Univerzita Pardubice

Fakulta elektrotechniky a informatiky

# Technologie tisku štítků a související softwarová podpora

Bc. Martin Fryml

Diplomová práce

2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Bc. Martin Fryml** |
| Osobní číslo: | **I17204** |
| Studijní program: | **N2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Technologie tisku štítků a související softwarová podpora** |
| Zadávající katedra: | **Katedra softwarových technologií** |

Zásady pro vypracování:

V teoretické části práce bude představena technologie tisku štítků, typy tiskáren (podporujících tento typ tisku) a existující vybrané jazyky pro popis tiskové stránky. Dále se student bude zabývat podrobněji tiskárnami Zebra a jazykem ZPL. Následně budou v práci popsány možné způsoby komunikace s tiskárnou (USB, LPT, LAN). V práci budou rovněž stručně popsány principy čárových kódů a metody konstrukce vybraných typů kódů V praktické části práce bude proveden návrh a implementace rozhraní pro komunikaci s tiskárnou (tisk a nahrání dat do paměti tiskárny). Dále bude navržen a implementován softwarový nástroj pro editor štítků. Editor umožní uživateli navrhnout typový štítek obsahující text, čárový kód, obrázek a vybrané grafické tvary. Dále bude možno typový štítek před tiskem dynamicky plnit daty z vybraného zdroje (databáze, soubor). Navržené štítky bude možno pomocí dávkového zpracování poslat k tisku. Implementační práce budou provedeny v jazyce C# s využitím knihovny ThermalLabel SDK od firmy Neodynamic.

Rozsah grafických prací:

Rozsah pracovní zprávy: **cca 60 stran**

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

Seznam odborné literatury:

**AXELSON, Jan USB complete: the developer's guide Fifth edition. Madison, WI: Lakeview Research, [2015]. ISBN 978-1931448284 MAKOFSKE, David B, Michael J DONAHOO a Kenneth L CALVERT.** *TCP/IP sockets in C#: practical guide for programmers*. **Boston: Elsevier, c2004. ISBN 978012466051 TIMPLEDON, Miriam T., Susan F. MARSEKEN a Lambert M. SURHONE.** *Zebra Technologies: Zebra Technologies, Barcode, RFID, Smart label, Fortune 500, ISO 9001, Jabil Circuit*. **Betascript Publishing, 2010, 96 s. ISBN 978-613-0-55262-6 Programming Guide: ZPL II ZBI 2 Set-Get-Do Mirror WML. In:Zebra Technologies[online]. Lincolnshire, USA, 2018 [cit. 2018-10-04]. Dostupné z: https://www.zebra.com/content/dam/zebra/manuals/printers/common/programming/zbi2-pm-en.pdf**

Vedoucí diplomové práce: **Ing. Petr Veselý**
Katedra softwarových technologií

Datum zadání diplomové práce: **22. října 2018**

Termín odevzdání diplomové práce: **18. května 2019**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 17. listopadu 2018

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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 14. 5. 2019 Bc. Martin Fryml

## PODĚKOVÁNÍ

Rád bych poděkoval rodině, přítelkyni a přátelům za neustálou podporu během studia.
Rovněž bych chtěl poděkovat vedoucímu práce Ing. Petru Veselému za odborné vedení práce.

## ANOTACE

V teoretické části je představena technologie tisku štítků, typy tiskáren (podporujících tento typ tisku) a existující vybrané jazyky pro popis tiskové stránky. Podrobněji jsou popsány tiskárny Zebra, jazyk ZPL a komunikace s tiskárnou (USB, LPT, LAN). Práce rovněž obsahuje principy čárových kódů a metody konstrukce vybraných typů kódů.

Praktickým výstupem práce je návrh a implementace rozhraní pro komunikaci s tiskárnou a softwarový nástroj pro návrh a úpravu štítků pro platformu Windows. Editor umožňuje navrhnout typový štítek obsahující text, čárový kód, obrázek a vybrané grafické tvary. Štítek lze před tiskem plnit dynamickými daty z databáze nebo souboru.

## KLÍČOVÁ SLOVA

C#, Editor štítků, síťová komunikace, tisk, USB, Windows, Zebra.

## TITLE

Label printing technology and related software support.

## ANNOTATION

The theoretical part of this thesis concentrates on label printing technology, printer types, and existing page description languages. The Zebra printers, ZPL, and printer communication types (USB, LPT, LAN) are described in more detail. The thesis also contains barcode principles.

The practical part covers the design and development of an API for printer’s communication and label editor for Windows platform. Label editor allows the user to design a label, containing text, barcode, image and/or some graphic shapes. The label can be filled with dynamic data using the database or a file.

## KEYWORDS

C#, Label editor, network communication, print, USB, Windows, Zebra.

# OBSAH

# SEZNAM ILUSTRACÍ A TABULEK

# SEZNAM ZKRATEK A ZNAČEK

| | |
|---|---|
| AM | Amplitude Modulation |
| API | Application Programming Interface |
| ASCII | American Standard Code for Information |
| B64 | Base64 |
| CRC | Cyclic Redundancy Check |
| DPI | Dots per inch |
| DRAM | Dynamic Random-Access Memory |
| EA | Enterprise Architect |
| EAN | European Article Number |
| EPL | Eltron Programming Language |
| EPROM | Erasable Programmable Read-Only Memory |
| FM | Frequency Modulation |
| GUID | Globally Unique Identifier |
| IBM | International Business Machines Corporation |
| ID | Identifier |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Internet Protocol |
| JAN | Japan Article Number |
| JPEG | Joint Photographic Experts Group |
| JSON | JavaScript Object Notation |
| LPT | Line Print Terminal |
| LZ | Lempel-Ziv |
| LZMA | Lempel-Ziv-Markov chain Algorithm |

| | |
|---|---|
| MSSQL | Microsoft SQL Server |
| PCL | Printer Command Language |
| PCMCIA | Personal Computer Memory Card International Association |
| PDL | Page Description Language |
| PID | Process Identifier |
| PNG | Portable Network Graphics |
| QR | Quick Response |
| RAM | Random-Access Memory |
| RFID | Radio-Frequency Identification |
| RIP | Raster Image Processor |
| SDK | Software Development Kit |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| UCC | Unique Country Code |
| UPC | Universal Product Code |
| URL | Uniform Resource Locator |
| USB | Universal Serial Bus |
| UTF | Unicode Transformation Format |
| UWP | Universal Windows Platform |
| VID | Vendor Identifier |
| XAML | Extensible Application Markup Language |
| XML | Extensible Markup Language |
| ZPL | Zebra Programming Language |

# TYPOGRAFICKÉ KONVENCE

V rámci této práce jsou pro zvýšení srozumitelnosti dodržovány zde uvedené konvence. Jedná se převážně o použití:

*Kurzívou* se vyznačují názvy konkrétních metod, názvy vlastních tříd, názvy souborů a cesty k umístění souboru.

`Bezpatkové písmo Verdana` o velikosti 11 se světle šedým stínováním je použito pro zobrazení zdrojových kódů.

# ÚVOD

Štítky jsou vidět denně v různých podobách, tvarech a místech. Jejich produkce a tisk začíná většinou již během samotné výroby daného produktu. Velké průmyslové firmy používají specializované tiskárny a software, který tiskne předem nadefinované štítky. Ty jsou doplněny aktuálními informaci a umisťovány na vyráběný produkt. To však není nutnou podmínkou. Jejich výroba se v dnešní době stala dostupnou a malou příruční tiskárnu na tisk takových štítků si může pořídit každý. Avšak právě dostupnost způsobila nárůst komunikačních jazyků s tiskárnou a podporu více typů komunikačních rozhraní.

Velké výrobní firmy v dnešní době čelí problému s nekompatibilními rozhraními a nutnosti aktualizovat svůj software o další nové komunikační rozhraní. Cílem práce je vytvořit komunikační API, které odstraní problémy s různorodostí komunikačních rozhraní a bude poskytovat jednotné a ucelené rozhraní pro komunikaci s cílovými tiskárnami. API by mělo zvládnout otevřít komunikační kanál s téměř jakoukoliv tiskárnou štítků na vybraných typech komunikačních rozhraní. Zároveň by mělo poskytovat relevantní chybová hlášení tiskárny (pokud to tiskárna štítků umožňuje) a manipulovat s pamětí tiskárny.

Dalším často řešeným problémem je samotný návrh štítku. Na trhu existuje pouze pár komerčních produktů, které poskytují editor pro návrh a správu štítků. Vydavatelé těchto produktů si uvědomují svoje výhodné postavení a podle toho licencují své produkty. Editory na návrh štítků často vyjdou výrobní firmy na velké množství peněz, a navíc jsou obvykle ročně licencované. Tento přístup si mnoho firem nemůže nebo nechce dovolit, a proto hledají alternativní řešení. Druhá část práce se proto bude zabývat Editorem štítků, který by měl poskytnout návrhové prostředí pro tvorbu štítků. Společně s Komunikačním API bude editor schopný návrhu štítku nebo štítkové šablony a komunikace s tiskárnou. Síla návrhového nástroje není v návrhu jediného štítku, ale v návrhu jeho šablony. Šabloně se následně přiřadí zdroj dat a jejich spojením je možné hromadně tisknout štítky s různými hodnotami bez nutnosti modifikace štítku. Tato funkcionalita by měla být klíčovou pro Editor štítků, který je součástí práce. Dále by Editor štítků a Komunikační API měly být plně kompatibilní s operačním systémem Windows a měly by mít ošetřené veškeré logické, designové i programovací chyby.

Diplomová práce bude rozdělena do dvou částí, teoretické (kapitoly 1–4) a praktické (kapitoly 5 a 6). Cílem teoretické části práce je uvést čtenáře do problematiky tisku štítků, představit mu různé způsoby a technologie tisku a poskytnout mu znalosti o stavbě a konstrukci čárových kódů.

V úvodu práce bude čtenáři představen štítek a terminologie tisku, která bude použita v celé práci. Následně se práce bude zabývat komunikací s tiskárnou, jazykům pro popis stránky a podobě dat během přenosu. Data se většinou do tiskáren neodesílají v čistém textu, ale využívají různých metodik a transformací, nejen pro snazší a rychlejší přenos, ale také pro rychlejší zpracování samotnou tiskárnou.

Součástí teoretické části bude také kapitola věnující se čárovým kódům, které jsou často spojovány s technologií štítků právě kvůli jejich výskytu na štítcích. Jelikož existuje velké množství čárových kódů, práce se bude věnovat pouze obecnému přehledu a popisu konstrukce vybraných typů čárových kódů.

Praktická část práce bude obsahovat návrh a implementaci Komunikačního API a Editoru štítků. Oba programy budou popsány v oddělených kapitolách. Rovněž bude popsána problematika připojení pomocí vybraných typů komunikačních rozhraní a manipulace s pamětí tiskáren. Práce se bude podrobněji zabývat tiskárnami Zebra a jazyku ZPL, pokud nebude uvedeno jinak. Praktická část bude odzkoušena na tiskárně Zebra 110Xi4.

# 1 TISKÁRNY ŠTÍTKŮ

Tiskárny štítků jsou speciální tiskárny určené ke specifickému tisku formátovaných štítků, cedulek a čárových kódů. Typicky zpracovávají žádosti o tisk pomocí vlastního zabudovaného operačního systému či firmwaru.

Je mnoho výrobců štítkových tiskáren, typů tiskáren a štítků na které mohou tyto tiskárny tisknout. Každý výrobce a tiskárna používá specifický jazyk a technologii tisku. Tato kapitola zavádí definice a pojmy nezbytné k pochopení procesu tisku štítků. Zároveň obsahuje představení typů tiskáren pro tisk štítků a vybraných výrobců štítkových tiskáren.

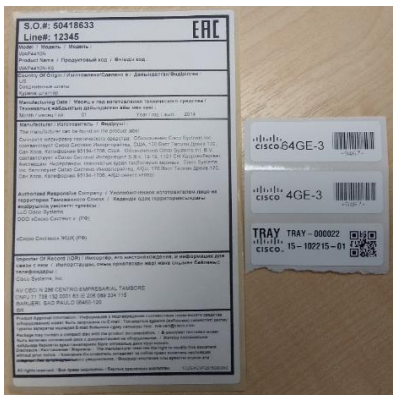
Kapitola vychází ze zdrojů [1–8].

## 1.1 Definice štítku

Štítky (také etiketa či anglicky label) mají mnoho podob a velikostí. Nejčastěji se vyskytují jako kusy papíru, papírové fólie nebo látkové etikety. Každý druh štítku vyžaduje jiný typ tiskového zařízení.

Štítek nemusí být nutně obdélníkového tvaru s jedním čárovým kódem. Štítky lze využít i mimo průmyslový obor. Takové štítky mají tvar visaček, nálepek či cenovek. Lze je nalézt na spotřebičích, potravinách, oblečení a obalech. Obsahují velkou variaci informací, vzhledem k produktu, na kterém jsou umístěné.

Štítky dále obecně rozdělujeme na dva typy. Na nelepící, které používají jako základní materiál papír, látku nebo kov, a na samolepící štítky. Samolepící štítky jsou také vyrobeny z papíru či fólie, ale zadní stranu mají opatřenou vhodným druhem lepidla. Obzvlášť na potravinových štítcích je třeba dbát na nezávadnost použitého lepidla. Takové štítky jsou běžně umisťovány např. na ovoce. V rámci práce bude slovem štítek myšlen papírový nebo fóliový štítek, určený k umístění na obalový materiál nebo elektronické zařízení (vizte obr. 1).




**Obrázek 1 – Samolepící papírové štítky použité pro diplomovou práci (zdroj: [Autor])**

## 1.2 Terminologie

Štítky se mohou dodávat navinuté na roli, v podobě archů nebo i samostatně po jednom. Nejčastěji se vyskytují na roli, aby se usnadnil jejich tisk a zařízení (např. tiskárna) mohlo tisknout více štítků najednou. V rámci tisku štítků je třeba zavést jisté technické pojmy a vysvětlit jejich význam.

### Width a Height

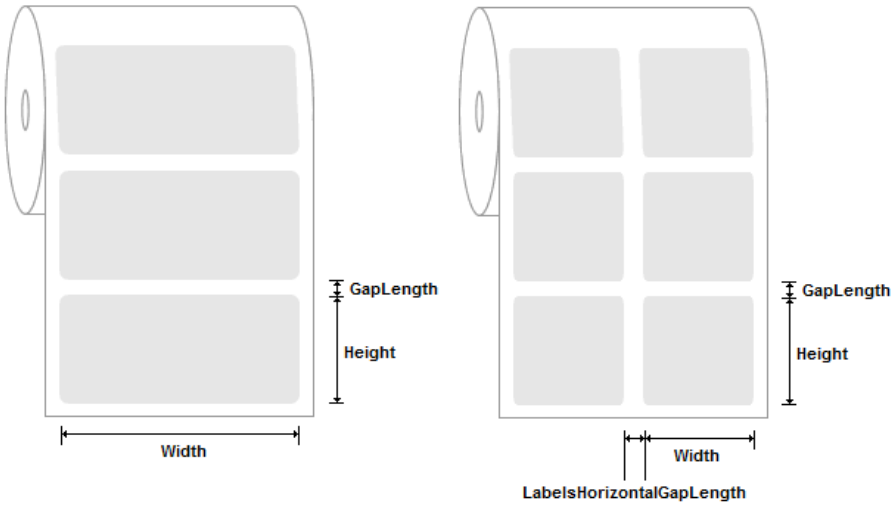
Mezi základní pojmy patří šířka a výška štítku. Tyto dva pojmy určují velikost samotného štítku. Je třeba dát pozor, že hodnoty udávají velikost samotného štítku, ne velikost podkladového papíru. V případě, že je podkladový papír širší než šířka štítku, je třeba nastavit tiskárnu a software tak, aby s touto mezerou a posunutím počítaly. Toho je možné docílit bočním posunutím (offset).

### Gap length

Druhým důležitým pojmem je mezera mezi jednotlivými štítky, která udává, jak daleko jsou od sebe vzdáleny dva štítky na podkladovém papíře.

### Horizontal gap length

Štítky mohou být na roli umístěny i po dvojicích. V takovém případě je třeba změnit šířku štítku, aby odpovídal šířce jednoho štítku, a ne celé roli. Opět je třeba mít na paměti, že velikost udává šířku štítku, a ne podkladového papíru. Také je třeba zavést nový termín, a to šířku horizontální mezery, která definuje horizontální šířku mezi dvěma štítky na roli. Ilustrace rozměrů je zobrazena na obr. 2.




**Obrázek 2 – Vizuální zobrazení měrných termínů na dvou typech rolí (zdroj: [2])**

## 1.3 Typy tiskáren

Dle kvantity, použití štítků, technologie tisku, vstupního rozhraní a požadavků se volí i vhodný typ tiskárny. Výběr nevhodného typu a materiálu může přinést problémy při používání tiskárny a následném tisku. Tiskárny štítků se dělí do třech základních kategorií: příruční tiskárny, stolní tiskárny a průmyslové tiskárny.

V následující kapitole budou blíže představeny nejběžnější typy tiskáren. Mezi další typy patří například tiskárny karet, dárkových poukazů, osobních identifikátorů, RFID tiskárny, tiskárny štítků na zakázku a tiskové komponenty. Tiskové komponenty jsou části tiskáren, které sami o sobě nedokáží fungovat a jsou využity jako náhradní díly nebo součásti pro výrobu tiskáren na míru. Příkladem takové komponenty je tiskový motor (print engine).

### 1.3.1 Příruční tiskárny

Příruční tiskárny (handheld, mobile nebo compact printers), jak název napovídá, jsou malé, bezdrátové, přenosné tiskárny, které lze držet v rukách a snadno s nimi manipulovat. Některé tiskárny obsahují pogumované části, aby snesly větší fyzické zacházení. Často se využívají v malých skladech pro potisk obalového materiálu nebo pro dodatečný tisk malého množství štítků. Příruční tiskárny obsahují malý zásobník na roli štítků a rychlost tisku nepřesahuje 120 mm za sekundu. Nejsou tedy vhodné pro hromadný tisk nebo tisk štítků velkých rozměrů. Vytisknuté štítky mají průměrnou kvalitu, která je dána nízkým DPI. U příručních tiskáren se DPI pohybuje mezi 180 až 300 DPI.

Jejich hlavní výhodou je kompaktnost a přenositelnost. Zabírají minimum místa a jsou snadno skladovatelné. Hmotnost příručních tiskáren nepřesahuje hodnotu jednoho kilogramu, a jelikož se jedná o bezdrátová zařízení, komunikují nejčastěji pomocí Bluetooth, WiFi nebo výjimečně pomocí USB. Cenově se pohybují v řádech několika stovek až pár tisíců korun. Vzhled typické příruční tiskárny je na obr. 3.

**Obrázek 3 – Příruční tiskárna Zebra (zdroj: [3])**

### 1.3.2 Stolní tiskárny

Stolní tiskárny (desktop printers) jsou větší verze příručních tiskáren. Jsou celé vyrobeny z plastových polymerů a snesou menší fyzickou zátěž než příruční tiskárny. Tyto tiskárny jsou určeny ke statickému umístění a nepředpokládá se jejich častý přesun. Proto mají, na rozdíl od příručních tiskáren, více možností připojení. Mezi často využívaná rozhraní patří USB, LAN, paralelní port a sériový port RS-232. Některé typy stolních tiskáren mohou podporovat i bezdrátový přenos pomocí Bluetooth. Ukázka stolní tiskárny je na obr. 4.

Výhodou stolních tiskáren oproti příručním tiskárnám je jejich provedení a dostupnost. Cena je však vyšší než u příručních tiskáren a pohybuje se v řádech tisíců korun. Obsahují větší prostor pro role, než mají příruční tiskárny, umožňují tisk širších štítků a lze na nich tisknout větší množství štítků. Rychlost tisku se pohybuje mezi 127 až 153 mm za sekundu. Hmotnost stolních tiskáren nepřesahuje hodnotu pěti kilogramů. Kvalita tisku dosahuje maximálně 300 DPI. Stolní tiskárny jsou vhodné pro malé firmy nebo jedince, kteří chtějí tisknout štítky pro malovýrobu. Nejsou vhodné pro nepřetržitý tisk.

**Obrázek 4 – Stolní tiskárna Zebra (zdroj: [3])**

### 1.3.3 Průmyslové tiskárny

Poslední kategorií jsou průmyslové tiskárny (industrial printers). Jsou navrženy tak, aby vydržely náročné podmínky, tisk velkého množství štítků a poskytly roky spolehlivé služby. Jsou proto vyrobeny převážně z kovu, na rozdíl od svých plastových předchůdců. Lze je snadno integrovat do existujícího podniku nebo produkce.

Uplatnění mají v široké škále průmyslových odvětví, dopravě, logistice a velkoobchodech. Dokáží dlouhodobě tisknout velké množství štítků s rychlostí až 400 mm za sekundu. Navzdory velké tiskové rychlosti poskytují průmyslové tiskárny i velkou kvalitu tisku. Ta může dosáhnout až 600 DPI. Obsahují velký skladovací prostor pro role štítků a místo pro dodatečné zásuvné moduly (např. RFID zapisovač). Některé typy průmyslových tiskáren mají také podporu gigabitového ethernetu a IPv6 protokolu.

Jejich nevýhodou jsou velké rozměry, hmotnost a cena. Hmotnost přesahující hodnotu deseti kilogramů naznačuje, že nejsou určeny k častému přemisťování. Cena je taktéž mnohonásobně vyšší než u předchozích typů a pohybuje se v řádech desítek až stovek tisíc korun. Obr 5. znázorňuje jednu z mnoha variant průmyslové tiskárny.

**Obrázek 5 – Průmyslová tiskárna Zebra (zdroj: [3])**

## 1.4 Výrobci štítkových tiskáren

Na trhu se vyskytuje mnoho výrobců štítkových tiskáren. Při výběru tiskárny je třeba dbát na mnoho faktorů. Mezi základní faktory výběru tiskárny patří kvalita, rychlost a způsob tisku, dále komunikační rozhraní, maximální šířka štítku, cena tiskárny a komunikační jazyk, který tiskárna používá (vizte kapitola 2).

Většina výrobců používá svůj vlastní proprietární jazyk, kterým komunikuje s tiskárnou. Jazyky jsou ve většině případů mezi sebou nekompatibilní. Je třeba vždy uvážit všechny možnosti a zda pro daný jazyk má uživatel potřebný software.

Mezi největší výrobce štítkových tiskáren patří Zebra Technologies[1], Epson[2], DYMO[3] a Intermec by Honeywell[4]. Většina výrobců nabízí všechny základní typy tiskáren, ale jsou i specializovaní výrobci, kteří se zaměřují pouze na jeden typ.

### 1.4.1 Zebra Technologies

Zebra Technologies (zkráceně Zebra) je celosvětově známá Americká firma zaměřená na výrobu a prodej tiskové a značkovací technologie. Jejich produkty se blíže specializují na tisk čárových kódů, štítků, účtenek, chytré RFID značkování a výrobu identifikačních karet. Zebra Technologies vznikla sloučením několika firem. Jednou z nich byla i firma Eltron International, Inc. Jednalo se o firmu zaměřenou také na tisk štítků. Současné logo Zebra Technologies je zobrazeno na obr. 6.

[1] https://www.zebra.com
[2] https://www.epson.cz
[3] https://www.dymo.com
[4] https://www.honeywellaidc.com

Tiskárny typu Zebra jsou určeny převážně pro průmyslový tisk a na místech, kde je vyžadován tisk velkého množství štítků. Vytisknuté štítky dosahují vysoké kvality. K tisku využívá převážně vlastní proprietární jazyk pro popis stránek ZPL a ZPL II. Starší typy mohou ještě podporovat starší jazyk EPL, který byl převzat z původní firmy Eltron. Více o těchto jazycích v kapitole 2.

**Obrázek 6 – Logo Zebra Technologies (zdroj: [5])**

### 1.4.2 DYMO Corporation

DYMO Corporation (zkráceně DYMO) je další světová značka pocházející z USA s více než šedesátiletou tradicí. Firma je zaměřena na výrobu příručních tiskáren na tisk štítků a plastových (reliéfních) pásků. Vzhled reliéfní pásky je zobrazen na obr. 7. Firma DYMO dále vyvíjí a prodává software určený pro návrh a tisk štítků.


**Obrázek 7 – Reliéfní páska (zdroj: [8])**

DYMO používá odlišný způsob pro popis stránek než např. tiskárny Zebra. Tiskárny DYMO používají speciální podskupinu příkazového jazyka LabelWriter (printer command language). Tyto příkazy se liší dle série tiskáren a lze je nalézt v manuálových stránkách. DYMO poskytuje pouze malou dokumentaci pro komunikaci s tiskárnou a většina TCP komunikace je skrytá.

Firma DYMO se specializuje převážně na výrobu příručních tiskáren. Jejich tiskárny jsou vhodné tam, kde je třeba rychlého tisku štítku a snadné přenositelnosti. Příruční tiskárny DYMO mohou být využity ve skladech a logistice. Jsou nevhodné pro tisk velkého množství štítků a proprietární štítkové role mohou být drahé. Logo firmy je zobrazeno na obr. 8.

**Obrázek 8 – Logo DYMO Corporation (zdroj: [6])**

# 2 KOMUNIKACE S TISKÁRNOU

Tiskárny komunikují se zdrojem dat pomocí různých rozhraní. Mezi často využívaná rozhraní patří USB, síťový ethernetový port RJ-45, paralelní LPT port, sériový port RS-232, WiFi nebo Bluetooth.

Data odesílaná pomocí rozhraní musí mít speciální podobu. Musí být napsána v takovém jazyku, kterému tiskárna rozumí. Tyto jazyky se nazývají jazyky popisu stránky (taktéž jazyky pro popis stránek). Pokud tiskárna dostane sérii příkazů nebo textu, kterému nerozumí, veškerá data zahodí a nic nevytiskne. Tiskárny zpracovávají informace až těsně před tiskem. Například, pro tisk čtverce se pošle speciální příkaz pro tisk čtverce, nikoliv bitmapa nebo vektorový obrázek čtverce.

Při odesílání dat a komunikaci s tiskárnou je důležitá odezva a rychlý přenos. K tomu slouží kódování a komprese dat. K odeslání dat ve správném formátu nebo dekomprimování dat je potřeba znát, jak funguje komprimace daného jazyka a jak s daty pracovat. Proto se část této kapitoly věnuje vysvětlení kódování a komprese ZPL dat.

Kapitola vychází ze zdrojů [9–25].

## 2.1 Jazyk pro popis stránky

Jazyk pro popis stránky (Page Description Language, zkráceně PDL) určuje uspořádání stránky pro tisk pomocí příkazů, které tiskárna provádí. Moderní PDL definují objekty a elementy na stránce pomocí geometrických obrazců, jako jsou čáry, oblouky a další.

