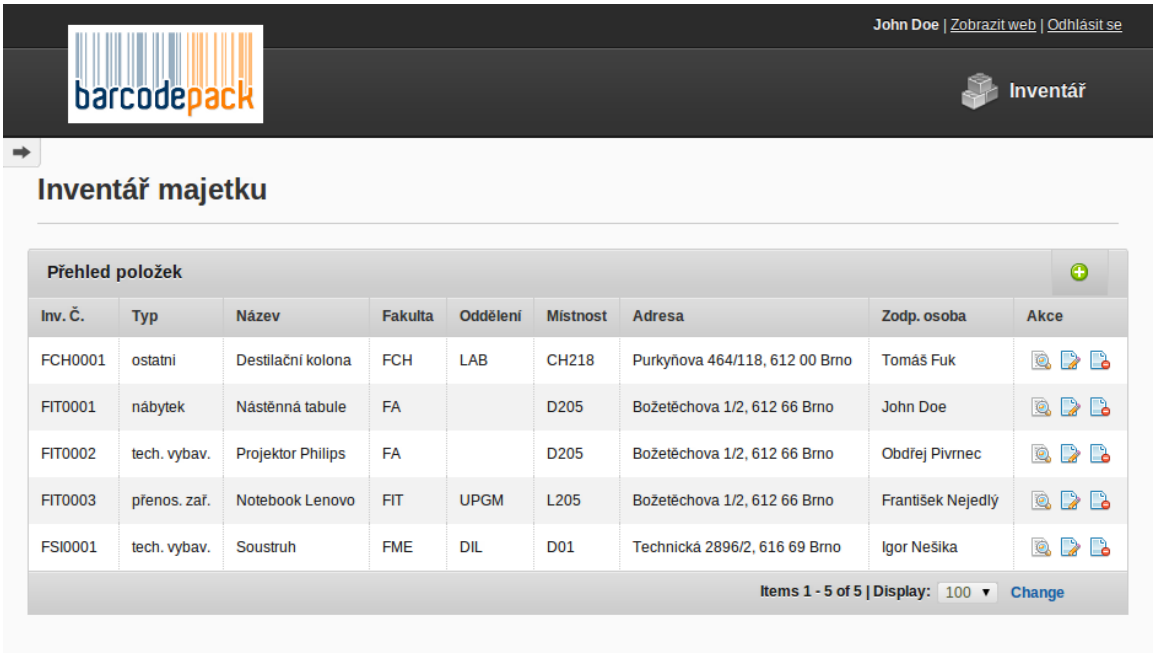


Obrázek B.1: Přihlašovací obrazovka

Ihned po přihlášení se zobrazí tabulka s výpisem inventárních záznamů (obr. B.4). Nový záznam se přidá kliknutím na ikonu "plus" v pravém horním rohu tabulky. Práce s jednotlivými záznamy probíhá prostřednictvím tří ikon, jejichž význam je uveden v tabulce B.1.

Stránka pro přidání nové inventární položky na obrázku B.4 obsahuje formulář, jehož jednotlivé položky jsou popsány v podkapitole 7.3. Pokud jsou zadané údaje vyplněny špatně, zobrazí se javascriptová hláška. Omezení délky vstupních dat je dáno návrhem databázové tabulky uvedené v tabulce 7.2.

Editace položky se provádí kliknutím na editační ikonu. Stránka s editací je totožná se stránkou pro přidání, formlář je zde jen předvyplněný.