Jazyky pro popis stránky mohou být přirovnány k jazykům pro popis obsahu. Typickým příkladem takového jazyka je značkovací jazyk XML. Tyto jazyky určují značkami obsah dokumentu. PDL určují značkami uspořádání a vzhled informací.

PDL definují elementy stránky nezávisle na technologii tisku. Vzhled stránky tedy bude stejný i na různých tiskárnách podporující daný jazyk. Existuje celá řada jazyků pro popis stránek a každý výrobce tiskárny používá jiný jazyk. Zpravidla nejsou mezi sebou kompatibilní a mají odlišnou syntaxi.

V digitálním tisku existuje více než 40[5] jazyků pro popis stránky. Mnoho z nich již zaniklo nebo jej nahradil jiný a lepší jazyk. Mezi nejznámější (ne však nejpoužívanější) PDL patří jazyky uvedené v tabulce 1.

[5] https://en.wikipedia.org/wiki/Page_description_language

**Tabulka 1 – Nejznámější jazyky pro popis stránky (zdroj: [Autor])**

| Zkratka | Celý název | Stručný popis | Vlastník |
|---|---|---|---|
| ASCII | ASCII | Nejjednodušší cesta odeslání dat tiskárně. Velmi omezené formátování. Žádná volba fontů. | |
| ZPL, ZPL II | Zebra Programming Language (II) | Více v podkapitole 2.1.1. | Zebra Technologies |
| PCL | Printer Command Language | Velmi rozšířený jazyk. Rozšiřuje a vylepšuje nedostatky ASCII. Obsahuje navíc formátování a výběr fontu. Poskytuje rychlý tisk. | Hewlett Packard |
| Fingerprint | Fingerprint | Podpora čárových kódů. Text ve formě příkazů. Každý příkaz musí být na novém řádku. Náročné na vytvoření a čtení. | Honeywell |
| ESC/P, ESC/P2 | Epson Standard Code for Printers (2) | Původně vytvořeno pro jehličkové a inkoustové tiskárny. Zastaralý PDL, který je postupně nahrazován PCL. Bylo vyvinuto pět dalších variant na tento jazyk. | Epson |

### 2.1.1 ZPL

ZPL je zkratkou pro Zebra Programming Language. Jedná se o jazyk pro popis stránky vyvinutý společností Zebra Technologies. Tento jazyk je zaměřený výhradně na návrh a tisk štítků a na jeho využití pro aplikace pracující se štítky. Byly vyvinuty dvě verze tohoto jazyka. Nejnovější verze, ZPL II, odstraňuje nedostatky první verze. Zebra Technologies poskytují k ZPL II rozsáhlou, více než tisícistránkovou dokumentaci. Samotné ZPL II obsahuje více než 170 příkazů pro práci a návrh stránky. Pomocí příkazů lze také komunikovat a měnit nastavení tiskárny Zebra.

ZPL II oproti svému předchůdci značně snížila dobu mezi zahájením přenosu dat a zahájení tisku prvního štítku. Toho bylo docíleno převážně změnou a optimalizací skriptů ZPL. ZPL II není zpětně kompatibilní s předchozí verzi, právě kvůli zmíněné změně některých příkazů. Výsledná změna však způsobila jen minimální dopad na zápis standardního ZPL skriptu.

### 2.1.2 ZPL II syntaxe

Příkazy tohoto jazyka začínají vždy stříškou (^) nebo tildou (~). Pokud příkaz nezačíná jedním z těchto znaků, tiskárna předpokládá, že se jedná o chybu nebo část předchozího příkazu. Takový příkaz může tiskárna ignorovat. Veškerá komunikace s tiskárnou musí začínat příkazem `^XA` a končit příkazem `^XZ`. Tyto příkazy označují začátek a konec jednoho štítku. V případě, že tiskárna obdrží sérii těchto příkazů, zachází s nimi jako s dalšími štítky a vytiskne tedy úměrné množství štítků.

Veškerá datová pole jsou formátována ihned, jak jsou přijímána tiskárnou. Ve standardním ZPL nebyla tato pole zpracována, dokud tiskárna neobdržela ukončovací příkaz (^XZ). To umožňuje rychlejší zpracování příkazů a rychlejší tisk štítku. Příklad syntaxe ZPL II je zobrazena na obr. 9.


**Obrázek 9 – Ukázka jazyka ZPL (zdroj: [Autor])**

Velká část ZPL se soustředí právě na formátování textu a příkazů určující, jak bude výsledný text vypadat. Pro tisk obyčejného textu je potřeba příkaz ^FD. Ten definuje datový řetězec ve formě datového pole. Může obsahovat jakýkoliv řetězec, kromě prefixu příkazu (stříška či tilda), až do velikosti 3072 bajtů. V krajních případech, když uživatel potřebuje vytisknout tyto dva speciální znaky, může použít příkaz pro hexadecimální indikátor (^FH) a znak napsat v hexadecimální podobě. Výchozím znakem pro hexadecimální kód je podtržítko. Příkaz umožňuje také volbu vlastního indikátoru. ^FD lze také použít pro zadání hesla pro zápis na RFID štítky (pokud tiskárna umožňuje RFID zápis). Pro ukončení řádku nebo jedné definice pole je třeba použít příkaz ^FS (zkratkou pro Field Separator), který zajistí, že další text se bude nacházet na dalším řádku. Alternativou tohoto příkazu je ukončit text ASCII kódem SI (hexadecimálně 0F).

Mezi nejdůležitější příkaz pro formátování nejen textu, ale veškerých grafických objektů, je příkaz ^FO (zkratkou pro Field Origin). Tento příkaz nastavuje relativní vzdálenost následujícího grafického objektu vzhledem k nulové (počáteční) pozici štítku. Příkaz ^FO nastavuje vždy levý horní roh objektu na dané body osy x a y. Celý zápis tohoto příkazu tedy zní ^FOx,y, kde písmena *x* a *y* značí body na štítku v horizontální a vertikální ose.

ZPL má nulovou verifikaci pozicování štítku a syntaxe. Je tedy možné nastavit hodnoty pozice větší, nežli je velikost štítku. Pokud se tak stane, daný objekt bude vytisknut mimo štítek, či nebude vytištěn vůbec. Uživatel si sám odpovídá za platnost a kontrolu ZPL kódu.

Pro používání mezinárodních jazyků a jazyků s rozdílnou znakovou sadou je důležitý příkaz `^CI`. Tento příkaz je zkratkou pro Change International Font/Encoding. Umožňuje výběr z různých jazykových sad a kódování. V základním nastavení není český jazyk podporován a tiskárna může některé znaky zaměnit za jiné. Pro povolení kódování UTF-8 je třeba provést příkaz `^CI28`. Jazykové sady a kódování lze v jednom štítku měnit a kombinovat. Je doporučeno dávat tento příkaz na začátek každého ZPL skriptu, aby se předešlo problémům s kódováním. Mezi další podporované sady ZPL patří například němčina, francouzština, italština nebo japonština.

V neposlední řadě, pro řádné formátování textu je potřeba příkaz na změnu fontu. Tiskárna může použít jakýkoliv font, který má uložený ve své paměti, pro změnu textu na štítku. Fonty lze na štítku kombinovat a mít tak na štítku text s rozdílnými fonty. Příkaz označován jako `^A@` má komplexní syntaxi, ve které je zahrnuté jméno fontu, velikost písmen, umístění fontu a orientace. Celý zápis je tedy `^A@o,h,w,d:f.x` kde *o* je orientace (normální, otočená o 90 stupňů nebo invertovaná), *h* a *w* pro výšku, respektive šířku písma (vyjádřena v bodech) a následuje umístění fontu. Umístění se skládá ze tří částí. První je písmeno označující paměť, kde se font nachází. Po dvojtečce následuje samotný název fontu a ten je zakončený příponou. Fonty se zpravidla vyskytují s příponou .FNT nebo .TTF.

Následuje příklad zápisu změny fontu textu na font Arial o velikosti 50, umístěný v paměťové části E s normální orientací.

```
^A2N,50,50,E:Arial.FNT
```

Jakýkoliv text, který následuje po příkazu `^A@` bude mít zvolený font až do jeho změny dalším příkazem `^A@` nebo ukončení štítku. Následující tabulka popisuje další časté příkazy pro sestavení jednoduché stránky.

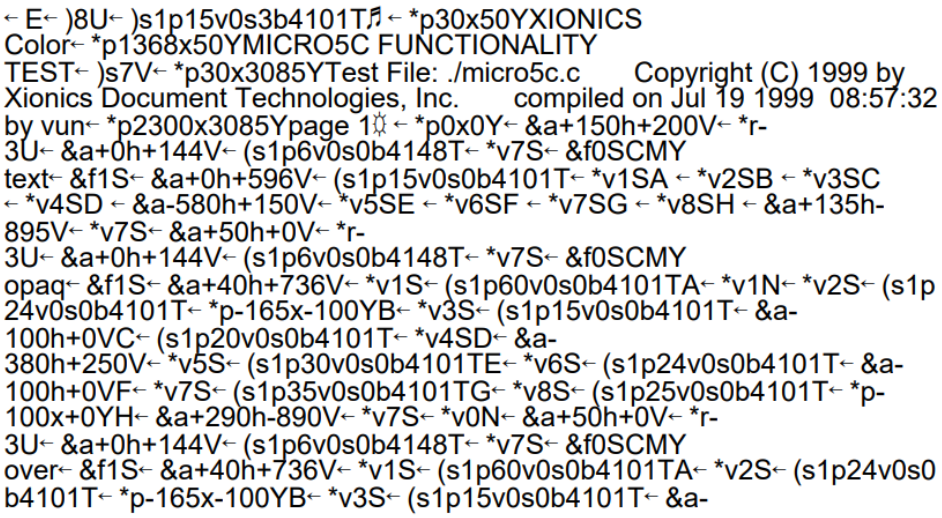
**Tabulka 2 – Časté příkazy ZPL (zdroj: [Autor])**

| **Příkaz** | **Popis** |
|---|---|
| **^FX** | Prostý komentář v kódu. Text za tímto příkazem se nevytiskne. |
| **^FHa** | Označení hexadecimálního kódu, parametr *a* udává kontrolní prefix. |
| **^GF,a,b,c,d,data** | Grafické pole (obrázek). Parametry udávají kompresi, počet bajtů a samotná data. |
| **^BQa,b,c,d,e** | Vytvoří čárový kód typu QR Code. Parametry udávají orientaci, model, zvětšení a korekci dat. Data musí být vložena za tento příkaz do příkazu ^FD. |
| **^BCo,h,f,g,e,m** | Vytvoří typický čárový kód typu Code 128. Parametry specifikují, orientaci, výšku, interpretaci a kontrolní součet. Data musí být vložena za tento příkaz. |

### 2.1.3 PCL

PCL je zkratkou pro Printer Command Language, často chybně zaměňováno za Printer Control Language, což je jiný název pro jazyk popisu stránky. Jazyk PCL byl vyvinutý firmou Hewlett-Packard jako tiskový protokol a s postupem času se stal průmyslovým standardem. Původně byl vyvinutý pro jehličkové tiskárny v pozdních sedmdesátých letech a představen na počátku 80. let. Od té doby prošel řadou revizí a změn. Ty jej přesunuly od jednoduchých tiskáren až k řízení úloh, objektů tisku a podpory vestavěných maker.

PLC má flexibilní strukturu příkazů založenou na řízení a ASCII znacích. Je vysoce výkonný a poskytuje vysokou kvalitu tisku, fontu i grafiky. PCL podporuje černobílý i barevný tisk, tisk bitmapových znaků, rastrovou grafiku a podporu pro tisk do různých směrů. Příklad PCL je zobrazen na obr. 10.




**Obrázek 10 – Ukázka kódu PCL (zdroj: [16])**

V současné době je již 6 verzí PCL označovány jako PCL 1 až PCL 6, které byly vyvinuty v reakci na měnící se technologii tisku a tiskáren. Verze PLC 1 až PCL 5e/c jsou jazyky založené na příkazech jejichž příkazové sekvence jsou zpracovány a interpretovány tak, jak přijdou na tiskárnu. Obvykle je datový tok generován ovladačem tiskárny.

V kancelářském prostředí se nejčastěji používá PCL 5e, vydaný v roce 1992. Tato verze podporuje obousměrnou komunikaci s tiskárnou a PC a různé fonty operačního systému Windows. Novější verze PCL 6 je již značně odlišná, zejména způsobem, jakým jsou data odesílána na tiskárnu.

Jazyk PCL se nejčastěji používá u kancelářských tiskáren, kde není potřeba tisk velkého množství stránek nebo štítků. Jedná se typicky o stolní tiskárny s maximálním zatížením 200 stránek za den. Vzhledem k menšímu vytížení mohou tyto tiskárny tisknout kvalitnější a barevné stránky.

Jedinou velkou nevýhodou PCL je oznamování chyb – není totiž žádné. V případě, že se tiskárna zasekne, dojde barva či toner nebo chybí papír během tisku, PCL není schopno tuto skutečnost oznámit. Mimoto, tisk nelze restartovat z pozice, kde se chyba vyskytla. Musí se začít s tiskem od začátku stránky.

### 2.1.4 EPL

Eltron Programming Language (EPL) je předchůdce jazyka ZPL pro popis stránky. Jazyk se používal pro tisk papírových štítků v tiskárnách modelu Eltron od firmy Eltron International (později spojena s firmou Zebra Technologies). Mnoho novějších tiskáren stále podporuje tento jazyk a existuje u nich zpětná kompatibilita.

Podobně jako ZPL, i EPL prošel vývojem a mnohými vylepšeními. Je možné se setkat s původním EPL i novější verzí EPL 2. V porovnání se ZPL (původní verzí i ZPL II), mnoho základních rysů jazyka je totožných a ZPL z větší části vychází právě z EPL 2.

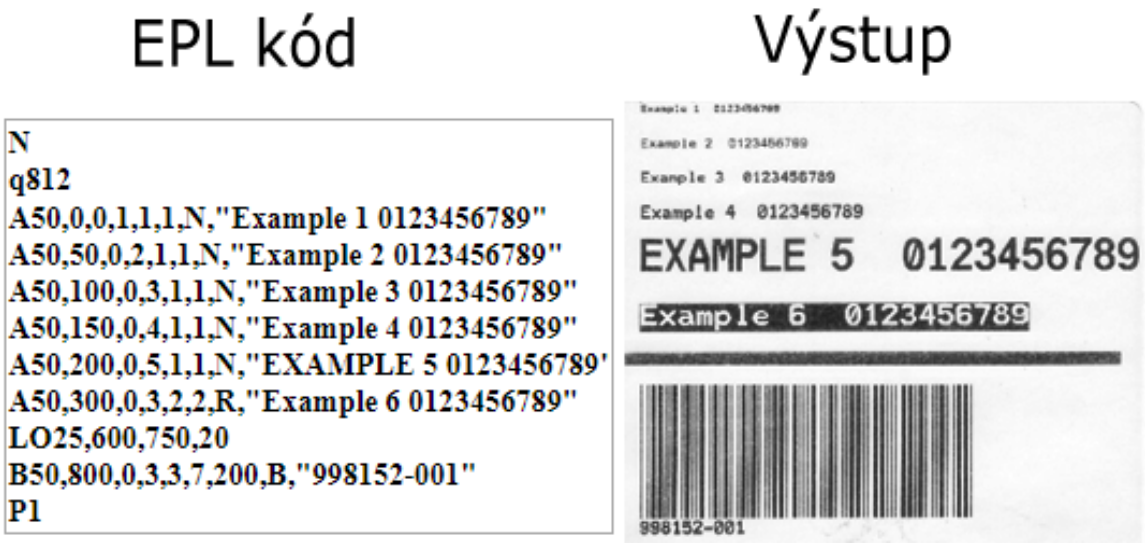
Na rozdíl od toho, co uvádí oficiální dokumentace jazyka, jedná se spíše o značkovací jazyk než o programovací jazyk. Neobsahuje totiž žádné řízení toku, podmínky ani metody. Proměnné jsou taktéž definovány jako očíslovaný obsah nežli klasické proměnné.

EPL 2 funguje na principu příkazů. Jeden řádek odpovídá jednomu příkazu, který tiskárna vykoná. Příkazy jsou tvořeny minimalisticky, aby se ušetřilo místo, paměť a zvýšila se tak rychlost komunikace s tiskárnou. Příkazy jazyka a jejich parametry rozlišují velká a malá písmena. Je třeba dodržet konvence jazyka. Skládají se z jednoho nebo více znaků. Jeden příkaz také může obsahovat i více parametrů, které jsou pevně stanoveny. Pokud parametr není explicitně označen jako volitelný, programátor musí vždy parametr vyplnit.

Klíčovou syntaxí jazyka je zápis řetězců a obrázkových bitmap. Jazyk podporuje tisk bitmap a obyčejných obrázků, ale největší použití má právě pro tisk čárových kódů. Hodnoty řetězců jsou uvedeny standardně ve dvojitých uvozovkách. Řetězec může být hodnota příkazu i libovolný text pro tisk (opatřený správným příkazem).

Zásadním rozdílem oproti ZPL je absence prefixů příkazů. Každý příkaz začíná na novém řádku a neobsahuje žádný prefix. To může vést k častým chybám při manuálním psaní skriptu. V případě opomenutí nového řádku nebo spojením dvou příkazů, tiskárna nerozpozná, co má tisknout a může dojít k přerušení tisku nebo tisku nesprávných hodnot.

Ačkoliv se syntaxe EPL 2 může zdát zdlouhavá a obtížná na napsání, jedná se o základní a krátké příkazy, které tiskárna může snadno zpracovat. Následuje příklad syntaxe EPL 2 a výstupu štítku, obsahující jednoduchý text různých velikostí, grafický objekt a čárový kód.




**Obrázek 11 – Příklad jazyka EPL 2 (zdroj: [18])**

Z příkladu je patrné, že velká část skriptu obsahuje pozicování a nastavení obyčejného textu. Nastavení umístění na osách X a Y veškerých objektů je klíčové pro návrh štítku. Skript také obsahuje velké množství dodatečných parametrů, určující konečný vzhled textu. K pochopení základních příkazů a jejich parametrů poslouží následující tabulka 3.

**Tabulka 3 – Základní příkazy jazyka EPL 2 (zdroj: [Autor])**

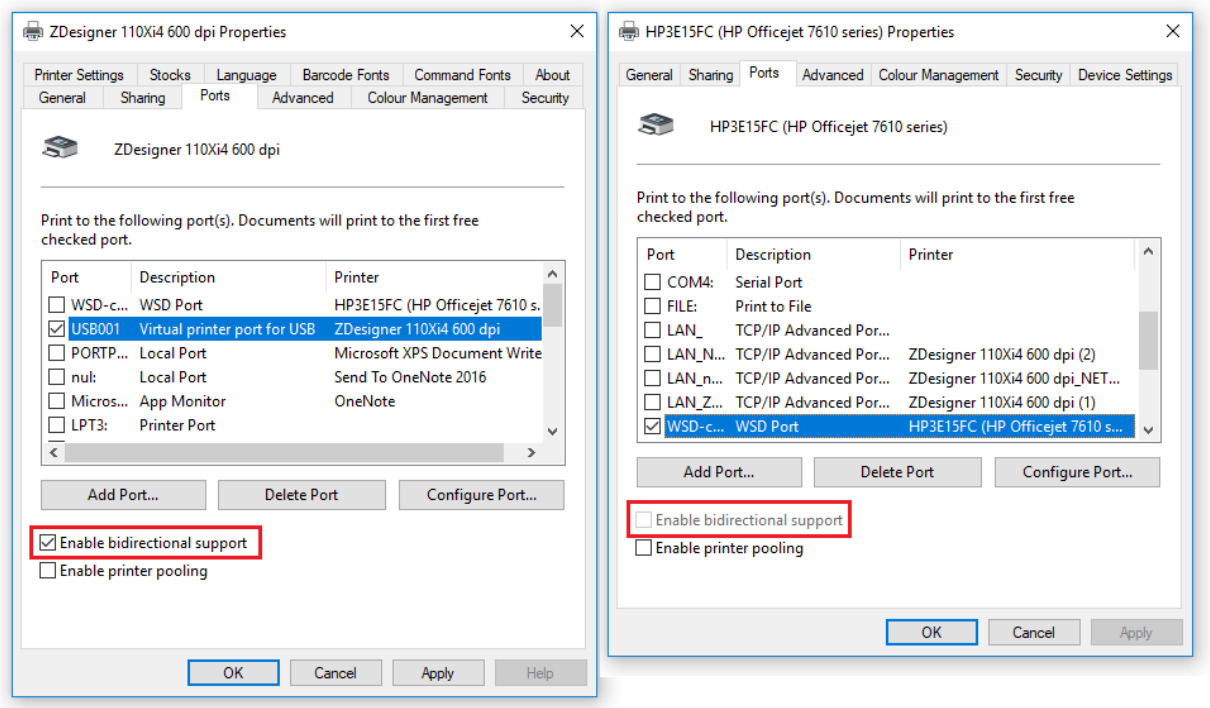
| **Příkaz** | **Popis** |
|---|---|
| **N** | Clear Image Buffer – Vyčistí zásobník pro obrázky, příprava na stavbu nového štítku. |
| **Q812** | Set Label Width – Nastaví šířku oblasti pro tisk. Parametr udává šířku v bodech. |
| **A50,0,0,1,1,1,N, "Text"** | ASCII Text – Standardní text na štítku v kódování ASCII. Parametry příkazu obsahují X a Y osu umístění, rotaci, font, horizontální a vertikální násobič, orientaci a samotná data. Font je zadán číslovkou dle pevně daných specifikací (více v dokumentaci EPL). |
| **Příkaz** | Popis |
| **LO25,600,750,20** | Line Draw Black – Vykreslí černou čáru, přepisující jakékoliv předchozí informace. Parametry určují X a Y osu umístění na štítku, horizontální a vertikální délku čáry. |
| **B50,800,0,3,3,7, 200,B, "Data"** | Bar Code – Vykreslení standardního čárového kódu. Parametry určují X a Y osu umístění, rotaci, typ čárového kódu, šířku čar, výšku čárového kódu, zobrazení textu a samotná data kódu. |
| **P1** | Print – Vytiskne veškerý obsah obrázkového bufferu. Jinými slovy vytiskne veškeré příkazy, které předcházeli tento příkaz. Parametr určuje počet štítků, který se má vytisknout. |

Největší nevýhodou EPL 2 je minimum nástrojů pro tvorbu a zobrazení štítků v tomto jazyce. Jsou těžko dohledatelné nebo částečně nefunkční. Ani vývojáři ze Zebra Technologies neposkytují software na převedení EPL/EPL 2 na grafický štítek, který by uživateli ukázal, jak bude jeho skript vypadat na reálném štítku. Návrh štítku jednoduše pomocí grafického rozhraní se tak stává převážně fikcí a programátoři musí hledat alternativní řešení. I pro tento důvod je lepší zvolit novější jazyk ZPL, se kterým se snáze pracuje, navrhuje a existují i online návrhářské aplikace.

## 2.2 Obousměrná komunikace (bidirectional communication)

Obousměrná komunikace, anglicky bidirectional communication či také bidi communication, je označení pro schopnost tiskárny komunikovat oběma směry. Jedná se o stěžejní prvek specifikace IEEE 1284. První tiskárny měly pouze jednosměrný tok dat. Tedy od zdroje (například počítač) do tiskárny. V některých případech je však potřeba získat zpětnou vazbu od tiskárny nebo si vyžádat nějaká data z tiskárny. Mezi taková typická data patří například stav tisku, obsah paměti tiskárny nebo získání chybového hlášení. Obecně se dá mluvit o monitoringu tiskárny.

V angličtině je toto slovní spojení také často nesprávně zaměňováno za bidirectional printing (také označováno jako boustrophedon printing). To označuje proces tisku z obou stran. Při obyčejném tisku (unidirectional) se tisková hlava pohybuje zleva doprava a po dokončení tisku jednoho řádku se vrací do původní polohy a začíná tisk druhého řádku. V obousměrném tisku se hlavice nevrací doleva do výchozí polohy, aby začala nový řádek, ale při pohybu doleva již rovnou probíhá samotný tisk. To umožňuje mnohem efektivnější tisk a méně pohybů tiskové hlavy. Mnoho moderních tiskáren tento proces podporuje.




**Obrázek 12 – Povolení obousměrné komunikace (zdroj: [Autor])**

Obousměrná komunikace tiskáren je v dnešní době již standardem. Avšak stále se vyrábějí typy tiskáren, které obousměrnou komunikaci nepodporují. Záleží na výrobci a modelu tiskárny. Podporu obousměrné komunikace lze vyčíst z dokumentace a specifikací tiskárny nebo pomocí správce zařízení v operačním systému. Operační systém Windows umožňuje zapnout či vypnout obousměrnou komunikaci s tiskárnou (pokud jej tiskárna poskytuje). Na obr. 12 je zobrazeno nastavení dvou tiskáren. Tiskárna vlevo poskytuje obousměrnou komunikaci a je v systému povolena. Tiskárna vpravo neposkytuje tento druh komunikace, možnost je zašedlá a nelze ji zvolit.

Výhodou obousměrné komunikace je využití informací, které tiskárna poskytuje po síti. Jakékoliv zařízení na stejné síti může získat například status tiskárny nebo monitorovat stav tiskárny a čekat na jeho změnu. Aplikace mohou tohoto monitoringu využívat a uživateli sdílet užitečné informace, jako například v jakém stavu se tiskárna nachází, připravenost tiskárny, stav toneru či barev v tiskárně, nastavení tiskárny a v případě chyby může poskytovat konkrétní chybová hlášení (nedostatek barvy nebo toneru, chybějící papír, zaseknutá tisková hlava a další). Uživatel může na tyto informace patřičně reagovat a usnadní se mu práce s tiskárnou, protože bude vědět, co se děje nebo kde může být chyba.

Jelikož se jedná o další funkci tiskárny navíc má to i svoje nevýhody. Udržování komunikace pro zpětnou komunikaci a monitoring tiskárny vyžaduje další zdroje. Tiskárna poskytuje pro tuto funkci vnitřní vyrovnávací paměť. Vypnutím této funkce se zruší i dodatečná vyrovnávací paměť a dojde tedy k navýšení paměti určené pro tisk a výkonu samotné tiskárny. Záleží pouze na uživateli a úkolu tiskárny, zda se vyplatí mít otevřený zpětný komunikační kanál. V případě, že je tiskárna využívána pro tisk a zpracování velkého množství příkazů, obrázků a dat, jinými slovy, pokud tiskárna potřebuje výpočetní výkon pro provedení příkazů a uživatele nezajímají chyby, které nastanou, je lepší vypnout podporu obousměrné komunikace. Zpravidla je to případ velkovýrobního tisku na průmyslových tiskárnách.

## 2.3 Kódování a komprese

Při komunikaci s tiskárnou je snaha redukovat počet a velikost odesílaných dat. To zajišťuje kódování (encoding) a komprese (compression) dat. Ačkoliv mnoho moderních standardů připojení podporuje vysoké rychlosti přenosu dat, konečná rychlost přenosu a zpracování nezávisí pouze na připojení, ale také na tiskárně. Tiskárna musí přijmout data do vyrovnávací paměti a data zpracovat. V případě pomalého procesoru tiskárny a pomalé vyrovnávací paměti může klesnout rychlost přenosu na řády kilobajtů, proto je důležité odesílat jen nezbytně nutná data a použít vhodnou kompresi dat. Komplexní komprese může značně zredukovat velikost posílaných dat, ale ušetřený čas přenosem je pak vynaložen na následnou dekomprimaci procesorem uvnitř tiskárny. Je nutné zvolit střed mezi objemem dat a mírou komprese.

### 2.3.1 Kódování

Kódování je proces změny datové reprezentace. Stejná data mohou být reprezentována různými způsoby. Kódování nijak nemodifikuje obsah a složení dat. Volba jiného kódování dat může mít za následek zvýšení nebo snížení velikosti dat. Kódování není určeno k ochraně dat.

Mezi typické zástupce kódování patří binární, hexadecimální, jednobajtové, vícebajtové a kódování Base64. ZPL II používá pro text ve výchozím nastavení jednobajtové kódování (SBCS). To znamená, že jeden bajt odpovídá jednomu tisknutelnému znaku. Konkrétně znakovou sadu U.S.A. A. Jednobajtové kódování může pojmout pouze 256 symbolů a česká písmena, cyrilice ani řecká abeceda nejsou podporována.

ZPL II umožňuje nastavit každému datovému poli jiné kódování pomocí příkazu `^CI`. Několik příkladů podporovaného kódování je uvedeno v tabulce 4.

**Tabulka 4 – Příklady kódování ZPL II (zdroj: [11])**

| Hodnota ^CI | Kódování |
|---|---|
| 0 | Single Byte Encoding |
| 14 | Double Byte Asian Encodings |
| 28 | Unicode – UTF-8 Encoding |
| 29 | Unicode – UTF-16 Encoding |
| 35 | CodePage 1254 |

### 2.3.2 Komprese

Komprese (či také komprimace) je technika, která využívá redundance v datech ke zmenšení jejich objemu. Cílem je zmenšit data pro snadnější přenos nebo uložení. Komprimovaná data jsou vhodná pro archivaci nebo sdílení přes síť s nízkou rychlostí přenosu. Podobně jako kódování, komprese není určena k šifrování nebo ochraně dat. Dělí se na ztrátovou a bezeztrátovou.

Ztrátová komprese, jak název napovídá, úmyslně vypouští některé informace a tím redukuje velikost dat. Tato komprese je permanentní a informace, které byly vypuštěny, jsou nenávratně ztraceny. Tento typ komprese se často používá u obrazu a zvuku, kde se dá určitá míra ztrát tolerovat a může být méně znatelná. Typickým příkladem ztrátové komprese je obrázkový formát JPEG nebo zvukový formát mp3.

Bezeztrátová komprese zachovává všechny informace a následnou dekompresí získáme původní data. Nevýhodou je menší efektivita a zmenšení dat oproti ztrátové kompresi. Avšak pro komprimaci textu je bezeztrátová komprese naprosto nezbytná, protože ztráta, byť jediného znaku by znamenala ztrátu celých dat nebo jejich hodnotu. Pro obrázky je možné použít bezeztrátovou kompresi PNG, zatímco pro text je vhodná jakákoliv varianta algoritmu LZ.

### 2.3.3 B64 a Z64

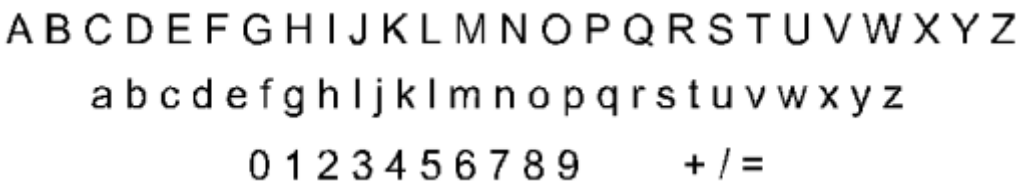
Data jako je obrázek, grafický objekt nebo font jsou v ZPL II kódována pomocí hexadecimálního ASCII kódování. Hexadecimální ASCII kódování dat přesně reprezentuje zdrojová data a neposkytuje žádnou komprimaci. V případě velkého nebo komplexního obrázku je tato podoba nevhodná a přenos dat do paměti tiskárny může být zdlouhavý.

Tento problém řeší ZPL II dvěma způsoby, a to kódováním Base64 (zkráceně B64) a komprimovacím algoritmem LZ77 (zkráceně Z64).

#### Base64

Base64 je typ kódování, které používá pouze ASCII znaky pro reprezentaci dat. Název vychází z faktu, že kódování využívá pouze 64 znaků. Často se využívá pro reprezentaci obrázků, grafických objektů, příloh e-mailu nebo obecně binárních dat. Výhodou Base64 kódování ZPL je absence speciálních řídících znaků, kterými jsou ve výchozím nastavení stříška (^) a tilda (~). Base64 nemá za úkol data komprimovat a zmenšovat. Ve většině případů opak navýší velikost dat.

Ačkoliv se to zdá kontraproduktivní, má to svůj význam pro přenos dat. K snadnému přenosu binárních dat je Base64 ideální, a navíc odstraňuje veškeré speciální znaky, které by mohly způsobovat komplikace při přenosu. Obr. 13 ilustruje dostupné znaky v kódování Base64.




**Obrázek 13 – Dostupné znaky v kódování Base64 (zdroj: [11])**

Tiskárna kontroluje veškerá data kódována pomocí Base64 nebo data, která jsou komprimována. Po provedení kódování Base64 se provede výpočet CRC-16 nad kódovanými daty. Výsledný CRC se v podobě čtyřciferného hexadecimálního čísla připojí na konec kódovaných dat za dvojtečku. CRC, česky cyklický redundantní součet, je metoda detekce nechtěných změn nebo chyb během komunikace. Tiskárna si po přijetí dat spočítá vlastní CRC nad přijatými daty a porovná jej s CRC, které bylo odesláno společně s daty. Pokud obě CRC nesedí, tiskárna veškerá přijatá data zahodí a ignoruje je. Příklad Base64 a CRC je ilustrován na obr. 14.

**Text k zakódování**

^XA^CI28^FDPříklad ZPL II^FS^XZ

**Zakódovaný text v Base64 s CRC**

XlhBXkNJMjheRkRQxZnDrWtsYWQgWlBMIElJXkZTXlha:A768

**Obrázek 14 – Příklad Base64 a CRC-16 (zdroj: [Autor])**

### LZ77

Před provedením kódování pomocí Base64 je možné data zpracovat a aplikovat na ně určitou míru komprese. Tu v ZPL II zajišťuje algoritmus LZ77, v ZPL dokumentaci označován jako Z64.

LZ77 je bezeztrátový algoritmus jehož autory jsou Abraham Lempel a Jacob Ziv. Číslovka 1977 označuje rok publikování. Je také označován jako LZ1 nebo Lempel-Ziv algoritmus. Dalo by se říci, že se jedná o slovníkovou metodu, ačkoliv využívá tzv. posuvné okno (sliding window).

Posuvné okno se posouvá po datovém toku a je rozděleno na prohlížecí okno (search buffer) a aktuální okno (look-ahead buffer). Jejich velikost je konstantní a posouvá se vždy celé okno. Prohlížecí okno obvykle obsahuje místo pro několik desítek tisíc znaků. Naopak aktuálního okno využívá pouze pár desítek znaků, protože delší shody většinou neexistují. Velikost aktuálního okna by teoreticky měla být menší nebo stejná jako prohlížecí okno. Algoritmus pracuje pouze s daty uvnitř posuvného okna, takže paměťová náročnost algoritmu neroste s délkou vstupního řetězce.

Algoritmus začíná s prázdným prohlížecím oknem a do aktuálního okna načte vstupní data (dle kapacity aktuálního okna). V každém kroku se vyhledá nejdelší slovo, které začíná v prohlížecím okně a je shodné s nějakou předponou v aktuálním okně. V případě nalezení je slovo zakódováno a posunuto o počet shodných znaků + 1. Posuvné okno se posouvá, dokud není aktuální okno prázdné (konec vstupních dat). Výstup tvoří zakódovaná slova ve tvaru `(i, j, z)`, kde *i* označuje vzdálenost slova od začátku aktuálního okna, *j* je délka slova a *z* je samotné zakódované slovo nebo znak. Příklad komprese je uveden na obr. 15.

| Krok | Prohlížecí okno | Aktuální okno | Zbytek vstupu | Výstup | Posun |
|---|---|---|---|---|---|
| Inicializace | _____ | ABR | AKADABRA | - | - |
| 1 | ___A | BRA | KADABRA | (0, 0, A) | 1 |
| 2 | __AB | RAK | ADABRA | (0, 0, B) | 1 |
| 3 | _ABR | AKA | DABRA | (0, 0, R) | 1 |
| 4 | BRAK | ADA | BRA | (3, 1, K) | 2 |
| 5 | AKAD | ABR | A | (2, 1, D) | 2 |
| 6 | ADAB | RA_ | | (4, 1, B) | 2 |
| 7 | DABR | A__ | | (0, 0, R) | 1 |
| 8 | ABRA | ___ | | (3, 1, konec) | 2 |

**Obrázek 15 – Příklad LZ77 komprese (zdroj: [24])**

V uvedeném příkladu se komprimuje slovo ABRAKADABRA s prohlížecím oknem velkém 4 znaky a aktuálním oknem o velikosti 3 znaky. Výsledek LZ77 algoritmu je následující řetězec.

```
(0, 0, A) (0, 0, B) (0, 0, R) (3, 1, K) (2, 1, D) (4, 1, B) (0, 0, R) (3, 1, konec)
```

Postup zpětné dekomprimace je ilustrován na obr. 16.

| Krok | Vstup | Buffer | Přidáno | Výstup |
|---|---|---|---|---|
| 1 | (0, 0, A) | | A | A |
| 2 | (0, 0, B) | A | B | AB |
| 3 | (0, 0, R) | AB | R | ABR |
| 4 | (3, 1, K) | ABR | AK | ABRAK |
| 5 | (2, 1, D) | ABRAK | AD | ABRAKAD |
| 6 | (4, 1, B) | ABRAKAD | AB | ABRAKADAB |
| 7 | (0, 0, R) | ABRAKADAB | R | ABRAKADABR |
| 8 | (3, 1, konec) | ABRAKADABR | A, konec | ABRAKADABRA |

**Obrázek 16 – Příklad LZ77 dekomprese (zdroj: [24])**

Kompresní algoritmus LZ77 je jednoprůchodový (vstupní data prochází pouze jednou a jedním směrem), adaptivní (dokáže se přizpůsobit různým délkám řetězců) a asymetrický (dekomprimace řetězce je mnohem rychlejší než komprimace, protože při komprimaci je nutné vytvořit celý slovník, zatím co u dekomprimace je slovník již k dispozici. Algoritmus je také méně náročný na paměťový prostor (dle velikosti oken). Obecně se algoritmus LZ77 označuje jako neefektivní a zastaralý. Jeho novější a výkonnější varianty se nazývají LZW, LZSS a LZMA. Komprimace LZMA je dále využita v praktické části práce.

V ZPL II lze použít kódování Base64 a komprimační algoritmus LZ77 v následujících příkazech:

- ~DB – Download Bitmap Font,
- ~DE – Download Encoding,
- ~DG – Download Graphic,
- ~DL – Download Unicode Bitmap Font,
- ~DS – Download Scalable Font,
- ~DT – Download TrueType Font,
- ~DU – Download Unbounded TrueType Font,
- ^GF – Graphic Field (s kompresním typem nastaveným na „ASCII Hex“).

Zmíněné příkazy obsahují parametr datové pole (Data Field). Tento parametr může obsahovat komprimovaná data. Datové pole musí striktně dodržet následující formát.

```
ID:Encoded_Data:CRC
```

ID je označení použitého kódování a komprese. Mezi validní identifikátory patří B64 a Z64. Encoded_Data jsou samotná kódovaná, resp. komprimovaná data a CRC je vypočítaný kontrolní součet (CRC-16 zmíněný výše).

# 3 TISK

Technologie moderního tisku není přímočará a jasná jako tisk na ručním tiskařském lisu. Mnoho částí procesu tisku je obyčejnému uživateli skryto pod krytem tiskárny. Jediné, co uživatel vidí je editor s požadovanou stránkou a konečný výsledek ve formě vytisknutého štítku nebo stránky. Zpracování obrazu a řízení tisku probíhá ve většině případů přímo v tiskárně, nezávisle na uživateli. Kapitola tisku se blíže věnuje zpracování obrazu v tiskárně a principu samotného tisku.

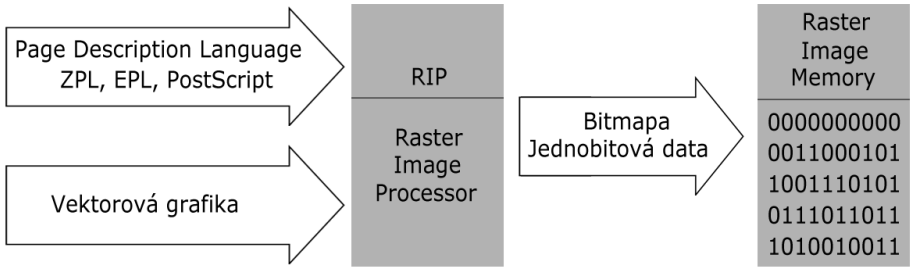
Dále se kapitola věnuje vybraným metodám tisku. Jak bylo zmíněno v kapitole 1, je třeba zvolit vhodný materiál. S materiálem a kvalitou tisku také souvisí různé metody tisku. Metod je několik a nelze je použít na všechny typy materiálů. Kapitola se blíže věnuje tepelnému tisku.

Kapitola vychází ze zdrojů [26–35].

## 3.1 Raster Image Processor

Procesor rastrového obrazu, zkráceně RIP, je komponenta umístěná v systému tiskárny, která převádí jazyk pro popis stránky a vektorovou grafiku na rastrový obrázek (bitmapu). Jedná se o klíčový prvek tiskárny, který poskytuje tiskárně výpočetní výkon. Výslednou bitmapu tvoří jednobitová data, která lze snáze zobrazit fyzickými zobrazovacími zařízeními. Obr. 17 ilustruje tok dat přes procesor rastrového obrazu.

Tisk probíhá ve formě maticové mřížky. Bez tohoto převodu není tiskárna schopná vytisknout požadovaný obrázek nebo stránku. Bity v bitmapě určují, jaké části mřížky se mají aktivovat. U inkoustových tiskáren aktivované části mřížky určují, kde se má nanést barva. Oproti tomu laserová tiskárna používá mřížku pro změnu elektrostatických nábojů na elektrostatickém válci. Každá barva má samostatný rastrový obrázek, který nabíjí válec na správném místě, aby připravil danou barvu toneru na správné místo.




**Obrázek 17 – Tok dat přes RIP (zdroj: [Autor])**

První procesory rastrového obrazu byly vytvořeny jako elektronické hardwarové komponenty, které získaly popis stránky pomocí sériového portu RS-232. RIP může být implementován také jako softwarová komponenta uvnitř operačního systému tiskárny nebo jako firmware program v mikroprocesoru tiskárny. Tato metoda je často použita v novějších typech tiskáren a v některých případech lze použít i kombinaci softwarové a hardwarové komponenty. Například každá tiskárna podporující jazyk PostScript obsahuje RIP jako softwarový firmware.

### 3.1.1 Etapy zpracování v RIP

Po obdržení vstupních dat, procesor rastrového obrazu začne data zpracovávat. Zpracování probíhá ve třech etapách, které se nazývají interpretation, rendering a screening. Tyto etapy zajišťují transformaci dat na jednoduché instrukce pro tiskárnu.

#### Interpretation

První etapou je interpretace. V této části obdrží procesor rastrového obrazu vstupní data ve formátu vektorové grafiky nebo jazyku pro popis stránky. Přijatá data převede na interní reprezentaci stránky ve své paměti. Velká většina procesorů rastrového obrazu zpracovává stránky sériově. Po zpracování jedné stránky se připraví stránka následující. Vždy se musí dokončit celý cyklus pro danou stránku.

#### Rendering

Po dokončení interpretace a přeložení vstupních dat se přechází na vykreslovací etapu. Proces vykreslování získá interní reprezentaci stránky z paměti a převede ji na bitmapu s nepřerušovaným tónem. Etapa vykreslování zajišťuje základ pro výslednou bitmapu. Více sofistikované procesory rastrového obrazu provádí interpretaci a vykreslení dohromady v rámci jedné etapy. Rozdělení prvních dvou etap může mít smysl u jednodušších jazyků, které jsou navrženy tak, aby fungovaly na minimálním hardwaru, takže mají tendenci sami přímo řídit vykreslování.

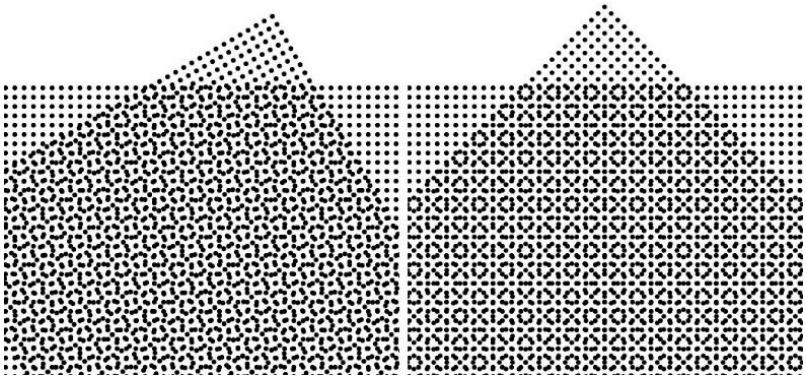
#### Screening

Konečnou a nezbytnou etapou zpracování je screening. Screening provádí další transformaci dat. Vstupem do screening procesu je bitmapa z vykreslovací etapy. Následně screening převede bitmapu s nepřerušovaným tónem na polotónovou bitmapu.

Polotón je technika zobrazení, která používá pravidelného rozmístění bodů různých velikostí, aby simulovala plynulé přechody v obrázku. Technika polotónu spoléhá na optickou iluzi a nedokonalost lidského oka. Díky tomu splývají jednotlivé body do plynulých přechodů.

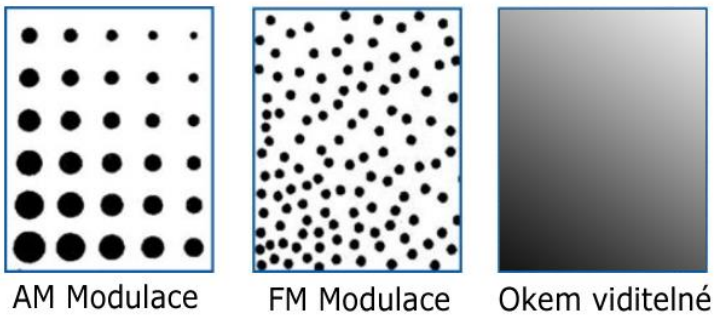
Screening lze provádět dvěma metodami. První a nejčastější metodou screeningu je amplitudová modulace (AM). Ta vytváří pravidelnou síť bodů různých velikostí. Je založená na geometrické a pevné vzdálenosti bodů. Síť obsahuje vždy stejný počet bodů. Výhodou AM je jednoduchost a malá výpočetní náročnost.

Nevýhodou je možný vedlejší efekt zvaný Rosette Pattern, volně přeloženo jako růžicový vzor. Jedná se o vzor, který mohou body vytvářet pod určitým úhlem či z určité perspektivy. Podobá se květině růží a degraduje kvalitu tisku. Obr. 18 ilustruje růžicový vzor.



**Obrázek 18 – Rosette Pattern (zdroj: [28])**

Druhou metodou je frekvenční modulace (FM), též známá jako stochastický screening. Metoda frekvenční modulace je založená na pseudonáhodném rozložení polotónových bodů. Všechny body jsou stejně velké. Hustota bodů vytváří tmavší nebo světlejší oblasti obrazu. Umístění bodů může řídit i sofistikovaný matematický algoritmus. Body jsou velké pouze několik mikrometrů (do 10 μm). Malá velikost zajišťuje větší přesnost a hladkost přechodů což vede k lepší kvalitě tisku. FM screening úplně odstraňuje Rosette Pattern, který vytváří AM screening. Rozdíly jednotlivých metod screeningu a výsledný obrazec viditelný lidským okem je ilustrováno na obr. 19.




**Obrázek 19 – Metody screeningu (zdroj: [29])**

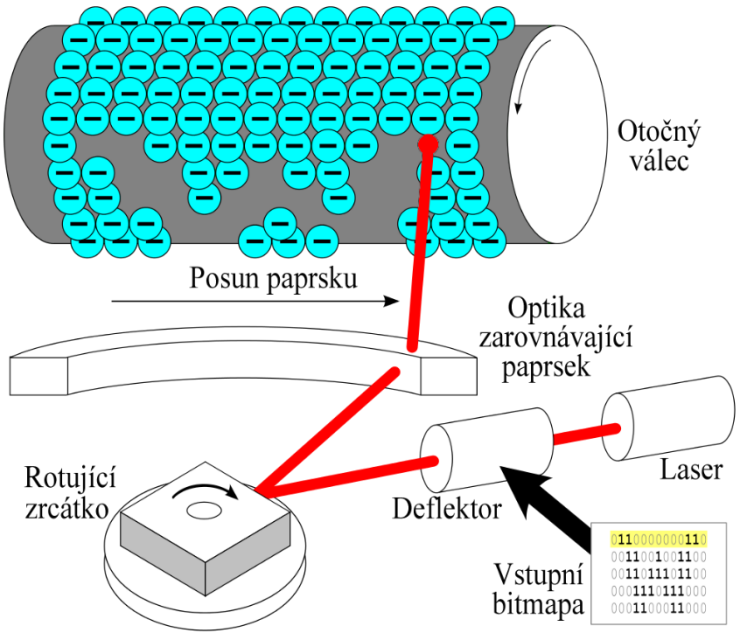
## 3.2 Technologie tisku

Technologie tisku ve velké míře ovlivňují kvalitu, cenu a rychlost tisku, podobně jako zvolený materiál. Při tisku je nutné zvážit masovou produkci štítků, kde se budou štítky používat a zda budou obsahovat čárové kódy nebo obrázky.

Mezi standardní tiskové technologie štítků patří laserový, inkoustový a tepelný tisk. Pro tisk štítků se nejčastěji využívá tepelný tisk. Ten je preferován na základě nízké ceny, vysoké kvality a také z důvodu snadného tisku. V následujícím textu budou popsány jednotlivé technologie tisku.

### 3.2.1 Laser printing

Laserový tisk pracuje na principu elektrostatického náboje. Na začátku každé rotace fotorecepčního válce je válec nabit negativním nábojem. Následně se pomocí laserového paprsku nanáší obraz na nabitý fotorecepční válec. Laser vybíjí negativně nabité náboje do středu válce. Toner je nabitý na stejnou polaritu jako válec (tedy negativní). Jelikož se stejné polarity odpuzují, na fotorecepčním válci se uchytí toner pouze na kladně nabitých místech. Uchycený toner na válci se tlakem obtiskne na potiskovaný materiál. Ilustrace vybíjení fotorecepčního válce je zobrazena na obr. 20.




**Obrázek 20 – Vybíjení fotorecepčního válce (zdroj: [31])**

Laserová technologie tisku byla vytvořena pro tisk kancelářských dokumentů. Ačkoliv mají vysokou rychlost tisku, nejsou vhodnou volbou pro velkoobjemový tisk štítků. Cena za tisk je vysoká a lepící materiál ze štítků se může hromadit uvnitř tiskárny a snižovat její výkon. To následně vede k blokování fotorecepčního válce a předčasnému selhání tiskárny.

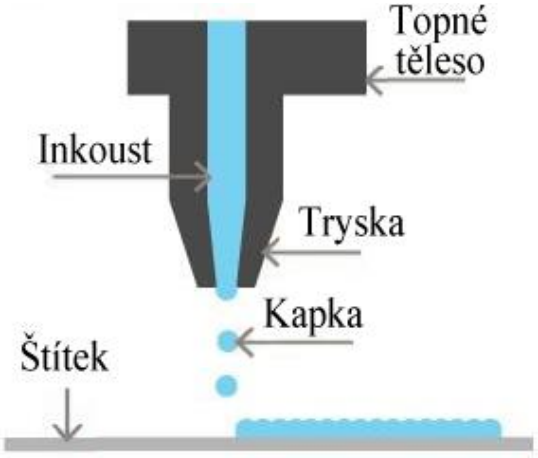
Existují štítky určené přímo pro laserový tisk, které obsahují slabší lepidlo. Avšak takové štítky nejsou vhodné pro dlouhodobé používání nebo pro drsnější prostředí. Obzvlášť nevhodné jsou pro tisk čárových kódů. Čárové kódy potřebují přibližně padesátiprocentní pokrytí inkoustem, což podstatně zvyšuje spotřebu toneru.

Laserový tisk štítků se používá jen v krajních případech. Používá se na místech, kde není vyžadována velká kvalita štítků a velkoobjemový tisk. Ze všech tří technologií je laserový tisk nejdražší (vizte tabulka 5).

### 3.2.2 Inkjet printing

Inkoustový tisk využívá k tisku kapičky inkoustu, které jsou velkou rychlostí vymršťovány na tiskovou plochu. Velikost kapek ovlivňuje kvalitu tisku (dnes řádově pikolitry). Obr. 21 ilustruje princip inkoustového tisku. Inkoustový tisk má podobná omezení a nevýhody jako laserový tisk. Ačkoliv inkoustové tiskárny jsou levné a mají nízké počáteční náklady, z dlouhodobého hlediska nejsou finančně výhodné z důvodu vysokých a opakujících se nákladů na výměnu a doplňování inkoustových kazet. Největší nevýhodou pro tisk štítků je kvalita tisku. Inkoustové tiskárny nemají konzistentní kvalitu a nedosahují kvalit laserového tisku. Důsledkem nízké úrovně barvy v tiskárně může docházet k vybledlé kvalitě, hluchým místům nebo dokonce pruhům napříč celého štítku. Tiskové nedokonalosti a vady jsou při tisku čárových kódů nepřijatelné, protože mohou změnit původní data čárového kódu nebo znemožnit jeho přečtení skenerem. Podobné problémy se vztahují i na text a jeho nečitelnost.

Inkoustový tisk štítků využívají firmy, které nechtějí investovat do drahých laserových tiskáren, tisknou malé množství štítků a neočekávají vysokou kvalitu štítků.