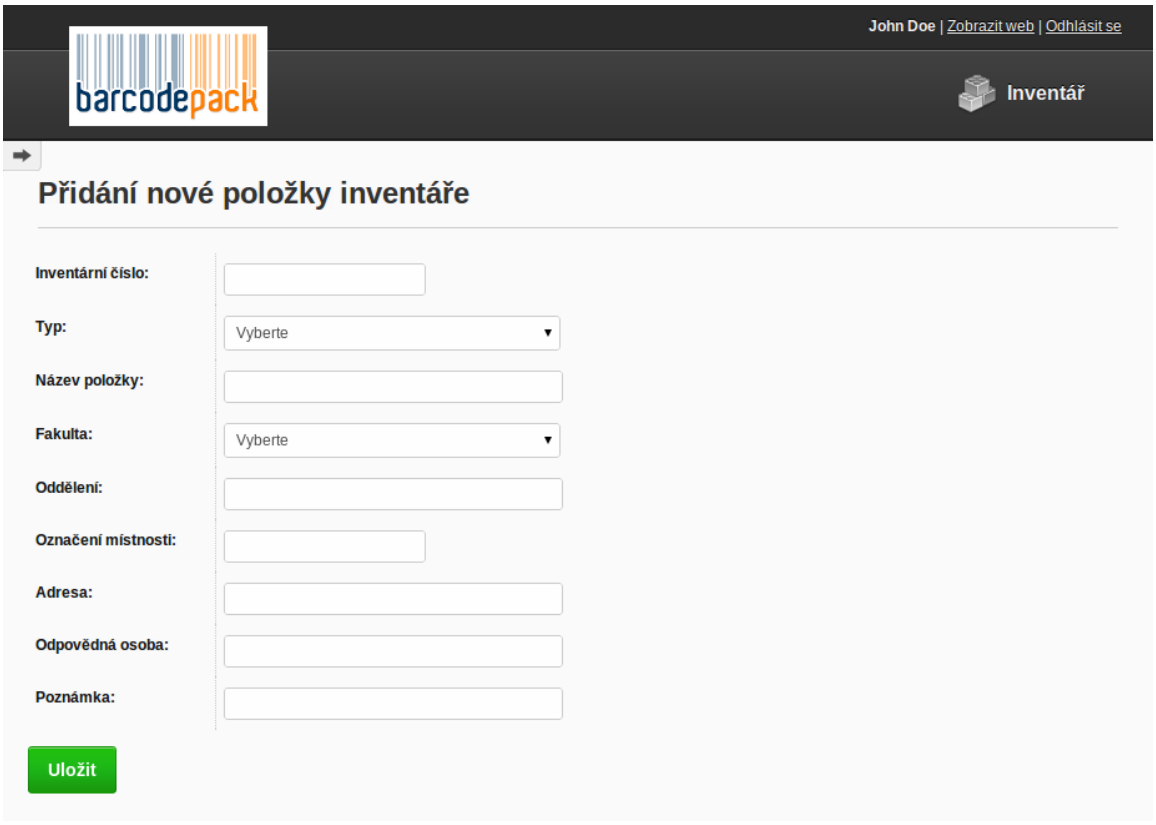
Obrázek B.2: Výpis inventárních položek

Tabulka B.1: Význam ikon.

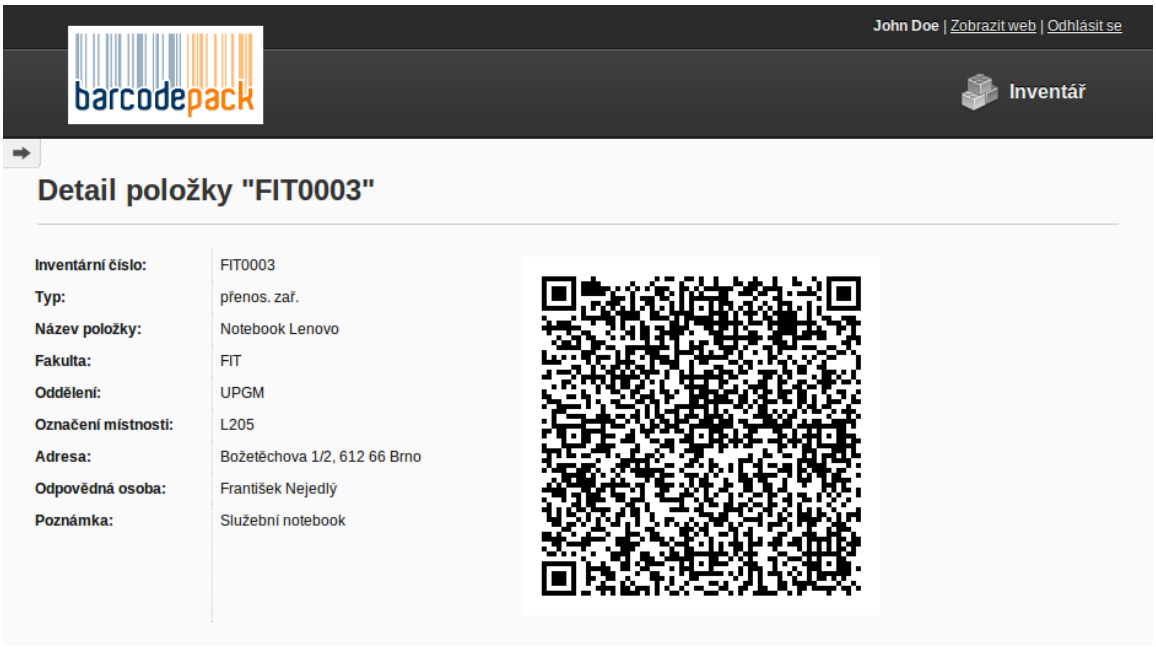
Ikona	Popis
	Zobrazí stránku s formulářem pro přidání nového záznamu.
	Zobrazí detail inventární položky.
	Zobrazí stránku pro editaci inventární položky.
	Smaže záznam z databáze.

Kliknutím na ikonu detail se zobrazí stránka s detailem inventární položky. Na této stránce je také QR kód příslušného záznamu, jehož strukturu jsme si popsali v podkapitole 7.3.1. Každý záznam se dá také smazat kliknutím na poslední ikonu.

Odhlášení ze systému probíhá kliknutím na odkaz “Odhlásit se”, který je na každé stránce vlevo nahoře.



Obrázek B.3: Přidání inventární položky



Obrázek B.4: Detail inventární položky

Příloha C

Obsah CD

Přiložené CD obsahuje následující adresářovou strukturu:

`api`	—	obsahuje soubory implementující API
`barcodepack`	—	soubory s knihovnou
`databaze`	—	v tomto adresáři je umístěn SQL skript pro vytvoření a naplnění databáze
`dokumentace`	—	programová dokumentace ke knihovně
`text`	—	text práce v elektronické podobě a její zdrojové soubory
`web`	—	adresář se soubory online generátoru a aplikace pro inventarizaci majetku