**Obrázek 21 – Princip inkoustového tisku (zdroj: [32])**

### 3.2.3 Thermal printing

Tepelná tiskárna pracuje na bázi řízeného tepla, pomocí kterého vytváří výsledné obrazce na štítku. Jedná se o nejpoužívanější metodu tisku štítků. Jejich největší výhodou je levný tisk a velkoobjemová produkce štítků. Na rozdíl od inkoustových tiskáren, tepelné tiskárny zachovávají konzistenci a kvalitu tisku. S velkým počtem vytisknutých štítků neklesá jejich kvalita. Z toho důvodu jsou oblíbené pro tisk obrázků a čárových kódů. Toho se často využívá na poštovních a přepravních štítcích. Tabulka 5 porovnává náklady několika štítků při tisku různými technologiemi.

**Tabulka 5 – Porovnání nákladů na tisk štítků (zdroj: [30])**

| Typ štítku (rozměry v palcích) | Laserový tisk | Inkoustový tisk | Tepelný tisk |
|---|---|---|---|
| **1000 poštovních štítků (1 x 25/8)** | $9.06 | $9.54 | $5.11 |
| **1000 přepravních štítků (3¾ x 4)** | $134.42 | $59.04 | $35.71 |

Z uvedených dat vyplývá, že tepelný tisk je značně levnější oproti ostatním tiskovým technologiím. Tepelná technologie se dělí na dvě kategorie. Termotisk a tisk tepelným přenosem. I přes veškeré výhody, tepelný tisk má i svoje nevýhody. Není vhodný pro běžný tisk nebo tisk štítků velkých rozměrů. Většina modelů tiskáren je optimalizována na tisk štítků o šířce od 5 do 20 centimetrů.

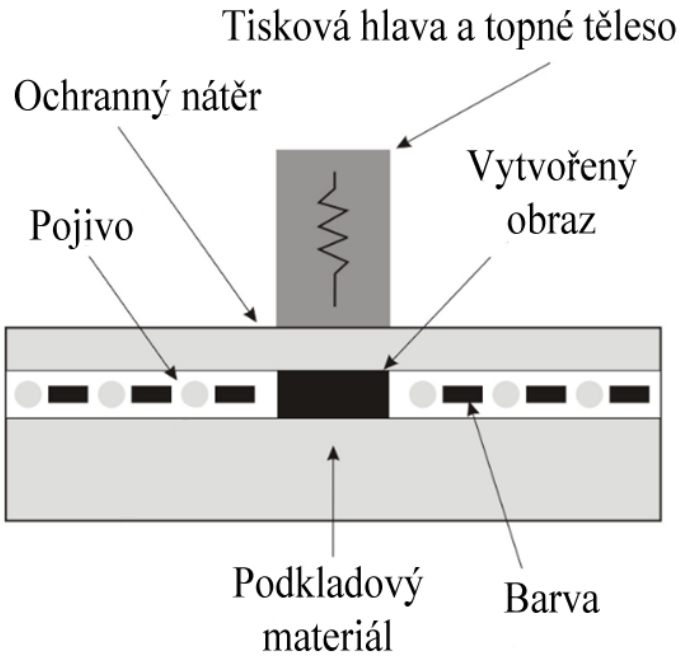
#### Direct Thermal Printing

Termotisk nebo také přímý tepelný tisk vyžaduje přímý kontakt štítkového materiálu s tiskovou hlavou. K tisku využívá speciálního termopapíru, který reaguje na teplo. Tisková hlava obsahuje topná tělesa, která zahřívají části štítku. Teplo způsobí změnu barvy a tisk výsledného obrazce. Princip termotisku je ilustrován na obr. 22.

Velká část termotisku je právě kvůli způsobu tisku limitována pouze na černobílý tisk. V případě tisku čárových kódů a textů to není žádný problém. Termotisk dokáže vytisknout i jednoduchá černobílá loga a obrázky. Avšak barevný tisk pomocí termotisku není zcela nemožný. Firma Zebra Technologies má patentované speciální barevné termopapíry. Tyto termopapíry, označované jako Zebra IQ Color, obsahují neviditelný, barevný inkoust, který se taktéž aktivuje teplem. Pro aktivaci barvy není potřeba žádná dodatečná modifikace tiskárny, pouze se vymění materiál pro štítek. Na jednom štítku je možné aktivovat až tři různé barevné odstíny.

Největší výhodou termotisku je jeho jednoduchost a snadná údržba. Termotisk nepotřebuje žádné inkoustové kazety, tonery ani barevné pásky. Jedinou komponentou, která se musí při tisku vyměňovat je termopapír. To šetří nejen čas a peníze, ale také rozměry tiskárny. Termotisk se často používá pro tisk účtenek, potvrzení nebo přepravních a poštovních štítků.

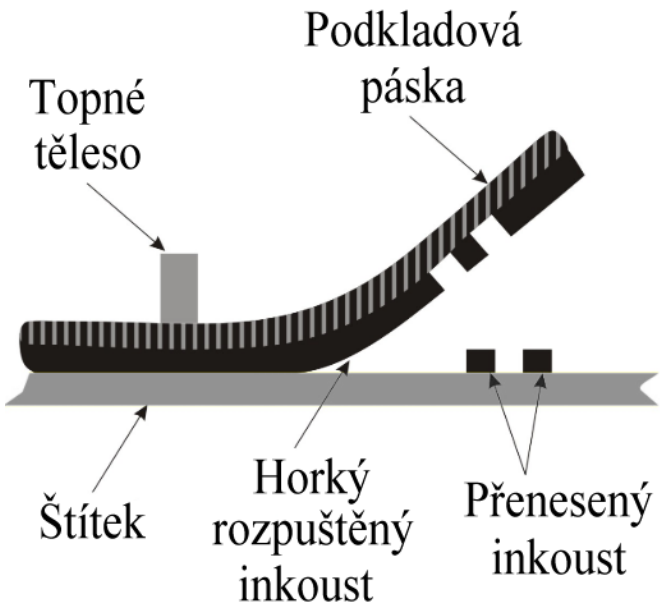
Jelikož byl štítek vytvořen za pomocí tepelné aktivace a je citlivý na teplo, je třeba dbát na správné skladování a zacházení se štítkem. Termotisk není vhodný na místech, kde se očekává vysoká teplota, nadměrné sluneční záření, chemikálie nebo dlouhodobé používání. Nevhodné podmínky mohou způsobit vyblednutí štítku nebo naopak neúmyslné ztmavnutí celého štítku. To by mohlo nenávratně poškodit štítek a data, která obsahuje. Termotisk je vhodný pro tisk dočasných štítků nebo štítků u kterých se neočekává dlouhá životnost, právě kvůli postupné degradaci štítku.




**Obrázek 22 – Princip termotisku (zdroj: [33])**

### Thermal-Transfer Printing

Druhou metodou tepelného tisku je tisk tepelným přenosem. Technologie využívá topného tělesa a podkladové pásky s inkoustem (backing ribbon). Celý proces začíná podobně jako u termotisku. Topné těleso se nahřeje pouze na místech, která jsou určena vstupní bitmapou. Nahřátá místa následně rozpustí inkoust na podkladové pásce, který se tlakem přichytí na materiál štítku. Proces probíhá téměř okamžitě a v době, kdy štítek vyjede z tiskárny, je barva již zaschlá. Princip tepelného přenosu je ilustrován na obr. 23.




**Obrázek 23 – Princip tisku tepelným přenosem (zdroj: [33])**

Materiál štítku nemusí být ničím speciální, může se jednat o obyčejný kus papíru. Jediným požadavkem na materiál je přilnavost inkoustu z podkladové pásky. Ta je většinou vyrobena z vosku, pryskyřice nebo kombinací těchto dvou složek. Barva inkoustu podkladové pásky bývá standardně černá, ale je možné zakoupit i barevné odstíny. Vzhledem k metodě tisku nelze kombinovat barevné odstíny na štítku a tisknout ve více barvách jako u Zebra IQ Color štítků. Celý štítek má takovou barvu, jakou barvu má inkoust na podkladové pásce.

Nevýhodou této technologie je nutnost výměny podkladové pásky s inkoustem. Na rozdíl od termotisku, technologie tepelného přenosu vyžaduje větší míru údržby a dodatečné dokupování podkladových pásek s inkoustem. To může vést k vyšším nárokům na tisk.

Přilepení inkoustu na štítkový materiál zajišťuje mnohem větší odolnost vůči nevlídnému prostředí, chemikáliím a výkyvům teplot, než kterou poskytuje termotisk. Technologie tisku tepelným přenosem se často využívá na štítcích, které mají vydržet déle než půl roku. Typické použití je pro inventární štítky, štítky majetku a štítky pro identifikaci produktů.

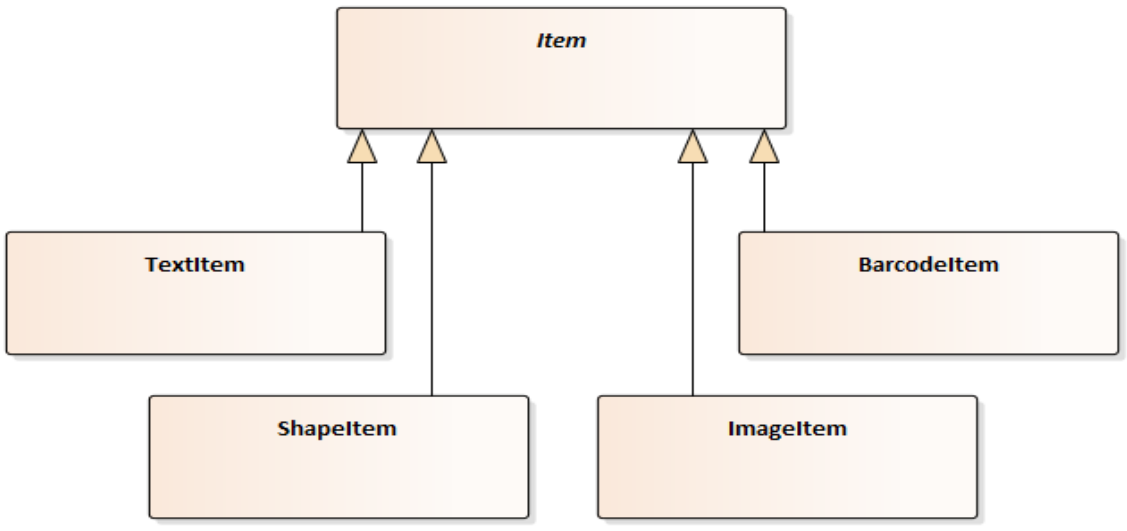
## 3.3 Neodynamic SDK

Neodynamic SRL je soukromá společnost založená v roce 2004, která se zabývá návrhem a vývojem štítkové a tiskové technologie. Zaměřuje se na vývoj nástrojů a knihoven pro .NET, PHP a webové prostředí. Firma Neodynamic má zákazníky ve více než 100 zemích po celém světě. Mezi známé zákazníky patří firmy Sony, Honda a IBM.

Úzce spolupracuje s firmami Zebra Technologies a Honeywell-Intermec. Vzhledem k této spolupráci jejich knihovny podporují jazyky ZPL, ZPL II, EPL a Fingerprint. Zároveň je software (dále označovaný jako Neodynamic SDK nebo zkráceně SDK) optimalizován na tepelné tiskárny těchto dvou velkých firem.

Neodynamic je odborníkem v oblasti čárových kódů. Neodynamic SDK podporuje širokou škálu čárových kódů, včetně těch nejpoužívanějších jako jsou Code 39, Code 128 a EAN 13. V současné době poskytují podporu pro 166[6] typů čárových kódů.

Praktická část diplomové práce využívá konkrétně knihovny Neodynamic ThermalLabel SDK. Ta poskytuje vykreslovací plátno (canvas) pro grafické objekty, které lze různými způsoby modifikovat a postavit nad nimi další rozšíření. Jedná se o rozsáhlou grafickou knihovnu poskytující různé měrné jednotky, jejich konverzi, vykreslování obrázků a generování čárových kódů. Dále umožňuje na základě vstupní šablony generovat příkazy ZPL, ZPL II, EPL a Fingerprint. V práci je zejména využíváno generování PDL.




**Obrázek 24 – Vztah grafických tříd Neodynamic SDK (zdroj: [Autor])**

[6] https://www.neodynamic.com/barcodes

Výhodou SDK je jeho přenositelnost a kompatibilita. Knihovna je vyvinuta v technologii .NET Standard, která zajišťuje podporu pro vývoj aplikací v technologiích UWP, ASP.NET, Xamarin, .NET Framework a .NET Core.

Obr. 24 zobrazuje vztah tříd, které lze vykreslit na plátno. Základní grafické objekty poskytuje samotné SDK a jejich vlastnosti se dále nastavují vlastním uživatelským rozhraním, které je popsáno v kapitole 6. SDK dokáže generovat XML výstup z existujících grafických objektů.

Jednou z dalších funkcí SDK je tisk štítků. SDK obsahuje komplexní tisk štítkových šablon definovaných pomocí XML. Tisk je realizovaný pomocí tříd *PrintJob, WindowsPrintJob* a *WebPrintJob*. Tisk může probíhat lokálně nebo pomocí webového rozhraní. Tato funkce však v rámci diplomové práce nebyla využita a tisk byl implementován vlastním způsobem. SDK také umožňuje export štítků do různých formátů. Mezi hlavní formáty patří PDF, JPEG a PNG.

Veškeré produkty, které Neodynamic nabízí jsou placené. U většiny jejich produktů se platí roční licence pro daný počet vývojářů, kteří s ní pracují. Licence nabízí v několika edicích a balíčcích. V případě ThermalLabel SDK, Neodynamic poskytuje plně funkční zkušební verzi pro nekomerční využití a testování. Zkušební verze obsahuje nesmazatelný text na vykreslovací ploše a při generování PDL z definovaného štítku přidává do horní části štítku logo TRIAL. Logo je náhodně generováno pro každý štítek a je složeno z mnoha menších čar, takže jeho ruční odstranění je téměř nemožné.

Použité funkce SDK jsou dále popsány v kapitolách 5 a 6.

# 4 ČÁROVÉ KÓDY

Čárové kódy jsou každodenní součástí našich životů. Lze je nalézt na téměř jakémkoliv produktu, slouží k identifikaci zboží a napomáhá automatizovanému sběru dat. Čárové kódy jsou často reprezentovány jako seskupení černých pruhů, které se snímají ručními skenery. První čárové kódy se v široké veřejnosti objevili až v roce 1973 (ačkoliv patentovány byly již v roce 1952) a od této doby jejich počet a využití stále roste. Jejich obliba roste převážně kvůli snadné dostupnosti a možnosti uchovat téměř jakákoliv data.

Ačkoliv je podoba čárových kódu nejznámější jako seskupení svislých pruhů, ve skutečnosti se jedná pouze o podskupinu jednorozměrných čárových kódů a existuje mnoho jiných tvarů a rozměrů (např. Code 49, PDF417, Aztec). Oficiálně je definováno přibližně 200 různých standardů čárových kódů (vizte tabulka 6). V České republice poskytuje registrace čárových kódů jediná firma – GS1 Czech Republic. Nadnárodní organizace GS1 International mimo jiné také hlídá standardy čárových kódů. Různé země mají vlastní lokalizovanou firmu dané organizace.

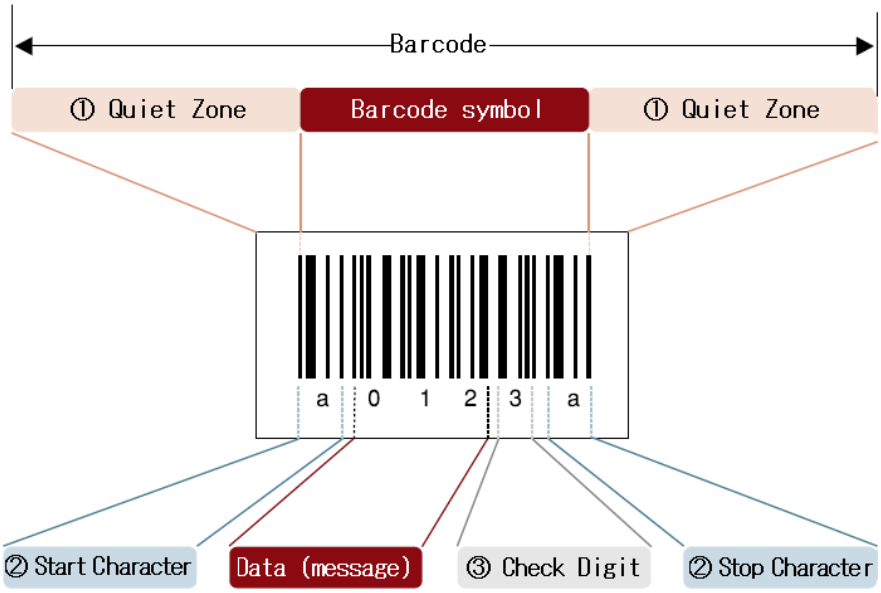
Vzhledem k dnešním technologiím a chytrým mobilním telefonům, snímání čárových kódů již není limitováno pouze příručními skenery v supermarketech. Drtivá většina mobilních telefonů umožňuje stažení aplikace, která využívá fotoaparát telefonu k přečtení většiny čárových kódů. Mezi nejčastější takové kódy patří QR Code, který se rozšířil právě jednoduchým čtením pomocí chytrých telefonů.

Kapitola se obecně zabírá strukturou a konstrukcí čárového kódu, jaké prvky obsahuje, jak uchovává data a jaký je rozdíl mezi vybranými typy kódů. Dále kapitola podrobněji popisuje vybrané typy čárových kódů, které se běžně používají v praxi.

Kapitola vychází ze zdrojů [36–46].

## 4.1 Konstrukce čárového kódu

Čárový kód se skládá z několika klíčových prvků. Mezi tyto prvky patří tichá zóna (quiet zone), zakódovaná data, počáteční znak, koncový znak a některé čárové kódy obsahují dodatečně ještě kontrolní součet (checksum) a opravu chyb (error correction). Obr. 25 ilustruje prvky na obecném jednorozměrném čárovém kódu.




**Obrázek 25 – Popis částí čárového kódu (zdroj: [38])**

### 4.1.1 Quiet Zone

Tichá zóna je prázdný okraj umístěný kolem čárového kódu. U jednorozměrných čárových kódů se tichá zóna nachází nalevo a napravo od kódovaných dat. Vícerozměrné čárové kódy (např. QR Code) mají tichou zónu kolem celého obvodu čárového kódu. Minimální šířka tiché zóny je typicky 2,5 mm. Pokud je šířka tiché zóny nedostatečná, čtecí zařízení může mít problémy se správným přečtením čárového kódu. Nejen, že bude často docházet k problémům se snímáním, ale také se mohou objevit chyby ve snímaných datech.

### 4.1.2 Start/Stop Character

Počáteční a koncový znak reprezentují začátek a konec dat v čárovém kódu. Ne vždy se jedná o znak. Začátek (start character) a konec (stop character) určuje samotná struktura čárového kódu. Start/Stop character se liší dle typu použitého čárového kódu. Například čárový kód typu Code 39 využívá jako počáteční a koncový znak hvězdičku (*). Jelikož hvězdička v tomto typu značí začátek a konec dat, nelze ji použít v kódovaných datech.

### 4.1.3 Checksum

Kontrolní součet pomáhá odhalit chybu při čtení čárového kódu. Po zakódování čárového kódu se provede nad výslednými daty funkce pro výpočet kontrolního součtu (např. modulo 10 u čárových kódů typu EAN). Kontrolní součet se přidá na čárový kód k zakódovaným datům.

Při skenování čárového kódu se znovu vypočítá kontrolní součet sejmutých dat a porovná se s kontrolním součtem načteným ze štítku. Výsledné součty musí být totožné. Kontrolní součet slouží pouze jako verifikace a pomáhá odhalit chybu skenování. Neumožňuje opravu samotných dat při poškození, jako to dělá oprava chyb.

Kontrolní součty se liší v závislosti na typu čárového kódu. Různé druhy čárových kódů využívají různé výpočty kontrolních součtů. Existují typy čárových kódů, které neobsahují žádný kontrolní součet nebo také čárové kódy, které naopak kontrolní součet vyžadují. Kontrolní součet je typický pro jednorozměrné čárové kódy. Tabulka 6 uvádí příklad typů čárových kódů s žádným, volitelným a povinným kontrolním součtem.

**Tabulka 6 – Kontrolní součty a čárové kódy (zdroj: [39])**

| Čárové kódy s žádným kontrolním součtem | Čárové kódy s volitelným kontrolním součtem | Čárové kódy s povinným kontrolním součtem |
|---|---|---|
| Add2 | Code 32 | Code 128 |
| Add5 | Code 39 | Code 93 |
| Codabar | Industry 2 of 5 | EAN-8 |
| DataLogic 2 of 5 | Inverted 2 of 5 | EAN-13 |
| Patch Code | BCD Matrix | EAN-128 |
| | Matrix 2 of 5 | UPC-A |

### 4.1.4 Error Correction

Záznam pro opravu chyb umožňuje pomocí algoritmu zrekonstruovat poškozenou část čárového kódu a zobrazit původní data. Čím více dat pro opravu chyb čárový kód obsahuje, tím větší poškození dokáže následně opravit. Následkem velkého záznamu pro opravu chyb je zmenšení kapacity pro samotná data čárového kódu. Oprava chyb je typická převážně pro vícerozměrné čárové kódy (např. QR Code). Jednorozměrné čárové kódy obsahují pouze kontrolní součty a poškozené části nelze softwarem opravit ani dopočítat.

Mezi čárové kódy podporující opravu chyb patří například:

- QR Code,
- Aztec,
- PDF417,
- Data Matrix,
- Australia Post Barcode.

## 4.2 Typy čárových kódů

Čárové kódy obsahují mnoho variací, podtypů nebo nových standardů předchozích typů. Obecně se čárové kódy rozdělují na dva typy dle stylu reprezentace dat. Čárové kódy se dělí na jednorozměrné a vícerozměrné.

### 4.2.1 1D čárové kódy

Jednorozměrné, 1D či lineární čárové kódy jsou označení pro čárové kódy první generace. Vyskytují se jako série svislých čar a mezer různých délek a šířek. Čáry následně tvoří lineární obrazec, který lze snímat čtečkami a skenery.

Jedná se o nejrozšířenější typy čárových kódů, které se ve velké míře používají v logistice, průmyslu a obchodu. Jednorozměrné čárové kódy jsou široké a zabírají více místa. Čím více dat obsahují, tím je čárový kód širší. Nevýhodou je jejich omezená kapacita. Vzhledem k použitému standardu mohou obsahovat maximálně 85 znaků. V některých případech mohou nastat i problémy se snímáním čárového kódu pomocí fotoaparátu v chytrém telefonu.

Mezi nejznámější jednorozměrné čárové kódy patří například Code 128, Code 39, EAN-8, EAN-13, UPC-A a Databar. Příklad jednorozměrných čárových kódů je ilustrován na obr. 26.

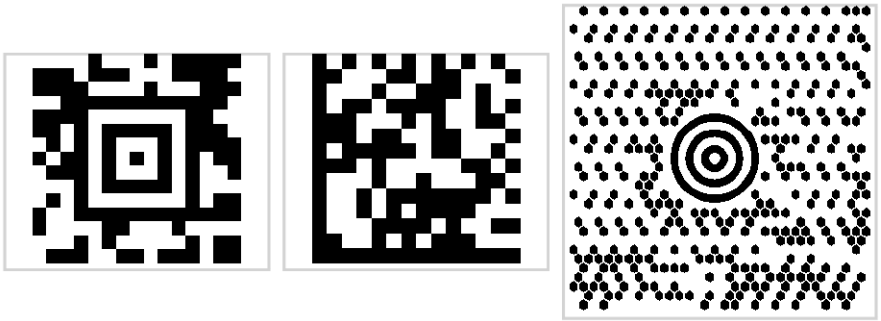

**Obrázek 26 – Jednorozměrné čárové kódy AustraliaPost (vlevo), ITF-14 (uprostřed) a PostNet (vpravo) (zdroj: [Autor])**

### 4.2.2 2D čárové kódy

Dvourozměrné, 2D nebo také maticové čárové kódy jsou novější a komplexnější čárové kódy. Na první pohled je lze rozeznat od jednorozměrných čárových kódů, díky absenci svislých čar. To je dáno plošnou, dvourozměrnou reprezentací dat. Data jsou kódována ve dvou rozměrech, a tudíž nemohou být vizualizována pomocí rovných čar. Naopak se jedná o změť čtverců, obdélníků, bodů, hexagonů či jiných geometrických obrazců. Některé dvourozměrné čárové kódy mají až absurdní reprezentaci a nevypadají jako nosiče informací (např. čárové kódy ShotCode, MaxiCode a DotCode).

Mezi největší výhodu patří velká kapacita nosných dat. Na rozdíl od jednorozměrných čárových kódů dokáží zakódovat až několik stovek znaků. Velký prostor pro data se může využít pro zakódování URL adresy, obrázků či informací o osobě (např. celý řidičský průkaz). Mezi další výhody patří velikost a snadné snímání. Dvourozměrné čárové kódy jsou navrhovány tak, aby je bylo možné snadno sejmout fotoaparátem chytrého telefonu. Navíc dosahují malých velikostí a dokáží využít celé plochy čárového štítku. Vzhledem k velkému prostoru pro informace obsahuje většina těchto čárových kódů i záznamy pro opravu dat.

Mezi nejčastější čárové kódy patří například QR Code, PDF417, Data Matrix, Aztec, MaxiCode. Příklad dvourozměrných čárových kódů je ilustrován na obr. 27.



**Obrázek 27 – Vícerozměrné čárové kódy Aztec (vlevo), DataMatrix (uprostřed) a MaxiCode (vpravo) (zdroj: [Autor])**

## 4.3 EAN

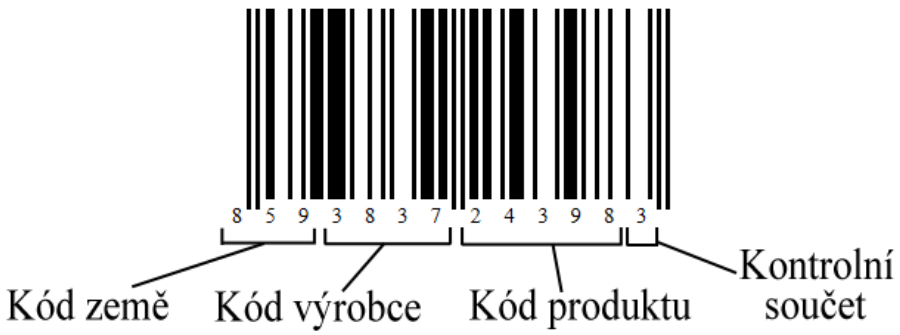
EAN je zkratkou pro čárový kód European Article Number, který byl v roce 2009 přejmenován na International Article Number. Jedná se o nejrozšířenější typ čárového kódu, který se používá v obchodní síti. EAN je celosvětově uznávaný čárový kód a využívají ho výrobci z více než 80[7] zemí světa. Může ho využívat jakýkoliv stát, který je v systému EAN UCC. Čárový kód EAN je kompatibilní s americkým a kanadským čárovým kódem UPC a japonským čárovým kódem JAN. Jedná se tak o univerzální čárový kód.

[7] https://www.makebarcode.com/specs/ean_cc.html

EAN lze nalézt na produktech v supermarketech a obchodech. Je využíván nejen pro snadné rozpoznání produktu, ale také pro marketingovou správu a analýzu trhu. Systém poskytuje například informace o počtu produktů ve skladu, v jaký čas byl produkt zakoupen nebo jaké produkty jsou nejvíce, či nejméně prodávané. Veškeré informace jsou dostupné v reálném čase. Tyto informace jsou velmi důležité pro rozhodování a vytváření marketingových strategií.

### 4.3.1 Složení dat

EAN je obecné označení pro čárový kód. Toto označení dále obsahuje podskupinu čárových kódů o různých délkách. Čárový kód EAN se může skládat z 8, 12, 13 nebo 14 číslic. Z počtu znaků následně vychází název čárového kódu. Čárový kód dlouhý 8 znaků nese označení EAN-8. Nejčastěji používaný čárový kód je EAN-13 (obsahující 13 znaků). EAN má omezení pouze na číslice od 0 do 9.




**Obrázek 28 – Příklad rozložení dat v čárovém kódu EAN-13 (zdroj: [41])**

Obr. 28 ilustruje rozložení dat v čárovém kódu EAN-13. Čárový kód EAN začíná vždy kódem země, ze které produkt pochází. Kód země je dvě až tři číslice dlouhý a určuje jej sdružení GS1. Česká republika má kód 859. Následuje čtyři až šest číslic, které určují výrobce. Každý výrobce má, podobně jako země, svůj vlastní číselný kód. Zbylé číslice čárového kódu, kromě poslední, již určují konkrétní produkt. Poslední číslice je kontrolní součet čárového kódu. Kontrolní součet se vypočítá pomocí funkce modulo 10 (vizte obr. 29). Násobič je konstanta, která se používá při výpočtu kontrolního součtu a liší se dle typu čárového kódu. Násobič je pevně stanovený.

| | |
|---|---|
| EAN číslo: | 8 59383724398 |
| Číslo: | 8 5 9 3 8 3 7 2 4 3 9 8 |
| Násobič: | 1 3 1 3 1 3 1 3 1 3 1 3 |
| Výsledek: | 8 15 9 9 8 9 7 6 4 9 9 24 Celkem = 117 |
| Kontrolní součet: | 3 (Výsledek 117, do dalšího násobku deseti zbývají 3) |

**Obrázek 29 – Výpočet kontrolního součtu (zdroj: [Autor])**

### 4.3.2 Složení čárového kódu

EAN nevyužívá počátečního a koncového znaku. Začátek a konec dat je určen samotnou strukturou čárového kódu. Ta obsahuje řídící čáry (guard bar) na začátku, ve středu a na konci čárového kódu. Ve skutečnosti obsahuje čárový kód EAN-13 pouze 12 čísel a třinácté číslo (první) je mimo hranice čárového kódu (vizte obr. 30)


**Obrázek 30 – Rozložení čar čárového kódu EAN-13 (zdroj: [41])**

Třináctimístné číslo je rozděleno na první, řídící číslo a dvě skupiny po šesti číslech. K zakódování čísla používá EAN sérii vzorů kódování, paritu a vzory čar. Vzory čar jsou pevně stanoveny (vizte obr. 31) a liší se dle kódovaného čísla a dále, zda se vyskytuje v levé či pravé části čárového kódu a dle parity. Parita je podstatná pro levou část čárového kódu, kde se rozlišuje sudá a lichá parita. V pravé části čárového kódu je parita vždy sudá.

| Character | Left odd parity | Left even parity | Right even parity |
|---|---|---|---|
| | Bar pattern | Bar pattern | Bar pattern |
| 0 | | | |
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |
| 6 | | | |
| 7 | | | |
| 8 | | | |
| 9 | | | |
| Left guard bar | | | |
| Center bar | | | |
| Right guard bar | | | |

**Obrázek 31 – Vzory čar pro čárový kód EAN (zdroj: [41])**

První číslice čárového kódu určuje vzor kódování. Různé vzory mají různé kombinace sudé a liché parity. Veškeré kombinace vzorů EAN kódování uvádí tabulka 7. Každý vzor začíná lichou paritou[8] a končí sudou paritou. Díky tomu nezáleží na směru skenování čárového kódu. Skenovací software dokáže identifikovat začátek a konec čárového kódu EAN dle pozice liché a sudé parity a přečíst data ve správném směru.

*L* značí lichou paritu a *S* značí sudou paritu.

**Tabulka 7 – Vzory kódování čárového kódu EAN-13 (zdroj: [41])**

| První číslice | První šestice čísel | Poslední šestice čísel |
|---|---|---|
| **0** | LLLLLL | SSSSSS |
| **1** | LLSLSS | SSSSSS |
| **2** | LLSSLS | SSSSSS |
| **3** | LLSSSL | SSSSSS |
| **4** | LSLLSS | SSSSSS |
| **5** | LSSLLS | SSSSSS |
| **6** | LSSSLL | SSSSSS |
| **7** | LSLSLS | SSSSSS |
| **8** | LSLSSL | SSSSSS |
| **9** | LSSLSL | SSSSSS |

Pokud je první číslo nula, všechny čísla první šestice se kódují dle vzoru LLLLLL. Stejný vzor je použitý i u čárového kódu UPC. Dalo by se říct, že UPC je stejný jako EAN-13 s nulou na prvním místě. Vzory kódování se v každém typu čárového kódu EAN liší. Tabulka 7 platí pouze pro čárové kódy EAN-13. Jiné typy (např. EAN-8) mají odlišné tabulky a pravidla kódování.

## 4.4 QR Code

QR Code je zkratkou pro Quick Response Code, volně přeloženo jako kód rychlé odezvy. Jedná se o dvourozměrný čárový kód čtvercového tvaru. Původně byl navržen pro automobilový průmysl v Japonsku. S postupem času se však rychle rozšířil do celého světa a stal se oblíbeným nejen pro automobilový průmysl. V současnosti se QR kód hojně využívá na webových stránkách, aplikacích a reklamních plochách.

[8] Parita odkazuje na sudý či lichý počet bitů s hodnotou jedna v rámci dané sady bitů.

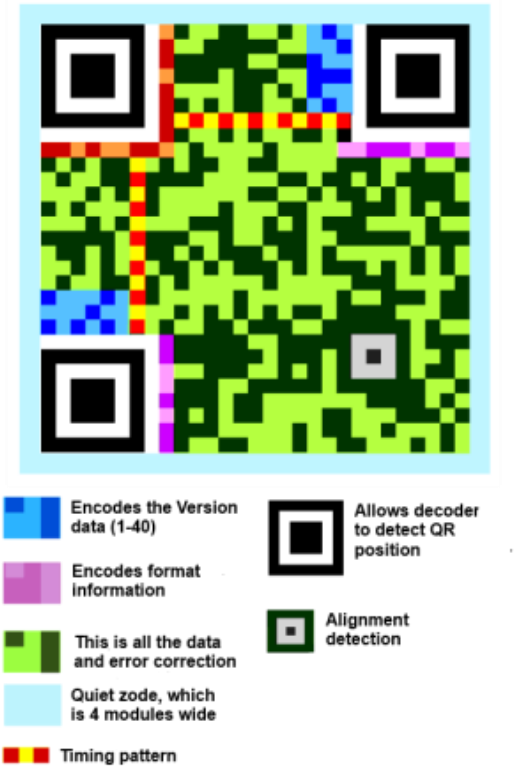
QR kód umožňuje snadné sdílení informací. Lze ho skenovat pomocí fotoaparátu v chytrém telefonu, a právě kvůli tomu se často kódují do QR kódu URL adresy webových stránek. Navíc umožní uživateli jednoduchý přístup na web bez nutnosti zdlouhavého opisování URL adresy (obzvlášť, když je URL adresa dlouhá).

### 4.4.1 Složení dat

QR kód kóduje data pomocí černých a bílých pixelových vzorů, které se nazývají moduly. Velikost musí být vždy čtvercová. Jedná se o dynamický čárový kód, což znamená, že může měnit svou velikost. Čím větší je QR kód, tím více dat dokáže pojmout. Velikost dat závisí na míře záznamu pro opravu chyb. S větší ochranou se snižuje velikost pro data. Největší QR kód s nejmenším záznamem pro opravu dat může obsahovat maximálně:

- 7089 číslic,
- 4296 alfanumerických znaků[9],
- 2953 binárních znaků.

QR kód se skládá z několika částí. Velké čtvercové obrazce slouží k detekování samotného kódu a pro lepší detekci. Zbylá data obsahují informace o verzi, kódování, velikosti a datech. Obr. 32 ilustruje rozložení dat na QR kódu.




**Obrázek 32 – Rozložení dat na QR kódu (zdroj: [43])**

[9] https://www.qr-code-generator.com/qr-code-marketing/qr-codes-basics/

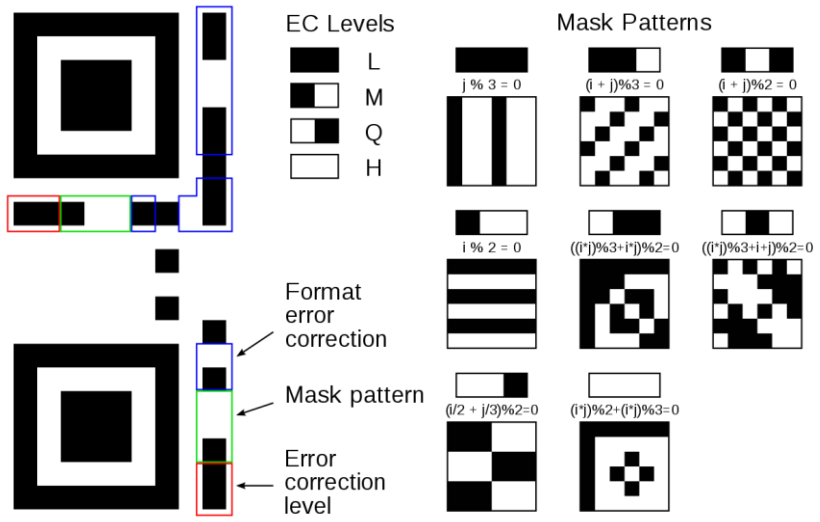
**Tabulka 8 – Jednotlivé části QR kódu (zdroj: [44])**

| Část obrazce | Název | Popis |
|---|---|---|
|  | Positioning markings | Polohovací značky pomáhají určit polohu QR kódu a správnou orientaci při skenování. |
|  | Alignment markings | Zarovnávací značky se objevují pouze u velkých QR kódů. Pomáhají dále s orientací čárového kódu, podobně jako polohovací značky. |
|  | Timing pattern | Dvoje série teček spojuje rohy polohovacích značek. Obě dvojce jsou totožné a říkají skeneru, jak velký je QR kód. |
|  | Version information | Informace o verzi poskytují číslo verze QR kódu. Momentálně existuje 40 různých verzí. Nejčastěji jsou používány verze 1-7. |
|  | Format information | Informace o formátu obsahují informace o míře záznamu pro opravu chyb, jejich formát a vzor masky (mask pattern). |
|  | Data and error correction | Samotná zakódovaná data QR kódu a záznam pro opravu chyb. |
|  | Quiet zone | Tichá zóna poskytuje prostor kolem čárového kódu, aby skener nebyl ničím rušen. Zvyšuje přesnost skenování. |

### 4.4.2 Dekódování QR kódu

Při dekódování QR kódu se nejdříve detekují tři polohovací značky v rozích čárového kódu. V případě velkého QR kódu se využijí pomocné zarovnávací značky. Tento proces umožní korektní detekování kódu a správnou orientaci, které je klíčové pro další zpracování.

Po detekování kódu a orientace lze začít určovat informace o QR kódu. Jako první se načte timing pattern. Ten skeneru řekne, jaká je velikost čárového kódu. Dále se načtou informace o QR kódu a jeho datech. V těchto informacích je zakódována míra a formát záznamu pro opravu chyb a vzor masky (mask pattern). Jedná se o konstanty, které lze určit pomocí obr. 33.




**Obrázek 33 – Pozice a vzory pro informace v QR kódu (zdroj: [45])**

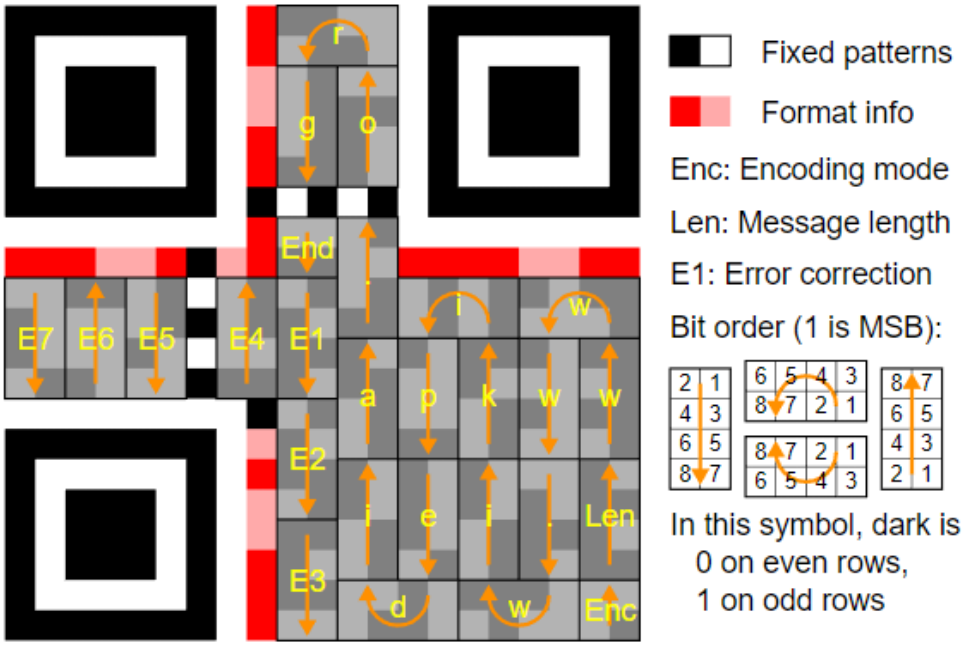
Na obr. 33 jsou vidět pozice jednotlivých informací a jaké hodnoty odpovídají konkrétním kombinacím. Hodnoty jsou v QR kódu umístěny duplicitně na více místech pro případ poškození části čárového kódu. QR kódy obsahují 4 úrovně záznamů pro opravu poškození (vizte tabulka 9). Vždy je třeba určit tuto úroveň, nelze vytvořit QR kód bez záznamu pro opravu poškození. K opravě se využívá algoritmus Reed-Solomon error correction[10].

**Tabulka 9 – Úrovně záznamů pro opravu chyb (zdroj: [46])**

| Úroveň | Rozsah poškození, které lze opravit |
|---|---|
| L (Low) | 7 % |
| M (Medium) | 15 % |
| Q (Quartile) | 25 % |
| H (High) | 30 % |

[10] https://en.wikipedia.org/wiki/Reed–Solomon_error_correction

Jako poslední se začínají číst a dekódovat samotná data. Na načtená data se aplikuje vzor masky (mask pattern), který se získal z předešlého kroku. Typy různých masek jsou uvedeny na obr. 33. Maska pouze invertuje černé a bílé moduly dle daného obrazce. Takto dekódovaná data lze začít číst.




**Obrázek 34 – Postup a směr čtení datových bloků a bitů (zdroj: [45])**

Obr. 34 ilustruje postup při čtení dekódovaných dat. Začíná se vpravo dole. Tato část (společně s ukončovací) je speciální, protože obsahuje pouze 4 bity. V prvních čtyřech bitech je uloženo kódování celých dat. Jedná se o indikátor módu, který je určen dle statických hodnot (vizte tabulka 10). K počátečnímu bloku čtyř bitů existuje párový ukončovací blok čtyř bitů, který je úplně prázdný a neobsahuje žádné bity. Jedná se o indikátor konce dat, aby skenovací software poznal, kde končí data a kde začínají další informace.

**Tabulka 10 – Tabulka indikátorů určující kódování dat QR kódu (zdroj: [45])**

| Indikátor | Popis |
|---|---|
| 0001 | Číselné kódování |
| 0010 | Alfanumerické kódování |
| 0100 | Byte kódování |
| 1000 | Kódování Kanji |
| 0011 | Structured append (použito pro rozdělení zprávy mezi QR kódy) |
| 0111 | Extended Channel Interpretation, alternativní kódování |
| 0101 a 1001 | FNC1 |
| 0000 | End of message (ukončovací indikátor) |

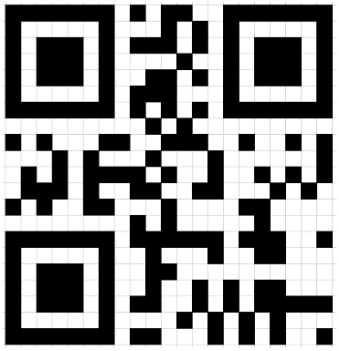
Zbylé bloky jsou tvořeny z 8 bitů, tedy jeden bajt informací. Tyto bloky se čtou stylem tzv. hada, jak je ilustrováno na obr. 34. Na stejném obrázku je také zobrazeno, jakým způsobem se čtou bity v jednotlivých blocích a směrech. Tento postup je nutné pevně dodržet. První osmice bitů obsahuje celkovou délku dat. Čtení dat ukončuje čtveřice prázdných bitů. Zbylá data slouží záznamu pro opravu chyb. Zde je vidět, že čím větší prostor zabírají data pro opravu chyb, tím méně místa zbývá pro zakódovaná data.

Při tvorbě QR kódu je třeba dát pozor na typ zvoleného kódování. Ačkoliv alfanumerické kódování se zdá vhodné pro text a čísla, toto kódování podporuje pouze čísla od 0 do 9, velká písmena a několik speciálních znaků. Neobsahuje žádná malá písmena.

Obzvláště velké QR kódy obsahují více zarovnávacích značek. Díky těmto značkám dochází k posunu a deformaci jednotlivých bloků. Z toho důvodu může být směr čtení dat změněn.

### 4.4.3 Příklad dekódování QR kódu

Pro příklad byl vygenerován QR kód o velikosti 21x21 modulů. Kapitola obsahuje všechny kroky pro ruční dekódování. Výchozí QR kód pro dekódování je zobrazen na obr. 35.



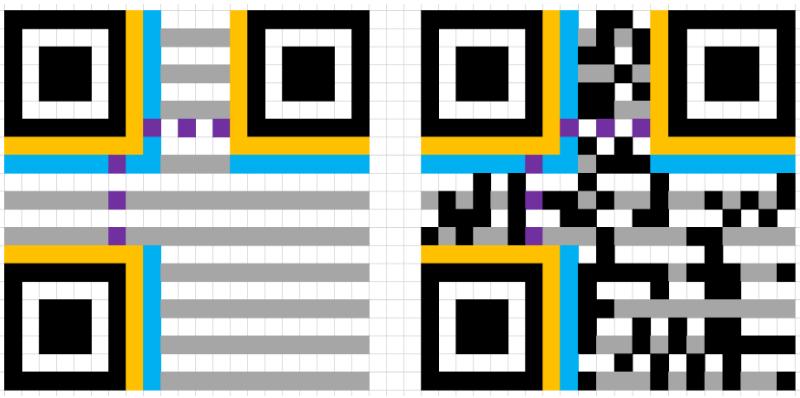
**Obrázek 35 – Příkladový QR kód pro dekódování (zdroj: [Autor])**

Prvním krokem dekódování je zjištění použité masky a úroveň záznamu pro opravu chyb. Na obr. 36 je fialovou barvou zobrazen timing, červeno-oranžovou barvou úroveň pro opravu chyb a zelenou barvou typ použité masky. Tmavá barva znamená černý modul.

**Obrázek 36 – QR kód se zvýrazněnými zónami (zdroj: [Autor])**

Pomocí obr. 33 je možné určit, že příkladový QR kód využívá masku číslo 1 a obsahuje úroveň M pro opravu chyb. Maska číslo 1 je jedna ze základních a udává, že v dalším kroku je nutné invertovat každý druhý řádek (i % 2 = 0). Úroveň M znamená, že QR kód obsahuje data pro opravení 15 % poškozených modulů.

V druhém kroku je nutné navrhnout cílovou masku a aplikovat ji na příkladový QR kód. Maska není aplikována na informační části čárového kódu, pouze na samotná data. Kvůli informacím o QR kódu (např. timing) může být maska posunuta, jak je možné vidět na obr. 37. V tomto kroku byla všechna informační data QR kódu označena modrou barvou a tiché zóny oranžovou barvou. Samotná maska je označena šedou barvou. Po kombinaci QR kódu a masky je nutné invertovat všechny moduly, které se nachází v šedé zóně (uvnitř masky).

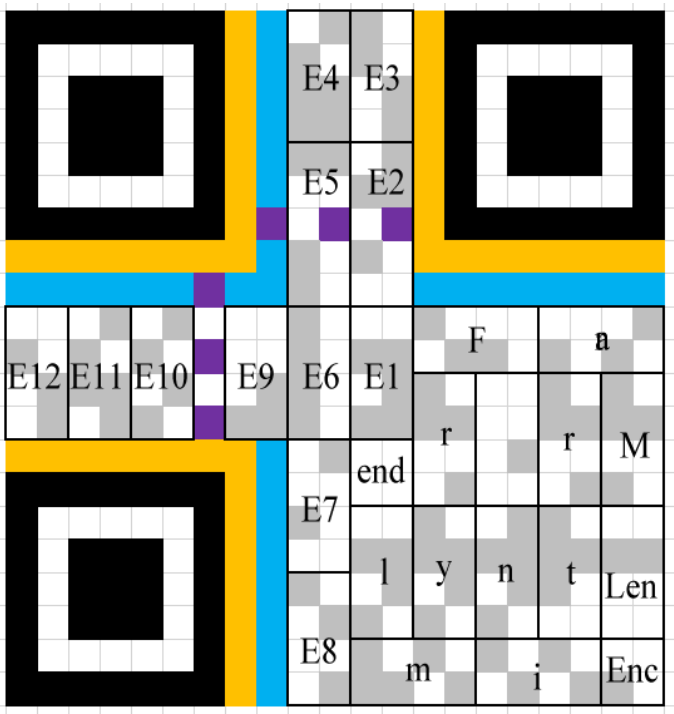


**Obrázek 37 – Maska (vlevo) a kombinace QR kódu a masky (vpravo) (zdroj: [Autor])**

Po dokončení invertování QR kódu pomocí masky se odhalí data, která jsou v něm zakódována. Odhalená data jsou zobrazena na obr. 38. Aby bylo možné data přečíst, je nutné určit kódování dat. Kódování se nachází v pravém dolním rohu QR kódu o rozsahu 4 modulů. K určení správného kódování pomůže tabulka 10. Směr čtení byl ilustrován na obr. 34.

**Obrázek 38 – QR kód s odstraněnou maskou (zdroj: [Autor])**

Blok kódování obsahuje binární kód 0100, což odpovídá byte kódování. Jelikož jeden bajt má osm bitů, jsou následující moduly rozděleny do bloků po osmi. Po určení kódování je možné začít dekódovat data. Prvním blokem je délka dat. V příkladu je nutné dodržovat standardní čtení dat zprava doleva. Každou hodnotu je nutné převést z binární soustavy do desítkové. Délka je 00001100, po převodu 12 znaků. První znak má binární podobu 01001101 a je převeden na číslo 77. Převedenou hodnotu je nutné vyhledat v tabulce ASCII pro získání znaku. Hodnota 77 odpovídá znaku M. Po dekódování všech hodnot získáme řetězec „Martin Fryml“. Jednotlivé bloky, včetně jejich dekódovaných hodnot, jsou ilustrovány na obr. 39. Zbylá data jsou použita pro obnovu, kdyby došlo k poškození QR kódu. Pro zpětnou kontrolu dekódovaného řetězce je možné naskenovat původní QR kód pomocí čtečky nebo skeneru.




**Obrázek 39 – Dekódovaný QR kód (zdroj: [Autor])**

Pro ilustraci obnovení poškozeného čárového kódu pomocí záznamu pro opravu chyb bylo z původního QR kódu úmyslně odříznuto několik modulů (vizte obr. 40). Čárový kód lze i přes jeho poškození přečíst.

**Obrázek 40 – Poškozený QR kód (zdroj: [Autor])**

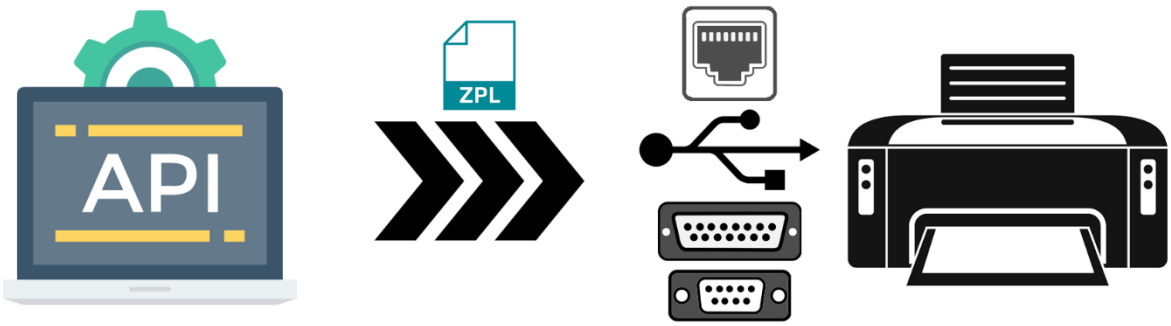
# 5 KOMUNIKAČNÍ API

Komunikační API (dále jen API) je softwarové rozhraní, které umožňuje komunikovat s tiskárnami štítků. Vytváří aktivní spojení s cílovou tiskárnou a odesílá jí poskytnutá data. Spojení lze otevřít s tiskárnou připojenou na rozhraní USB, TCP/IP, LPT nebo COM. Kromě odesílání dat do tiskárny umožňuje API manipulaci s pamětí tiskárny. Manipulací s pamětí se rozumí čtení obsahu paměti, získání velikosti volné paměti a nahrání dat do paměti.

Sekundární funckionalitou API je vázání dat. Do API je možné odeslat šablonu štítku s dynamickými hodnotami a zdroj dat (např. soubor). API dokáže navázat data z datového zdroje do šablony štítku a připravit PDL pro všechny hodnoty ve zdroji. Následně odešle svázaná data do tiskárny na tisk. API taktéž obsahuje podporu pro náhled ZPL II ze šablony štítku. Obr. 41 ilustruje komunikační API.

Pro realizaci komunikačního API byl použit programovací jazyk C#, knihovna Neodynamic ThermalLabel (vizte kapitola 3.3) a systémové funkce operačního systému Windows. Celé API využívá ZPL a ZPL II jako jazyk pro popis stránky. S výjimkou USB připojení, které je určeno pouze pro tiskárny typu Zebra, je API univerzální a lze ho použít na téměř jakoukoliv tiskárnu štítků, která podporuje PDL ZPL či ZPL II. Komunikace byla testována na průmyslové tiskárně Zebra 110Xi4.

Kapitola využívá zdrojů [11], [35] a [47–51].




**Obrázek 41 – Komunikační API (zdroj: [Autor])**

## 5.1 Analýza a návrh

Před samotnou realizací komunikačního API je nutné provést analýzu a návrh rozhraní. Prvotní návrh vychází z obecné myšlenky komunikace s tiskárnami štítků. Následně je možné rozšířit návrh o různá komunikační rozhraní a další funkcionality.

Na začátku analýzy byly vytvořeny funkční a nefunkční požadavky. Funkční požadavky definují, co by API mělo dělat a jak se chovat. Nefunkční požadavky naopak určují omezení API a použitou technologii. Celý model funkčních a nefunkčních požadavků vypracovaný v programu Enterprise Architect se nachází v příloze A.

### 5.1.1 Funkční požadavky

V tabulce 11 jsou uvedeny základní funkční požadavky pro komunikační API. Převážně se jedná o podporu komunikačních rozhraní a práce s daty. Požadavky jsou rozděleny do logických celků (balíčků).

**Tabulka 11 – Funkční požadavky komunikačního API (zdroj: [Autor])**

| **Komunikace** | |
|---|---|
| R1000 | API bude poskytovat komunikaci pomocí sítě. |
| R1001 | API bude poskytovat komunikaci pomocí USB. |
| R1002 | API bude poskytovat komunikaci pomocí sériového portu. |
| R1003 | API bude poskytovat komunikaci pomocí LPT. |
| R1004 | API bude umožňovat připojení do databáze. |
| **Data** | |
| R1100 | API bude umožňovat vypsání paměti tiskárny. |
| R1101 | API bude umožňovat nahrání fontu do paměti tiskárny. |
| R1102 | API bude umožňovat získání volného místa v paměti tiskárny. |
| R1103 | API bude umožňovat vázání dat. |
| R1104 | API bude umožňovat parsování[11] XML souboru. |
| R1105 | API bude umožňovat parsování JSON souboru. |

[11] Parsování je procházení, dělení a transformace dat pro další použití.

### 5.1.2 Nefunkční požadavky

Nefunkční požadavky jsou podobně jako funkční požadavky rozděleny do logických balíčků. Blíže určují omezení komunikace a specifikují použití technologií. Nefunkční požadavky jsou uvedeny v tabulce 12.

**Tabulka 12 – Nefunkční požadavky komunikačního API (zdroj: [Autor])**

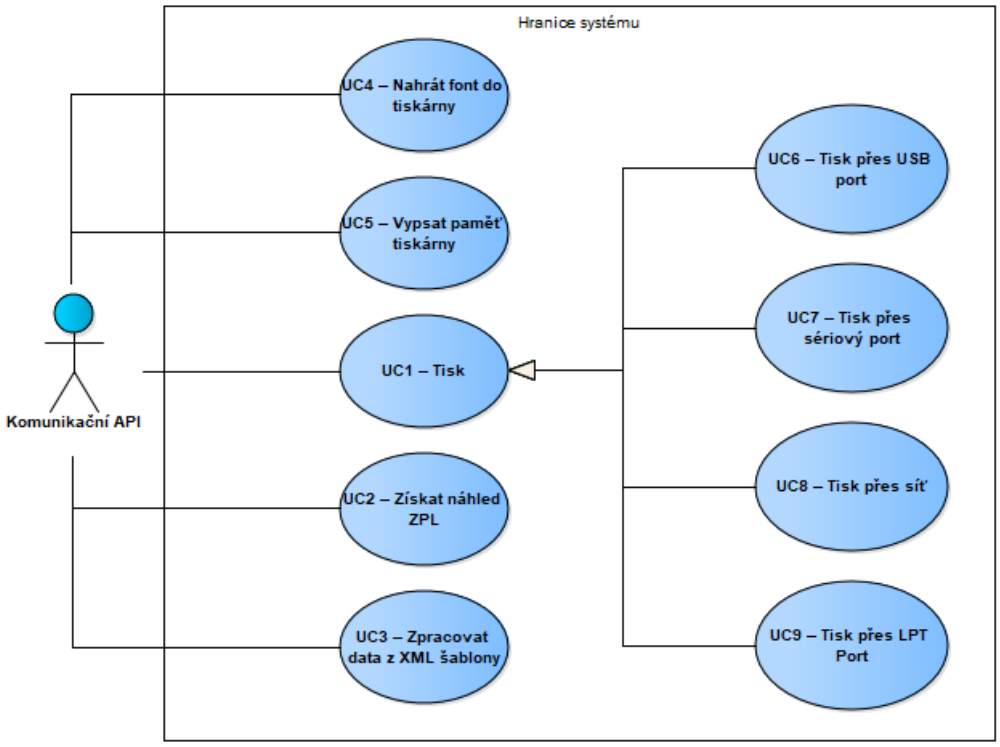
| **Komunikace** | |
|---|---|
| R2000 | API bude podporovat tiskárny firmy Zebra. |
| R2001 | API bude podporovat připojení do databáze Oracle. |
| **Zabezpečení** | |
| R2100 | Síťová komunikace API bude probíhat na TCP/IP protokolu. |

### 5.1.3 Aktéři

Aktéři reprezentují entitu, která využívá daný systém. Jelikož se nejedná o informační systém, ale o komunikační rozhraní, byl při analýze nalezen pouze jeden aktér. Komunikaci bude zprostředkovávat samotné API a použití více aktérů není potřeba. Pro případy užití komunikačního API byl vytvořen jediný aktér *Komunikační API*.

### 5.1.4 Případy užití

Případy užití popisují chování systému z pohledu aktérů. Každý obsahuje unikátní identifikátor pro lepší orientaci a scénář. Scénář popisuje činnost (případně sled činností), které případ užití vykonává, společně s omezujícími podmínkami. Obr. 42 zobrazuje případy užití komunikačního API. Mezi hlavní schopnosti komunikačního API patří tisk štítku, zpracování dat z Editoru štítků (vizte kapitola 6), získání náhledu ZPL II kódu a přístup do paměti tiskárny.




**Obrázek 42 – Případy užití komunikačního API (zdroj: [Autor])**

Následuje popis vybraných případů užití a jejich scénářů (vizte tabulky 13–15). Kompletní model případů užití a scénářů je uveden v příloze A.

**Tabulka 13 – Případ užití UC1 – Tisk (zdroj: [Autor])**

| |
|---|
| **Případ užití:** UC1 – Tisk |
| **ID:** UC1 |
| **Stručný popis:** Tisk ZPL kódu pomocí zvolené cílové tiskárny. |
| **Aktéři:** Komunikační API |
| **Podmínky:** Žádné |
| **Hlavní scénář:**<br>1. Uživatel odešle data do API.<br>2. API verifikuje vstupní data.<br>3. API naváže spojení s cílovou tiskárnou.<br>4. API odešle data do tiskárny.<br>5. Tiskárna vrátí API výsledek tisku.<br>6. API zpracuje výsledek a vrátí podrobnosti tisku. |
| **Výjimky:**<br>2a. Data neprošla vstupní validací.<br>3a. API se nespojilo s cílovou tiskárnou. |

**Tabulka 14 – Případ užití UC2 – Získat náhled ZPL II (zdroj: [Autor])**

| |
|---|
| **Případ užití:** UC2 – Získat náhled ZPL II |
| **ID:** UC2 |
| **Stručný popis:** Získání náhledu štítku ve formátu ZPL II jazyka z existující šablony Editoru štítků a datového zdroje. |
| **Aktéři:** Komunikační API |
| **Podmínky:** Vstupní data musí obsahovat XML šablonu štítku. |
| **Hlavní scénář:**<br>1. Uživatel odešle data do API.<br>2. API verifikuje vstupní data.<br>3. API provede vázání dat (pokud je potřeba) a odešle data do SDK.<br>4. SDK vygeneruje ZPL II kód.<br>5. API vrátí náhled štítku v jazyce ZPL II. |
| **Výjimky:**<br>2a. Data neprošla vstupní validací. |

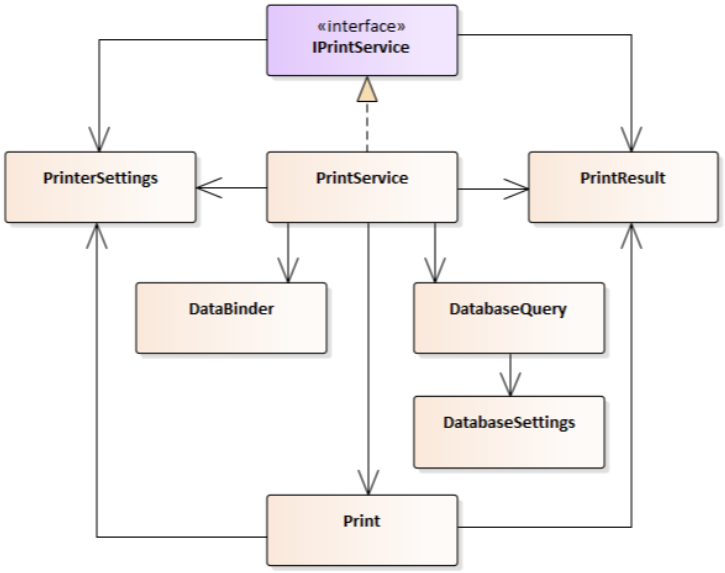
**Tabulka 15 – Případ užití UC5 – Vypsat paměť tiskárny (zdroj: [Autor])**

| |
|---|
| **Případ užití:** UC5 – Vypsat paměť tiskárny |
| **ID:** UC5 |
| **Stručný popis:** Získání celého obsahu zvolené paměti z cílové tiskárny. |
| **Aktéři:** Komunikační API |
| **Podmínky:** Žádné |
| **Hlavní scénář:**<br>1. Uživatel odešle data do API.<br>2. API verifikuje vstupní data a naváže spojení s cílovou tiskárnou.<br>3. API odešle do tiskárny příkaz pro získání obsahu paměti.<br>4. Tiskárna vrátí API obsah paměti.<br>5. API zpracuje obdržená data a vrátí je uživateli. |
| **Výjimky:**<br>2a. Data neprošla vstupní validací.<br>3a. API se nespojilo s cílovou tiskárnou. |

### 5.1.5 Analytické třídy

Analytické třídy umožňují abstrakci nad konkrétním problémem. Komunikační API bude obsahovat jedno programátorské rozhraní (*IPrintService*) pro přístup k vnitřním třídám a metodám. Informace o připojení k tiskárně budou do API předávána pomocí třídy *PrinterSettings*.

Každá metoda, která bude komunikovat s tiskárnou, bude vracet svoje výsledky o připojení pomocí třídy *PrintResult*. Zbylé třídy zajistí vázání dat, připojení k databázi a samotný tisk, obsažený v třídě *Print*. Výsledná implementace se může lišit. Analytické třídy mají reprezentovat myšlenku dané problematiky. Návrh analytických tříd ilustruje obr. 43.




**Obrázek 43 – Částečné schéma tříd komunikačního API (zdroj: [Autor])**

Následující kapitoly se věnují implementaci API a některé třídy a metody se liší od původní analýzy. Model a schéma implementace se nachází v příloze A.

## 5.2 Typy komunikace

Různorodost komunikačních rozhraní u jednotlivých modelů tiskáren štítků nutí uživatele a programátory využívat více knihoven pro komunikaci nebo hledat alternativní řešení. API se snaží poskytnout univerzální rozhraní pro komunikaci nezávisle na modelu tiskárny. Různé typy komunikací vyžadují různé metody implementace. Pro rozlišení jednotlivých typů komunikace využívá API výčtu *CommunicationType*. API podporuje komunikaci s tiskárnou pomocí rozhraní USB, LPT, COM a protokolu TCP/IP. Následuje ukázka výčtu *CommunicationType*.

```
public enum CommunicationType {
    USB = 0,
    Serial = 1,
    Parallel = 2,
    Network = 3,
    Other = 4
}
```

### 5.2.1 Síť

Prvním implementovaným typem komunikace je komunikace přes síť. Komunikační rozhraní využívá protokolu TCP/IP. Cílová tiskárna musí být připojená pomocí síťového kabelu nebo musí obsahovat modul pro bezdrátový přenos. Při prvním spuštění je nutné nastavit IP adresu tiskárny.

Síťový tisk je implementován ve třídě *NetworkPrint* ve statické metodě *Print*. Pro otevření komunikačního kanálu vyžaduje metoda pouze IP adresu tiskárny a port. Pro PDL a jejich datové toky je rezervovaný TCP port 9100. Ve výchozím nastavení je na většině tiskáren nastaven právě komunikační port 9100. Port lze v konfiguraci tiskárny přenastavit.

Informace o cílové tiskárně se předávají do metody pomocí struktury *PrinterSettings*. Pro otevření komunikačního kanálu se využívá TCP Socket a IP End Point. Následuje ukázka připojení k cílové tiskárně pomocí TCP/IP.

```
IPEndPoint remoteEP = new IPEndPoint(printerSettings.NetworkIPAddress,
                                     printerSettings.NetworkPort);
Socket sender = new Socket(printerSettings.NetworkIPAddress.AddressFamily,
                           SocketType.Stream, ProtocolType.Tcp);
sender.Connect(remoteEP);
```

ZPL II kód, který se má odeslat do tiskárny je do metody předán ve formě řetězce (string). Řetězec se před odesláním musí transformovat na pole bajtů, které se odešle otevřeným TCP spojením. Následuje ukázka odeslání dat otevřeným socketem.

```
byte[] dataToSend = Encoding.UTF8.GetBytes(printerCommands);
sender.Send(dataToSend);
```

V některých případech (například čtení paměti), chceme po odeslání příkazu číst data, které nám vrátí tiskárna. To lze provést aktivním čekáním a naslouchání na otevřeném kanálu. Přijatá data se ukládají do pole bajtů (buffer) a pro získání celé zprávy je nutná transformace na řetězec. Následující kód umožňuje číst otevřený socket[12] do té doby, dokud druhá strana odesílá nějaká data a zároveň data transformuje do čitelného řetězce.

[12] Socket je koncový bod připojení, umožňující obousměrnou komunikaci.

```
byte[] dataBuffer = new byte[1024];
string message = string.Empty;
int bytesReceieved;
do {
        bytesReceieved = sender.Receive(dataBuffer);
        message += Encoding.UTF8.GetString(dataBuffer, 0, bytesReceieved);
} while (sender.Available > 0 && bytesReceieved > 0);
```

Po ukončení komunikace je nutné uzavřít aktivní spojení pomocí následujícího kódu.

```
sender.Shutdown(SocketShutdown.Both);
sender.Close();
```

Nevýhodou síťové komunikace je ochrana přenášených dat. Odeslaná data nemusí být šifrována, a tudíž nijak chráněna před zásahem zvenčí. Některé tiskárny štítků nepodporují zabezpečenou komunikaci.

### 5.2.2 USB

Komunikace pomocí USB je jednoduchá pro uživatele či programátora, ale náročná na implementaci. USB tisk funguje na principu prohledání připojených zařízení pomocí rozhraní USB a vybrání správného výstupního portu na základě jeho názvu. Vstupem do metody *Print* ve třídě *UsbPrint* jsou pouze data a název cílové tiskárny.

Jazyk C# se obtížně dostává k datům jako jsou USB porty. Z toho důvodu byla použita otevřená knihovna WinUSBNet[13] a systémová knihovna Kernel32.dll. Knihovna WinUSBNet poskytuje seznam všech připojených zařízení. Zařízení lze filtrovat a omezit je pouze na USB zařízení pomocí GUID identifikátoru, ve Windows dokumentaci označován jako *GUID_DEVINTERFACE_USB_DEVICE*. GUID má následující podobu.

```
{A5DCBF10-6530-11D2-901F-00C04FB951ED}
```

Po získání všech USB zařízení se jednotlivá zařízení procházejí a kontroluje se shoda jejich názvu s názvem cílové tiskárny. Cílem tohoto procesu je získání korektního identifikátoru USB portu, pro následné otevření komunikace.

[13] https://github.com/madwizard-thomas/winusbnet

Knihovna WinUSBNet poskytuje informace o zařízení ve struktuře *USBDeviceInfo*. Mimo jiné také obsahuje cestu k zařízení, VID a PID[14]. Následuje příklad cesty (device path) k tiskárně Zebra, která se používá pro připojení k tiskárně štítků.

```
\\?\usb#vid_0a5f&pid_00ab#jjk011718#{a5dcbf10-6530-11d2-901f-00c04fb951ed}
```

Z cesty USB zařízení lze otevřít konkrétní adresář v registrech[15] a získat další informace o zařízení, například sériové číslo, jméno rozhraní nebo číslo a popis daného portu, ve kterém je zařízení zapojeno. Pomocí jména tiskárny a čísla portu, ve kterém je tiskárna zapojena, kontroluje metoda *IsMatchingPort*, zda se opravdu jedná o správnou tiskárnu štítků.

Aby se redukoval počet kontrolovaných USB zařízení a urychlil se tak celý proces, vyhledávání korektního portu je omezeno pouze na tiskárny firmy Zebra. Omezení je dáno podle VID. VID je unikátní označení prodejce a každý počítačový hardware ho musí obsahovat. Identifikátory prodejců lze veřejně dohledat[16]. Označení VID pro tiskárny Zebra je `0A5F`.

Po úspěšné validaci portu a získání správné cesty k tiskárně štítků je možné otevřít komunikační spojení. Ke komunikaci byla použita funkce *CreateFile*, která je definována v systémové knihovně Kernel32.dll a slouží k otevírání souborů, datových toků a zápisů na disk. Jelikož se jedná o funkci napsanou v jiné knihovně a jazyce C++, bylo nutné použít klíčové slovo `extern` a znovu si deklarovat metodu dle C# syntaxe. Systémová syntaxe a extern deklarace jsou ilustrovány na obr. 44.

```
HANDLE CreateFileA(
        LPCSTR                  lpFileName,
        DWORD                   dwDesiredAccess,
        DWORD                   dwShareMode,
        LPSECURITY_ATTRIBUTES   lpSecurityAttributes,
        DWORD                   dwCreationDisposition,
        DWORD                   dwFlagsAndAttributes,
        HANDLE                  hTemplateFile
);
[DllImport(„kernel32.dll“, SetLastError = true, ChatSet = ChatSet.Auto)]
private static extern SafeFileHandle CreateFile (string lpFileName, EFileAccess dwDesiredAccess,
                                                 EFileShare dwShareMode, IntPtr lpSecurityAttributes,
                                                 ECreationDisposition dwCreationDisposition,
                                                 EFileAttributes dwFlagsAndAttributes,
                                                 IntPtr hTemplateFile);
```

**Obrázek 44 – Systémová deklarace funkce (nahoře) a C# extern definice (dole) (zdroj: [Autor])**

[14] PID je číselné označení procesu.
[15] Registry slouží pro ukládání klíčů a hodnot v OS Windows.
[16] https://www.the-sz.com/products/usbid/

Funkci se do parametrů zašle dříve získaná USB cesta cílové tiskárny společně s hexadecimálními hodnotami práv a přístupů. Jsou otevírány dva komunikační kanály. Jeden pro zápis (writePort) a druhý pro čtení zpětné vazby (readPort). Funkce vrací ukazatel *SafeFileHandle* na otevřené spojení datového toku (data stream).

```
SafeFileHandle writePort = CreateFile(
                                printerPort,
                                EFileAccess.GENERIC_WRITE,
                                EFileShare.FILE_SHARE_WRITE,
                                IntPtr.Zero, ECreationDisposition.OPEN_ALWAYS,
                                EFileAttributes.FILE_FLAG_SEQUENTIAL_SCAN |
                                EFileAttributes.FILE_ATTRIBUTE_NORMAL,
                                IntPtr.Zero);
```

Pro přehlednost se místo hexadecimálních hodnot používají vlastní výčty s definovanými hodnotami. Například EFileAccess.GENERIC_WRITE obsahuje hodnotu 0x40000000. Pro práci s otevřeným datovým tokem se využívá třída *StreamWriter*.

```
using (var sw = new StreamWriter(new FileStream(writePort, FileAccess.Write)))
{
      sw.WriteLine(printerCommands);
      sw.Flush();
}
```

Při implementaci čekání na odpověď USB portu (metoda *ReadUsbPort*) bylo nutné řešit omezené čekání. V případě, že tiskárna nevrátí žádnou opověď, ale metoda na ni i přesto čeká, nedochází k žádnému automatickému vypršení časového limitu, jako je tomu u TCP/IP spojení. Metoda *ReadPrinter* řeší a hlídá vypršení časového limitu pomocí úloh, vláken a tokenu pro zrušení (*CancellationTokenSource*).

Otevřené ukazatele je nutné po dokončení přenosu korektně uzavírat pomocí metod *Close* a *Dispose*. Alternativou přenosu dat je využití systémové knihovny winspool.drv.

### 5.2.3 Paralelní port (LPT)

Paralelní port, zkráceně LPT je zastaralá technologie přenosu dat. Byla nahrazena novějším a rychlejším rozhraním USB. I přes jeho nahrazení se paralelní port stále využívá u starších typů tiskáren štítků a na výrobních linkách. Paralelní port obsahuje 25 pinů. Jednotlivé porty nesou označení LPT#, kde # označuje číslo portu. Na tiskárnách štítků se vyskytuje většinou pouze jeden paralelní port, ale existují výjimky a počet portů může být vyšší.

Tisk pomocí rozhraní LPT je implementován ve třídě *LPTPrint*. Metoda tisku vyžaduje pouze data určená k odeslání do tiskárny a název paralelního portu. Paralelní port musí vždy začínat písmeny LPT, jinak neprojde vstupní validací. Validní název portu je například `LPT1`.

Metoda LPT tisku využívá stejného principu připojení jako metoda USB tisku. Využívá systémové knihovny Kernel32.dll a systémovou funkci *CreateFile*. Připojení pomocí paralelního portu přináší výhodu okamžitého navázání spojení. Není nutné určovat cestu k zařízení a korektní název, jako je tomu u USB tisku. Malý počet paralelního portu umožňuje snadné určení čísla port a uživatel zadává do metody přímo přesný název.

Po otevření komunikačního kanálu a získání ukazatele se vytvoří instance třídy *FileStream* s právy zápisu. Před zápisem dat se musí data transformovat z řetězce na pole bajtů, jak je ilustrováno níže.

```
FileStream lpt = new FileStream(lptPointer, FileAccess.ReadWrite);
var buffer = Encoding.ASCII.GetBytes(printerCommands);
lpt.Write(buffer, 0, buffer.Length);
```

Při zápisu do instance *FileStream* se odesílá pole bajtů dat, posun dat a celková délka dat. Po dokončení přenosu je nutné uzavřít komunikační kanál pomocí metody *Dispose*.

```
lpt.Dispose();
lptPointer.Dispose();
```

### 5.2.4 Sériový port

Sériový port, COM, či také standard RS-232, je podobně jako paralelní port zastaralé komunikační rozhraní. Byl taktéž nahrazen novějším rozhraním USB. Konektor obsahuje 9 pinů a využívá se jen v krajních případech, když nejsou předešlé typy komunikace dostupné. U tiskáren štítků má malé využití.

Tisk pomocí sériového portu je velkým kontrastem pro USB tisk. Sériový port vyžaduje rozsáhlá a uživatelsky nepříjemná nastavení. Mezi základní charakteristické údaje pro připojení patří název sériového portu, rychlost přenosu baud, počet bitů, parita a stop bity. C# poskytuje třídu *SerialPort* a výčty pro navázání spojení pomocí sériového portu. Následuje ukázka kódu vytvoření spojení s výchozím nastavením.

```
SerialPort serialPort =
        new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One) {
                Handshake = Handshake.XOnXOff
};
serialPort.Open();
```

Správné nastavení parametrů je klíčové pro navázání spojení se sériovým portem. Rychlost se udává v jednotách Baud a reprezentuje počet změn linky za vteřinu. Obvykle se volí hodnota mezi 110 až 115200. Parita slouží jako detektor chyb. Počet bitů určuje, kolik bitů je v každém znaku. Standardně se používá 7 pro ASCII a 8 pro jakákoliv data, protože 8 se rovná velikosti bajtu. Posledním významným nastavení jsou stop bity. Stop bity se odesílají na konci každého znaku a slouží pro detekci konce a synchronizaci přenosu. Standardně se využívá jeden stop bit, ale pomalejší zařízení mohou využívat jeden a půl či dva stop bity.

Po úspěšném otevření komunikačního kanálu se transformují vstupní data na pole bajtů a odešlou se sériovým portem. Parametry pro zápis jsou pole bajtů, posun dat a celková délka dat.

```
byte[] dataToSend = Encoding.UTF8.GetBytes(printerCommands);
serialPort.Write(dataToSend, 0, dataToSend.Length);
```

## 5.3 Paměť tiskárny

Vnitřní paměť je důležitá u jakéhokoliv zařízení, které zpracovává nějaká data. Tiskárny štítku nejsou výjimkou. Standardně se v paměti tiskárny nachází firmware, který tiskárně umožňuje tisk a zpracování vstupních dat. Dále firmware umožňuje komunikaci s externím zařízením. Firmware se obvykle nachází na flash paměti, která udržuje data i po odpojení zdroje napětí. Některé tiskárny štítků neposkytují přístup k firmwaru ani k flash paměti, kde se firmware nachází. Důvodem může být například bezpečnost zařízení. Mimo firmwaru potřebuje tiskárna i paměť RAM, kde bude uchovávat data během jejich zpracování. S pamětí RAM lze většinou manipulovat a ukládat do ní soubory pro rychlejší a snadnější tisk. RAM je však závislá na napájecím zdroji, a tak není vhodná pro trvalé ukládání dat. K tomu slouží rozšiřující sloty pro různé paměťové karty, které dále navyšují externí úložnou kapacitu.

### 5.3.1 Typy paměti tiskárny

Tiskárny Zebra obsahují několik typů pamětí. Paměťové jednotky v tiskárně si lze představit jako diskové oddíly. Každá paměťová jednotka je označená identifikačním písmenem. Identifikační písmeno (či zkráceně ID), umožňuje tvorbu absolutních cest a snadnější přístup do paměti z externích zařízení. ID lze za běhu tiskárny změnit pomocí ZPL II příkazu `^CM`. Tabulka 16 poskytuje typy pamětí, výchozí označení a využití v tiskárnách Zebra. Je nutné poznamenat, že ne všechny varianty paměti jsou najednou dostupné v jedné tiskárně. Například série Xi4 tiskáren Zebra nepodporuje paměti PCMCIA a Compact Flash.

**Tabulka 16 – Typy pamětí v tiskárnách Zebra (zdroj: [11])**

| Typ paměti | Označení | Využití |
|---|---|---|
| Compact Flash | A: | Volitelná část paměti. Jedná se o dodatečnou datovou kartu, která slouží pro navýšení úložné kapacity tiskárny. |
| Socket Flash / Battery backed-up RAM / PCMCIA | B: | Další volitelná část paměti, která využívá větších paměťových karet či RAM čipů s baterií, aby dokázaly udržet data i po vypnutí tiskárny. Navyšuje úložnou kapacitu tiskárny. |
| EPROM / Flash | E: | Obsahuje firmware tiskárny. Jedná se o typ paměti, která udržuje data i po odpojení zdroje napětí. Může být použita pro uložení fontů, obrázků a jiných dat. |
| DRAM | R: | Dynamická paměť určená pro dočasná data a data použitá během tisku. Počet štítků, které lze najednou tisknout je omezeno pamětí DRAM. Po odpojení zdroje napájení ztrácí veškerá data. |

### 5.3.1 Manipulace s pamětí

Pro rychlejší tisk a opakované použití zdrojů je nutná manipulace s pamětí. Manipulací s pamětí se rozumí čtení a zápis dat z a do paměti tiskárny. Velikost přenášených dat navyšují převážně obrázky a fonty. Z toho důvodu je tendence minimalizovat počet odesílaných obrázků a fontů do tiskového procesu. Řešením je nahrání obrázku, respektive fontu do paměti tiskárny namísto opakovaného odesílání ve formátu ZPL II.

Během nahrávání se určí identifikátor datového objektu. Na obrázek či font v paměti tiskárny se musí ZPL II odkazovat pomocí absolutní cesty a identifikátoru. Následuje příklad absolutní cesty pro font v paměti tiskárny.

```
E:ARI001.FNT
```

Pro minimalizování typových chyb a lepší čitelnost byl vytvořen výčet typů paměti *PrinterMemoryType* s výchozím označením umístění. Díky zmíněnému výčtu lze snadno určit cílové umístění souboru, vizte obr. 45.

```
public enum PrinterMemoryType
{
        CompactFlash = 'A',
        MemoryCard = 'B',
        Flash = 'E',
        DRAM = 'R'
}
```

**Obrázek 45 – Výčet typů pamětí tiskárny (zdroj: [Autor])**

Pro nahrání fontu do paměti tiskárny využívá API příkaz ^DU. Příkaz vyžaduje absolutní cestu do paměti tiskárny, včetně názvu fontu, počet bajtů, které bude zabírat v paměti a samotný font ve formě hexadecimálního ASCII řetězce. Název fontu (identifikátoru) je v paměti tiskárny omezen na 8 znaků. Omezení je dáno jazykem ZPL II. API umožňuje dynamické generování identifikátoru na základě názvu fontu a vlastností písma. Následuje příklad sestavení ZPL II příkazu pro nahrání fontu Arial.

```
byte[] fontByteArray = File.ReadAllBytes(@"C:\Windows\Fonts\Arial.fnt");
var fontData = BitConverter.ToString(fontByteArray);
fontData = fontData.Replace("-", string.Empty);
var zplCommand = string.Format("~DU{0}:{1}.FNT,{2},{3}",
    PrinterMemoryType.DRAM, "Arial001.FNT", fontByteArray.Length, fontData);
```

API explicitně čte paměť tiskárny za účelem zjištění volné paměti nebo zjištění, jaká data jsou v paměti nahrána. Výpis celé paměti lze provést pomocí příkazu ^HW, který vyžaduje jako jediný parametr označení paměti, ke které má přistupovat. Příkaz vrátí formátovaný ASCII text se všemi daty v paměti (vizte obr. 46).

```
- DIR R:*.*
* R:VER000.FNT  243420
* R:NAT001.FNT   12264
* R:NAT002.FNT   25584
* R:NAT003.FNT    9728
* R:VER004.FNT  211376

-  10604284 bytes free R: RAM
```

**Obrázek 46 – Výpis paměti DRAM (zdroj: [Autor])**

Výpis paměti také obsahuje zbývající volnou paměť, která se vždy udává v bajtech. Pro další práci s tímto údajem nebo pro konverzi na jiné, větší, jednotky, je třeba data extrahovat pomocí regexu[17]. Následuje příklad kódu extrahování volné paměti, použité v metodě *GetPrinterFreeMemory*.

```
var pattern = @"-\W.?(.+) bytes free";
var match = Regex.Match(printerMemoryData, pattern);
if (Regex.IsMatch(printerMemoryData, pattern))
        long freeMemory = long.Parse(match.Groups[1].Value);
```

Alternativou digitálního výpisu fontů je příkaz ^LF, který vytiskne seznam všech fontů ve všech pamětích na štítek (případně více štítků, pokud jich je hodně).

Zápis i čtení paměti vyžaduje připojení k tiskárně. Proto metody využívající manipulace s pamětí vyžadují vždy objekt třídy *PrinterSettings* s nastavením cílové tiskárny.

[17] Regulární výraz (zkráceně regex) popisuje množinu řetězců. Více na https://www.regexbuddy.com/regex.html.

## 5.4 Vázání dat (Data Binding)

Vázání dat (data binding) je jednou z nejdůležitějších částí velkoobjemového tisku štítků. Při tisku velkého množství štítků, například ve výrobě nebo logistice, není žádoucí statický text. Místo toho je vyžadována dynamika dat na štítku, která se budou během tisku aktualizovat dle specifického zdroje. Zdrojem může být například čas, datum nebo seznam hodnot.

Dynamická pole lze definovat v Editoru štítků (vizte kapitola 6). Veškerá dynamická pole se v API zpracovávají a pomocí knihovny od firmy Neodynamic se vážou na navrženou šablonu štítku.

ThermalLabel SDK umožňuje generování ZPL II a vazbu dat ve specifickém formátu. Knihovna se využívá převážně pro generování jazyka ZPL II. Na trhu existuje jen velmi málo knihoven, které umožňují generování PDL jazyků. Nad SDK byl vytvořen wrapper[18] *NeodynamicWrapper*, který zastřešuje generování a vázání dat a poskytuje prostor pro vlastní implementaci generování.

ThermalLabel SDK podporuje pouze několik málo formátů zdrojových dat, mezi které patří například .NET objekty, DataSet, XML soubory nebo ADO.NET. Komunikační API dále rozšiřuje stávající podporované formáty dat pomocí třídy *DataBinder*. Podpora je postavená na konverzi vstupních dat na tabulkový objekt třídy *DataSet*, který SDK podporuje. Tímto způsobem přidává podporu například pro datový formát JSON[19]. Následuje příklad konverze XML dat na DataSet.

```
internal DataSet BindXmlDataToDataSet(string xmlData) {
	using (StringReader sr = new StringReader(xmlData)) {
		DataSet data = new DataSet();
		data.ReadXml(sr);
		return data;
	}
}
```

Textové formáty dat (XML, JSON) se předávají do API pomocí rozhraní *IPrinterService*. Dalším, způsobem vázání dat je vázání na data z databáze. Místo vstupních dat se rozhraní poskytnou přihlašovací údaje do databáze s konkrétním dotazem ve struktuře *DatabaseSettings*.

[18] Wrapper je návrhový vzor umožňující obalení aplikace či knihovny. Wrapper programátorovi poskytne možnost implementace vlastního rozhraní a odstřižení externího programu.
[19] https://www.w3schools.com/whatis/whatis_json.asp

Třída *DatabaseQuery* vytvoří na základě poskytnutých informací připojení do databáze (connection string) a spustí zadaný dotaz do databáze. Výsledná data umístí do struktury *DataTable*, která umožňuje vázání dat pomocí SDK. Obr. 47 ilustruje sestavení připojení do databáze Oracle.

```
return $"Data Source=(DESCRIPTION =" +
        $"(ADDRESS = (PROTOCOL = TCP)(HOST = {host})(PORT = {port}))" +
        $"(CONNECT_DATA =" + "(SERVER = DEDICATED)" +
        $"(SERVICE_NAME = {serviceName})));" +
        $"User Id={user};Password={password};";
```

**Obrázek 47 – Sestavení připojení do databáze Oracle (zdroj: [Autor])**

Následuje ukázka připojení do databáze a naplnění struktury *DataTable*. Po naplnění tabulkové struktury se připojení uzavírá. Implementace v metodě *QueryToDataTable* dodatečně ošetřuje neplatné dotazy, chyby při připojení do databáze a další uživatelské chyby.

```
var cmd = new OracleCommand(
                                query, new OracleConnection(connectionString));
var adapter = new OracleDataAdapter();
adapter.SelectCommand = cmd;
var dataTable = new DataTable();
cmd.Connection.Open();
adapter.Fill(dataTable);
cmd.Connection.Close();
```

Třída *DatabaseQuery* je implementována tak, aby dokázala poskytnout připojení do více typů databází. Databáze Oracle je plně podporována s možností implementace dalších databází jako jsou MSSQL a PostgreSQL.

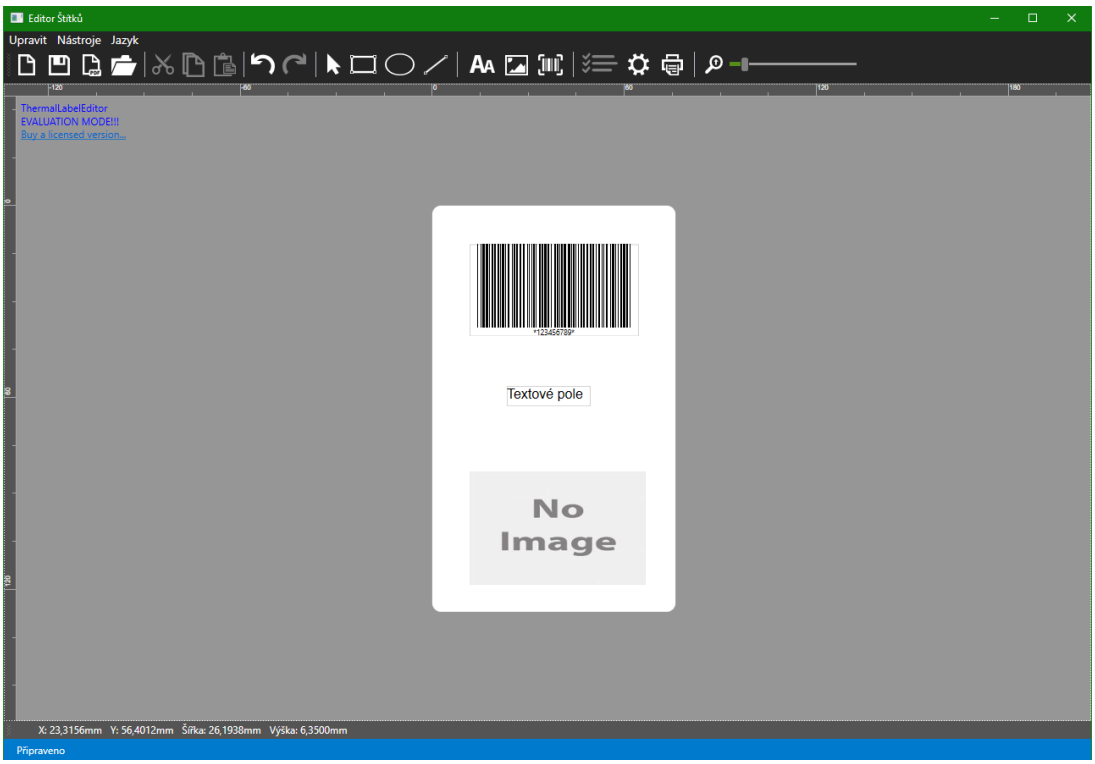
Pro každý záznam v tabulce či souboru se vytvoří jeden štítek dle dynamicky definovaných polí. V případě, že databázový dotaz vrátí 20 záznamů, SDK vygeneruje dle vložené šablony 20 rozdílných štítků. Vygenerované ZPL II s navázanými daty lze rovnou tisknout (v případě, že byla zavolána metoda *Print*) nebo pouze vrátit náhled ZPL II. Náhled ZPL II se využívá v Editoru štítků pro získání představy o velikosti ZPL II, jeho struktuře a případně dalšímu využití.

# 6 EDITOR ŠTÍTKŮ

Editor štítků (dále jen editor) je desktopová aplikace pro operační systémy Windows, která umožňuje návrh a tisk štítků. Editor poskytuje vykreslovací plochu pro návrh štítku nebo šablony štítku. Na vykreslovací plochu je možné vložit textové pole, vlastní obrázek, čárový kód či jeden ze tří základních grafických objektů. Prostředí je navrženo pro co největší pohodlí uživatele, poskytující funkce přiblížení, oddálení nebo kopírování.

Editor rovněž umožňuje tisk navrženého štítku či štítkové šablony nebo získání náhledu ZPL II. Veškerá komunikace je zprostředkována pomocí Komunikačního API (vizte kapitola 5). Mezi hlavní funkci editoru patří možnost definice dynamického štítku, do kterého se během tisku doplní data dle poskytnutého datového zdroje. Datové zdroje je možné poskytnout v podobě souborů XML, JSON nebo databázového připojení.

Pro realizaci editoru byl použit programovací jazyk C#, značkovací jazyk XAML, knihovny Neodynamic ThermalLabel SDK, SevenZip a grafický toolkit[20] MahApps.Metro[21] pro moderní vzhled aplikace. Editor nabízí připojení k tiskárně pomocí rozhraní USB, TCP/IP, LPT a COM. Návrh a tisk byl testován na průmyslové tiskárně Zebra 110Xi4. Ukázka vzhledu editoru je ilustrována na obr. 48.




**Obrázek 48 – Editor štítků se štítkem obsahující čárový kód, textové pole a obrázek (zdroj: [Autor])**

[20] Toolkit je software poskytující řadu nástrojů pro dosažení konkrétního cíle.
[21] https://mahapps.com/

## 6.1 Analýza a návrh

Před implementací editoru byla, podobně jako u Komunikačního API, provedena analýza a návrh aplikace. V rámci analýzy a návrhu byly vytvořeny funkční a nefunkční požadavky, které stanovují, co by editor měl umožňovat. Dále byly vytvořeny případy užití a jednotlivé scénáře. Celý model vypracovaný v programu Enterprise Architect se nachází v příloze A.

### 6.1.1 Funkční požadavky

V tabulce 17 jsou uvedeny funkční požadavky pro Editor štítků. Funkční požadavky určují jednotlivé funkce editoru a jaké služby by měl poskytovat uživateli. Požadavky byly rozděleny do balíčků stejně, jako v modelu EA.

**Tabulka 17 – Funkční požadavky Editoru štítků (zdroj: [Autor])**

| **Data** | |
|---|---|
| R1 | Editor štítků bude umožňovat ukládat vytvořenou šablonu štítku. |
| R2 | Editor štítků bude umožňovat načtení šablony štítku z disku. |
| R3 | Editor štítků bude umožňovat export štítku do formátu PDF. |
| R4 | Editor štítků bude poskytovat náhled ZPL II dat. |
| R5 | Editor štítků bude umožňovat dynamické vázání dat. |
| R6 | Editor štítků bude umožňovat vázat dynamická data na XML soubor. |
| R7 | Editor štítků bude umožňovat vázat dynamická data na JSON soubor. |
| R8 | Editor štítků bude umožňovat vázat dynamická data na databázové výsledky. |
| **Prostředí** | |
| R20 | Editor štítků bude umožňovat přepnutí jazyka. |
| R21 | Editor štítků bude umožňovat vložení a úpravu čárového kódu. |
| R22 | Editor štítků bude umožňovat vložení a úpravu základních grafických objektů. |
| R23 | Editor štítků bude umožňovat vložení a úpravu textového pole. |
| R24 | Editor štítků bude umožňovat vložení a úpravu obrázku. |
| R25 | Editor štítků bude umožňovat změnu fontu textů. |

### 6.1.2 Nefunkční požadavky

Nefunkční požadavky určují omezení editoru, jakým způsobem by měl komunikovat s tiskárnou a jak by měl ukládat navržené štítky. Nefunkční požadavky jsou uvedeny v tabulce 18.

**Tabulka 18 – Nefunkční požadavky Editoru štítků (zdroj: [Autor])**

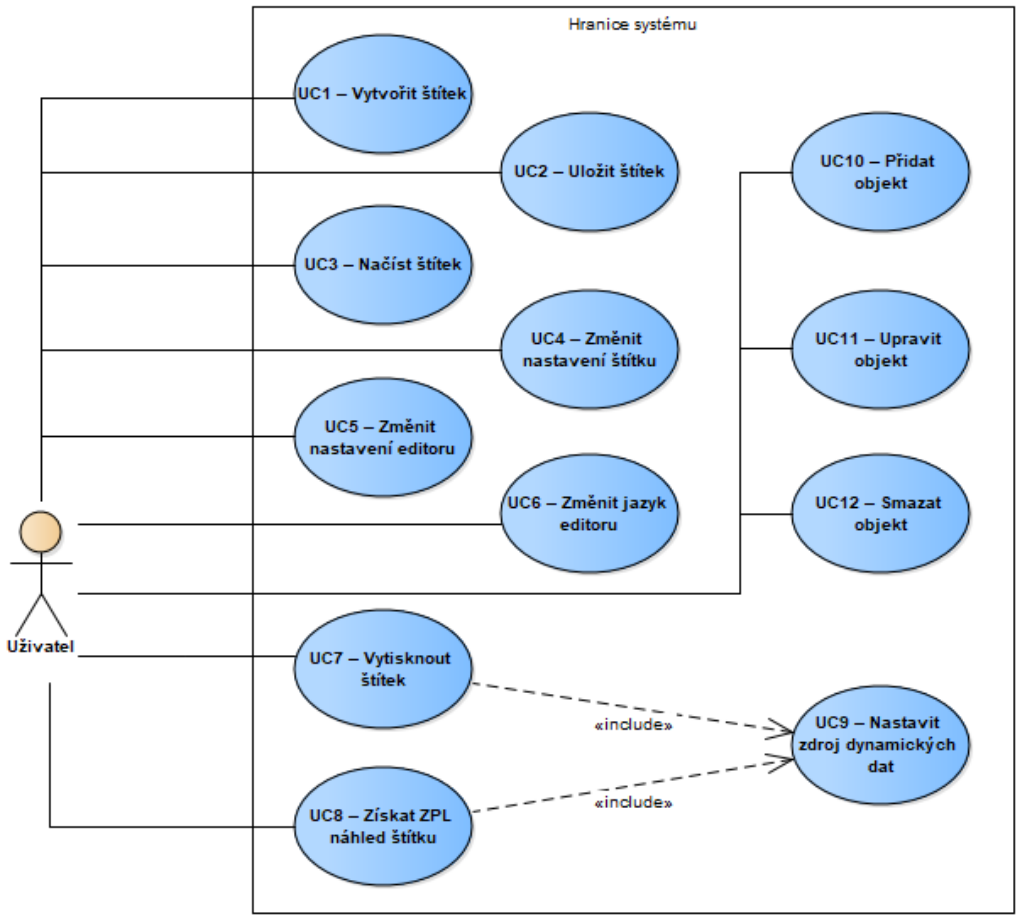
| | |
|---|---|
| **Data** | |
| R100 | Editor štítků bude ukládat šablony v komprimované formě. |
| **Prostředí** | |
| R200 | Editor štítků bude poskytovat více jazykových mutací. |
| R201 | Editor štítků bude podporovat více měrných jednotek. |
| R202 | Editor štítků bude poskytovat minimálně 5 typů čárových kódů. |
| R203 | Editor štítků bude umožňovat nahrát obrázek ve formátu Base64. |
| R204 | Editor štítků bude verifikovat uživatelské vstupy. |
| **Komunikace** | |
| R300 | Editor štítků bude komunikovat s tiskárnou pomocí Komunikačního API. |

### 6.1.3 Aktéři

Editor není informační systém, podobně jako Komunikační API, a tudíž nebylo možné nalézt více než jednoho aktéra. Jediným nalezeným aktérem je sám uživatel, který obsluhuje editor.

### 6.1.4 Případy užití

Případy užití a jejich scénáře popisují činnost aktérů a následně sled činností editoru. Mezi hlavní popisované případy užití patří vytvoření nového štítku, změna nastavení existujícího štítku a úprava grafického objektu na vykreslovací ploše. Celý diagram případů užití je ilustrován na obr. 49. Na model byla aplikována určitá míra abstrakce, aby se diagram udržel snadno čitelný a přehledný. Některé případy užití by totiž mohly obsahovat podrobnější popis (například přidání čárového kódu nebo přidání textového pole).




**Obrázek 49 – Případy užití Editoru štítků (zdroj: [Autor])**

Následuje popis vybraných případů užití a jejich scénářů (vizte tabulky 19–21). Kompletní model případů užití a scénářů je uveden v příloze A.

**Tabulka 19 – Případ užití UC1 – Vytvořit štítek (zdroj: [Autor])**

| |
|---|
| **Případ užití:** UC1 – Vytvořit štítek |
| **ID:** UC1 |
| **Stručný popis:** Vytvoření nového štítku na vykreslovací ploše se zadanými parametry. |
| **Aktéři:** Uživatel |
| **Podmínky:** Žádné |
| **Hlavní scénář:**<br>1. Uživatel zvolí možnost "Vytvořit štítek".<br>2. Editor štítků zobrazí dialogové okno.<br>3. Uživatel zadá parametry štítku.<br>4. Editor vytvoří nový štítek na vykreslovací ploše. |
| **Alternativní scénář:**<br>3a. Uživatel zruší akci. |
| **Výjimky:** Žádné |

**Tabulka 20 – Případ užití UC4 – Změnit nastavení štítku (zdroj: [Autor])**

| |
|---|
| **Případ užití:** UC4 – Změnit nastavení štítku |
| **ID:** UC4 |
| **Stručný popis:** Změna nastavení již vytvořeného štítku (například změna jeho velikosti). |
| **Aktéři:** Uživatel |
| **Podmínky:** Na vykreslovací ploše musí být aktivní štítek. |
| **Hlavní scénář:**<br>1. Uživatel zvolí možnost "Změnit nastavení štítku".<br>2. Editor štítků zobrazí dialogové okno s nastavením štítku.<br>3. Uživatel změní nastavení štítku.<br>4. Uživatel uloží změny. |
| **Alternativní scénář:**<br>4a. Uživatel zruší akci. |
| **Výjimky:** Žádné |

**Tabulka 21 – Případ užití UC12 – Upravit objekt (zdroj: [Autor])**

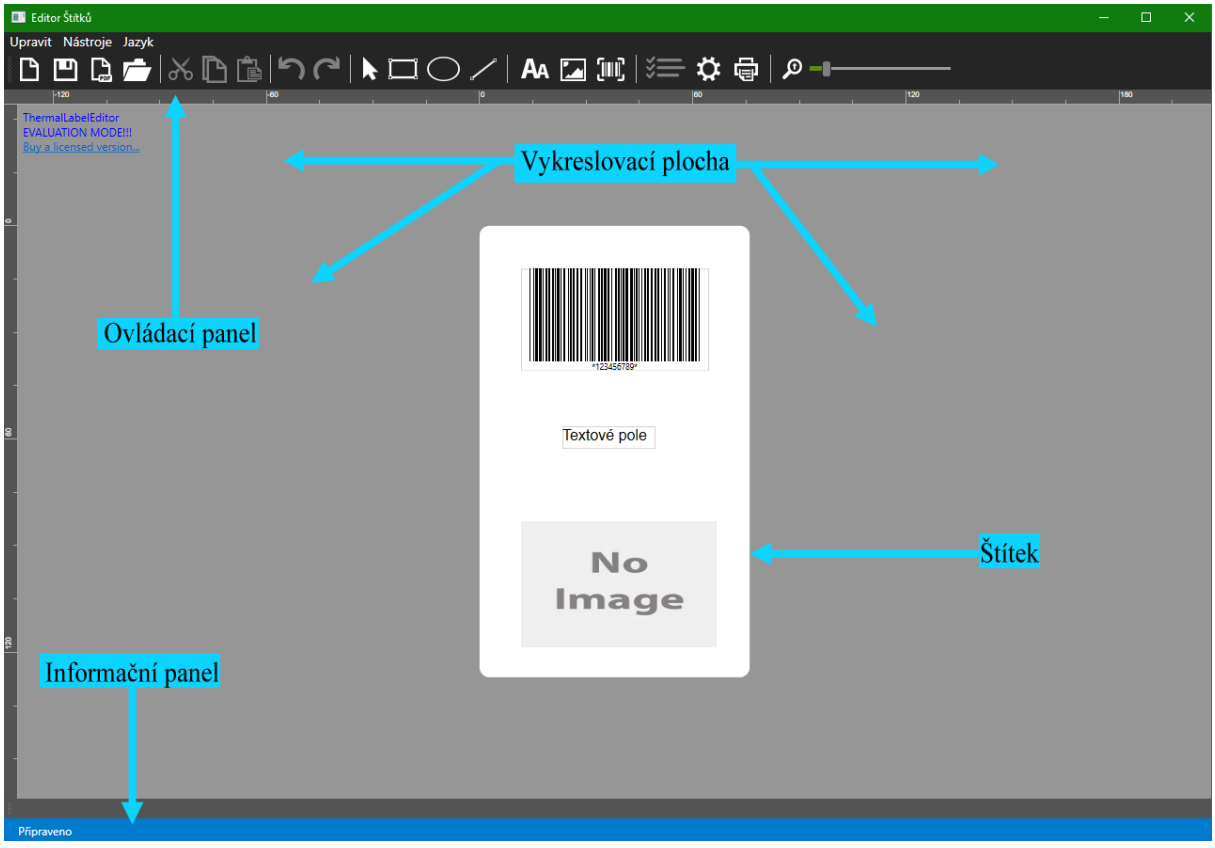
| |
|---|
| **Případ užití:** UC11 – Upravit objekt |
| **ID:** UC12 |
| **Stručný popis:** Úprava parametrů existujícího objektu (například čárového kódu). |
| **Aktéři:** Uživatel |
| **Podmínky:**<br>Na vykreslovací ploše musí být aktivní štítek.<br>Na vykreslovací ploše musí být grafický objekt. |
| **Hlavní scénář:**<br>1. Uživatel vybere objekt na vykreslovací ploše.<br>2. Uživatel zvolí možnost "Upravit objekt".<br>3. Editor zobrazí dialogové okno s parametry objektu.<br>4. Uživatel změní parametry objektu.<br>5. Uživatel uloží změny. |
| **Alternativní scénář:**<br>5a. Uživatel zruší akci. |
| **Výjimky:** Žádné |

## 6.2 Návrhové prostředí editoru

Návrhové prostředí editoru se skládá z několika částí. Hlavní a zároveň největší částí je vykreslovací plocha, která umožňuje vykreslování všech grafických objektů v editoru. Na této ploše je možné vytvořit nový štítek, který je symbolizován bílou barvou na šedém pozadí.

Aby uživatel mohl snadno vytvářet štítek a manipulovat s ním, v horní části editoru se nachází ovládací panel, který poskytuje nástroje a funkce pro práci se štítkem. Poslední funkční částí editoru je informační panel, který se nachází v dolní části. Informační panel informuje uživatele o stavu editoru. Zobrazuje informační hlášky o činnostech, které editor provádí v pozadí (např. tisk štítku) a poskytuje uživateli informace o chybách, které se vyskytly během používání. Jednotlivé části editoru jsou vizualizovány na obr. 50.

Celá vizuální stránka editoru je napsána v jazyce XAML. Vykreslovací plochu poskytuje knihovna Neodynamic ThermalLabel. Tato knihovna byla využita zejména kvůli vykreslování čárových kódů a manipulaci s grafickými objekty (např. obrázek).




**Obrázek 50 – Popis částí Editoru štítků (zdroj: [Autor])**

### 6.2.1 Ovládací panel

Ovládací panel poskytuje uživateli přístup k manipulaci se štítkem na vykreslovací ploše (vizte obr. 51). V první části panelu se nachází tlačítka pro vytvoření nového štítku, uložení, načtení a generování PDF dokumentu. Každé z těchto tlačítek vyvolá nové dialogové okno.


**Obrázek 51 – Ovládací panel Editoru štítků (zdroj: [Autor])**

Následují nástroje pro práci s existujícími grafickými objekty na vykreslovací ploše. Těmi jsou vyjmutí, kopírování, vložení a funkce zpětné či dopředné akce. Funkční část tlačítek byla realizována pomocí Neodynamic SDK. Tlačítka byla opatřena patřičným bindingem a příkazem ukazující na příkaz v knihovně. Zároveň jim byl definovaný vlastní styl zobrazení. Následuje ukázka kódu pro tlačítko kopírování.

```
<Button Focusable="False" Margin="10 0 5 0" Name="menuButtonCopy"
Style="{DynamicResource TransparentMenuButton}"
Command="neodynamic:EditorCommands.Copy" CommandTarget="{Binding
ElementName=thermalLabelEditor}" />
```

Dále ovládací panel obsahuje samotné grafické objekty, které lze použít pro návrh štítku. Mezi grafické objekty patří základní grafické obrazce, textové pole, obrázek a čárový kód. Po kliknutí na tlačítko se editoru nastaví aktivní nástroj a vygeneruje se zvolený objekt s přednastavenými hodnotami. Objekt je umístěn na vykreslovací plochu až poté, co uživatel tažením vyznačí polohu a velikost zvoleného grafického objektu. Pro zobrazení se využívají objekty SDK, které byly popsány v kapitole 3.3. Následuje kód vytvoření výchozího textového pole. Výchozím fontem je písmo Arial o velikosti 7 milimetrů. Dále se dynamicky nastavuje text dle zvoleného jazyka a generuje unikátní identifikátor.

```
this.thermalLabelEditor.ActiveTool = EditorTool.Text;
TextItem txtItem = new TextItem();
txtItem.Text = Properties.Localization.TypeHere;
txtItem.Font.Name = "Arial";
txtItem.Font.Size = 5;
txtItem.Font.Unit = FontUnit.Millimeter;
txtItem.TextAlignment = Neodynamic.SDK.Printing.TextAlignment.Left;
this.thermalLabelEditor.ActiveToolItem = txtItem;
```

Poslední část ovládacího panelu je věnována nastavení a tisku. Uživatel může modifikovat existující grafický objekt na vykreslovací ploše pomocí tlačítka nebo dvojím kliknutím levého tlačítka myši (výjimkou je textové pole). Mimo grafického objektu může uživatel měnit i nastavení samotného editoru, které poskytuje například polohovací mřížku. Nastavení tisku se věnuje kapitola 6.3.

Jelikož editor obsahuje mnoho nastavení a dialogů, budou v této kapitole popsány pouze některé funkcionality. Podrobnější popis jednotlivých funkcí editoru se nachází v uživatelské příručce, vizte příloha B.

### 6.2.2 Uložení a načtení štítku

Uživatel má možnost svůj navržený štítek uložit a opětovně otevřít v editoru. Po vybrání cílového umístění souboru a zvolení vlastního názvu zahájí editor proces ukládání. Data mají na začátku procesu podobu XML. Prvním elementem (po kořenovém elementu) je nastavení editoru. Nastavení se neukládá do dočasného adresáře, ale do každého štítku zvlášť. Tímto způsobem uživatel může mít různá nastavení editoru pro různé štítky. Příkladem uložení nastavení do elementů je následující kód.

```
XmlElement settingsElement = doc.CreateElement("Settings");
settingsElement.SetAttribute("ShowGrid",
                             this.thermalLabelEditor.ShowGrid.ToString());
settingsElement.SetAttribute("GridSize",
                             this.thermalLabelEditor.GridSize.ToString());
settingsElement.SetAttribute("AngleSnap",
                             this.thermalLabelEditor.AngleSnap.ToString());
```

Po nastavení editoru následuje element `<Items>`, který obsahuje všechny grafické objekty, které byly v době generování na vykreslovací ploše. Každý objekt má svůj vlastní element s vlastními atributy, který umožňuje zpětné načtení. Zapsáním všech grafických objektů do XML končí generování struktury souboru. Výsledkem generování je XML v podobě řetězce, které se následně pošle do komprimační metody. Komprimační metoda *CompressFileLZMA* využívá komprese LZMA z knihovny SevenZip[22]. Komprimace umožní zmenšení souboru a získání pole bajtů dat. Posledním krokem procesu ukládání je zapsání získaného pole bajtů do cílového adresáře pomocí následujícího kódu. Soubory editoru mají koncovku `.label`.

```
File.WriteAllBytes(saveDialog.FileName, compressedXml);
```

[22] https://www.7-zip.org/sdk.html

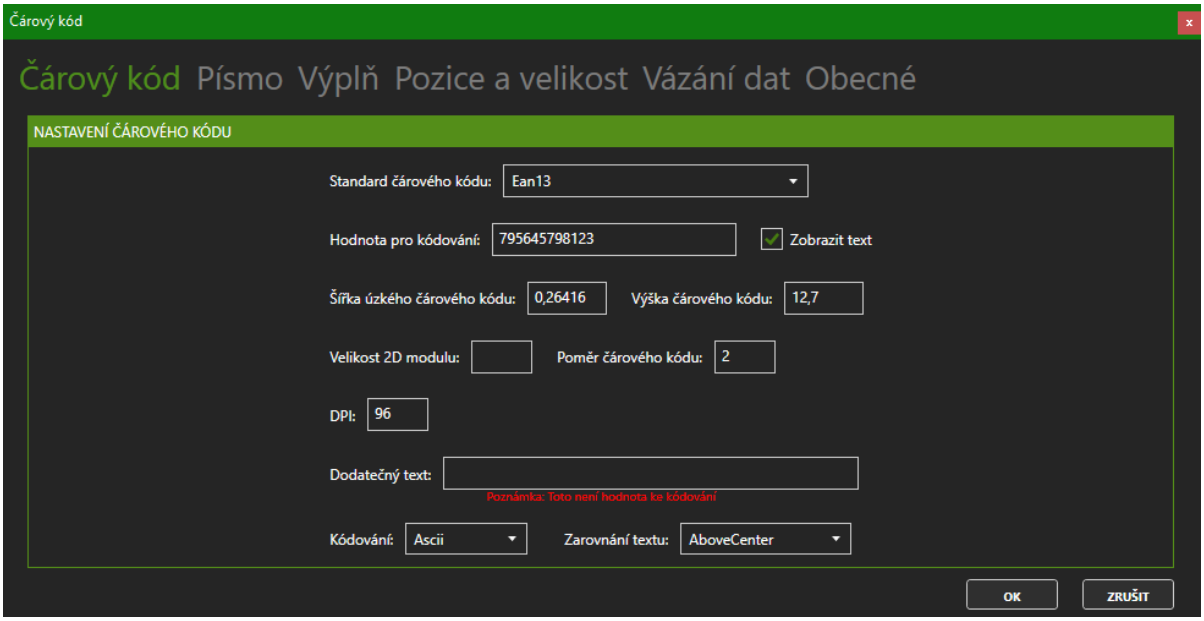
Načítání uloženého štítku je obrácený proces. Po vybrání souboru se do paměti načte pole bajtů, které obsahuje komprimované XML a zpětně ho dekomprimuje, opět pomocí LZMA. Výsledkem je čitelné XML (vizte obr. 52), které lze parsovat například třídou *XmlDocument*. Editor načte nová nastavení a všechny grafické objekty odešle do SDK pro vykreslení na vykreslovací plochu.

```
<ThermalLabel Version="7.0" Width="76" Height="127" GapLength="0" MarkLength = "0"  …
  <Settings DPI="600" ShowGrid="False" GridSize="0" AngleSnap="5" …
  <Items>
      <BarcodeItem Name="" X="11.674" Y="12.0788" DataField="ISBN" ….
      <ImageItem Name="" X="11.674" Y="85.125" DataField="" ….
      <TextItem Name="" X="23.3156" Y="56.426" DataField="BookTitle" ….
  </Items>
</ThermalLabel>
```

**Obrázek 52 – Ukázka čistého XML (zdroj: [Autor])**

### 6.2.3 Čárový kód

Uživatel má možnost modifikovat čárový kód na vykreslovací ploše a měnit jeho parametry. Po vybrání možnosti „Formátovat čárový kód“ se uživateli zobrazí dialogové okno (vizte obr. 53). Okno je složeno z několika menších uživatelských XAML komponent, které se využívají i v dalších oknech nastavení. Hlavní částí je nastavení samotného čárového kódu, které poskytuje změnu parametrů čárového kódu (např. kódování, typ čárového kódu a DPI). V dalších záložkách se nachází nastavení fontu, pozicování, výplně a uživatelské komentáře. Další významnou záložkou je „Vázání dat“. V této sekci uživatel poskytuje prvotní dynamické nastavení pro vázání dat (data binding). Podobným způsobem lze modifikovat veškeré grafické objekty na vykreslovací ploše.




**Obrázek 53 – Nastavení parametrů čárového kódu (zdroj: [Autor])**

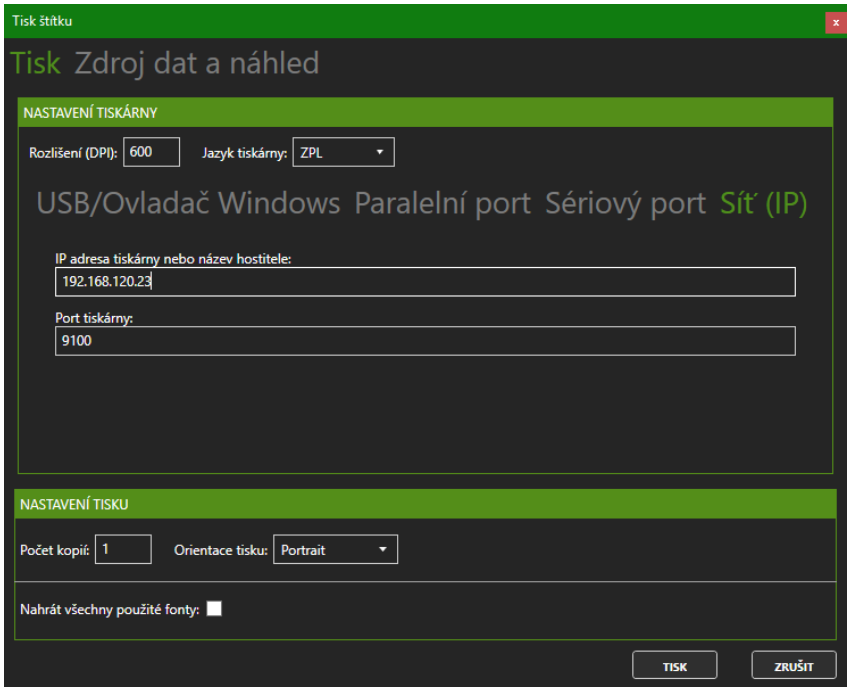
Po dokončení úprav a stisknutí tlačítka OK se zavolá metoda *UpdateItem*, která notifikuje SDK o změnách a následkem toho se aktualizuje grafický objekt na vykreslovací ploše. Následuje zjednodušený výtah kódu z metody *UpdateItem*.

```
private void UpdateItem<T>(bool dialogResult, Item newItem) where T : Item {
        Item currentItem = this.thermalLabelEditor.CurrentSelection as T;
        currentItem.UpdateFrom(newItem as T);
        this.thermalLabelEditor.UpdateSelectionItemsProperties();
}
```

## 6.3 Tisk štítku

Navržený štítek či šablonu štítku může uživatel odeslat k tisku. Před komunikací s tiskárnou musí uživatel nastavit typ připojení a údaje k připojení. Další nastavení je volitelné. Tisk se vyvolá tlačítkem „Tisk“ na ovládacím panelu nebo klávesovou zkratkou Ctrl+P. Editor zobrazí dialogové okno s nastavením tisku (vizte obr. 54). Hlavní částí dialogového okna je nastavení komunikace s cílovou tiskárnou. Jednotlivé záložky poskytují nastavení pro zvolený typ komunikace (například síťový tisk vyžaduje IP adresu tiskárny a port). Mezi další nastavení tisku patří například počet kopií a orientace tisku.

Potvrzením tisku odešle editor zadaná data do API přes rozhraní *IPrintService*. Zároveň si uchová poslední nastavení pro případ opakovaného tisku. Nastavení je uložené pouze v paměti a po ukončení programu je nutné nastavit připojení znovu.




**Obrázek 54 – Tisk štítku (zdroj: [Autor])**

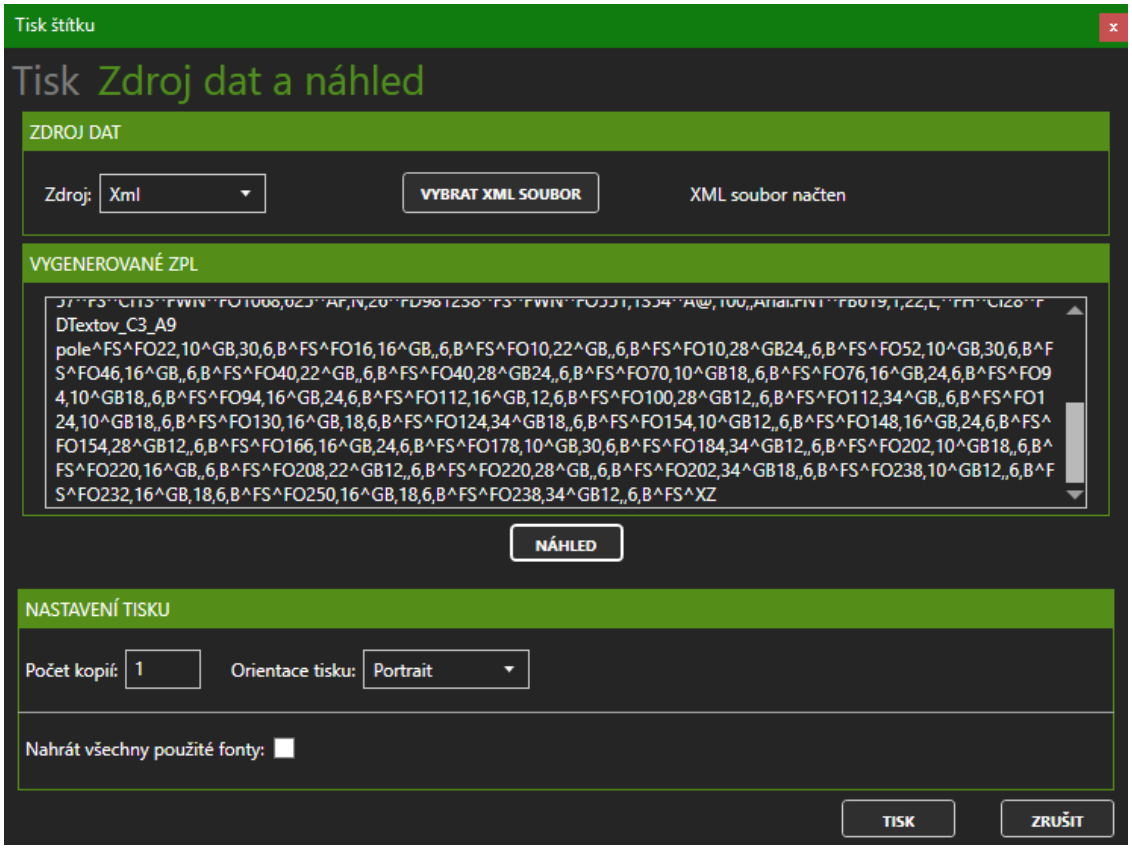
### 6.3.1 Zdroj dat a náhled ZPL II

Součástí tisku štítku je i nastavení zdrojových dat a náhled ZPL II (vizte obr. 55). Jako zdroje dat jsou podporovány soubory XML, JSON a databázové připojení. Po zvolení zdroje dat je nutné vybrat soubor s daty pomocí tlačítka „Vybrat soubor“. V případě databázového připojení je nutné zadat přihlašovací údaje do databáze, společně s dotazem, který vrátí datový set.

Data z datového zdroje se napojí dle názvu na dynamicky nastavený grafický objekt. V případě souborů se do dynamických objektů zadává název atributu elementu a v případě databázového spojení by se měl v objektu nacházet název sloupce tabulky. Vázání dat je citlivé na velká a malá písmena.

Místo tisku štítku je možné vygenerovat pouze ZPL II kód a použít ho k jiným účelům. Náhled ZPL II podporuje i vázání dat. V případě, že se ve zdroji nachází víc než jeden záznam, je vygenerován štítek pro každý záznam zvlášť. Pro náhled ZPL II se využívá Komunikační API.

Přesná struktura a formátování souborů XML a JSON pro vázání dat je popsána v uživatelské příručce (vizte příloha B).




**Obrázek 55 – Nastavení zdroje dat a náhled ZPL II (zdroj: [Autor])**

## 6.4 Jazykové mutace

Editor je vícejazyčný software a podporuje jazykové mutace všech textových řetězců. Jazyk editoru je možné přepnout pomocí ovládacího panelu v sekci „Jazyk“. V současné době mezi podporované jazyky editoru patří čeština a angličtina.

Jazyková mutace byla realizována pomocí zdrojových souborů *resx* a XAML bindingu. Texty výchozího jazyka se nachází v souboru *Localization.resx*. Další jazykové mutace je možné snadno přidávat pomocí dalších *resx* souborů se stejným názvem a kódem země. Česká lokalizace má název *Localization.cs-CZ.resx*. Ukázka jazykového souboru se nachází na obr. 56. První sloupec obsahuje unikátní kódové označení, na které se odkazuje XAML binding a v druhém sloupci se nachází text pro zobrazení. Unikátní kódové označení musí být ve všech jazykových *resx* souborech stejné.

| Name | Value |
|---|---|
| IPNetwork | Síť (IP) |
| IsContinuous | Je nepřerušovaný |
| Italic | Kurzíva |
| ItemRotation | Otáčení položek |
| JSONFileLoaded | JSON soubor načten |
| KeepOriginalSize | Zachovat původní velikost |
| LabelEditor | Editor Štítků |
| LabelFailedToLoad | Štítek se nepodařilo načíst |
| LabelSetup | Nastavení štítku |
| LabelTemplates | Šablony štítků |
| Language | Jazyk |

**Obrázek 56 – Lokalizační soubor pro český jazyk (zdroj: [Autor])**

Pro dynamickou změnu textu je nutné použít v XAML kódu binding (vázání) místo statických textů. Binding se nastavuje tak, aby dynamicky ukazoval na jednu hodnotu v lokalizačním souboru pomocí unikátního identifikátoru. Následuje ukázka nastavení bindingu textového pole s lokalizací `EditMenuItem`.

```
{Binding Source={x:Static loc:Localization.EditMenuItem}}
```

Aby XAML komponenta mohla využívat zdroje projektu, je nutné provést import těchto zdrojů. Toho lze docílit například následujícím kódem.

```
xmlns:loc="clr-namespace:DP.Properties"
```

Problém během implementace jazykové mutace nastává ve chvíli, kdy je vyžadována změna jazyka při běhu programu. XAML nedokáže měnit lokalizační soubory za běhu programu. Při vytvoření okna s dynamickým vázáním hodnot se načte zvolený lokalizační soubor do paměti a všechny dynamické hodnoty nahradí statickým textem. Při změně jazyka a následně i lokalizačního souboru již neprobíhá žádná aktualizace a okno si ponechává původní hodnoty textu.

Tento problém je možné řešit pouze několika způsoby. Jedním z nich je znovunačtení všech grafických částí programu a jejich textů. V případě, že program obsahuje mnoho oken a lokalizačních textů, může se znovunačtení stát náročné pro procesor i implementaci. U větších aplikací je jediným východiskem generické znovunačtení. Implementace takového řešení není triviální. Na trhu naštěstí existují knihovny, které znovunačtení zařizují[23].

V rámci diplomové práce bylo implementováno vlastní řešení změny jazyka. Řešení spočívá ve změně jazyka vlákna programu a následného uzavření hlavního okna aplikace. V rámci procesu je po uzavření hlavního okna vytvořeno nové okno aplikace s již změněným jazykem. Pro uživatele se tato akce může jevit jako restartování programu. Nevýhodou tohoto řešení je ztráta rozpracovaných dat uživatele. Před změnou jazyka je uživatel vyzván k uložení všech rozpracovaných změn.

Vlastní řešení bylo zvoleno pro jeho jednoduchost a pro lepší pochopení problematiky změny jazykových mutací za běhu programu.

[23] https://github.com/XAMLMarkupExtensions/WPFLocalizationExtension

# ZÁVĚR

Práce vznikla za účelem vytvoření univerzálního komunikačního API a návrhového editoru, který umožní uživateli snadný návrh štítku. První část práce se zaměřila na typy tiskáren, jejich výrobce a teoretické představení štítků a štítkové technologie. Rovněž byly představeny tři odlišné jazyky pro popis stránky, které lze použít pro komunikaci s tiskárnou. Z těchto jazyků byl vybrán jazyk ZPL II, kterému se zbytek práce podrobněji věnuje. Dále se práce zabývala tiskovým procesem a technologiemi tisku.

Konec teoretické části je věnován čárovým kódům a jejich konstrukci. Byla zde popsána jejich terminologie, typický vzhled a základní rozdělení. Podrobněji byl popsán jednorozměrný čárový kód EAN a dvourozměrný QR kód. U obou čárových kódů byla probrána konstrukce, složení dat a jakým způsobem lze čárový kód přečíst.

Na základě teoretických znalostí bylo vytvořeno Komunikační API a Editor štítků. Výsledkem práce je knihovna, poskytující univerzální komunikační rozhraní pro komunikaci s tiskárnami štítků pomocí USB, TCP/IP, LPT a sériového portu. Dále umožňuje manipulaci s pamětí tiskárny (například nahrání fontu a čtení paměti) a dynamické vázání dat. Druhým praktickým výstupem je softwarový nástroj Editor štítků, který umožňuje návrh štítku či štítkové šablony. S využitím Komunikačního API poskytuje uživateli ucelený software pro práci s tiskárnou a díky grafickému frameworku vypadá editor elegantně a moderně.

Pro vývoj byla použita knihovna Neodynamic ThermalLabel SDK, která mi pomohla s generováním ZPL II kódu a s vykreslovací plochou editoru. Rovněž bylo použito několik menších knihoven například pro komprimaci dat nebo získání informací o USB zařízeních. Během vývoje vlastní komunikace jsem se naučil navázat spojení různými způsoby, využívat externí funkce systémových knihoven a zacházet se C# ukazateli. Jednou z klíčových výzev pro mě byla USB komunikace, kdy jsem musel řešit nalezení korektního portu pro odeslání dat a následné otevření komunikace. Nad rámec zadání byla vytvořena podpora jazykové mutace pro Editor štítků. Díky tomu jsem si vyzkoušel tvorbu vícejazyčného softwaru.

Oba programy mají velký potenciál pro využití ve velkovýrobě a pro další vývoj. API je konstruováno tak, aby bylo možné snadno odstranit SDK a místo něj implementovat vlastní generování ZPL II kódu a provádět větší zásahy do tiskového procesu tiskárny. Taktéž je možné vytvořit větší podporu pro stávající databázové spojení.

# POUŽITÁ LITERATURA


[1] NOE, Audra. 7 Unique Label Types and When to Use Them. *Inland Packaging* [online]. April 20, 2016 [cit. 2019-03-20]. Dostupné z: https://www.inlandpackaging.com/blog/7-unique-label-types-and-when-to-use-them/

[2] ThermalLabel Basics. *Neodynamic* [online]. © 2003-2019 Neodynamic SRL [cit. 2019-03-20]. Dostupné z: https://www.neodynamic.com/Products/Help/ThermalLabel7.0/articles/thermal-label-basics.html

[3] Zebra Printers. *Zebra Technologies* [online]. ©2019 Zebra Technologies Corp. [cit. 2019-03-20]. Dostupné z: https://www.zebra.com/us/en/products/printers.html

[4] Desktop Printers vs. Industrial Printers. *TSC Blog* [online]. June 16, 2017 [cit. 2019-03-20]. Dostupné z: https://blog.tscprinters.com/desktop-vs-industrial-printers/

[5] *Zebra Technologies* [online]. ©2019 Zebra Technologies [cit. 2019-03-20]. Dostupné z: https://www.zebra.com/us/en.html

[6] *DYMO | Label Makers & Printers, Labels, CardScan, LabelWriter* [online]. ©2019 DYMO [cit. 2019-03-20]. Dostupné z: http://www.dymo.com/en-US

[7] DYMO. *LabelWriter 400 Series Printers Technical Reference Manual* [online]. ©2002-2008 Sanford, L.P. A [cit. 2019-03-20]. Dostupné z: http://download.dymo.com/Usermanuals/LabelWriter400SeriesTechRef_5_08.pdf

[8] Embossing tape. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2019 [cit. 2019-03-20]. Dostupné z: https://en.wikipedia.org/wiki/Embossing_tape

[9] ROUSE, Margaret. Page description language (PDL). *The Tech Dictionary and IT Encyclopedia: TechTarget* [online]. March, 2011 [cit. 2019-03-20]. Dostupné z: https://whatis.techtarget.com/definition/page-description-language-PDL

[10] Common Page Description Languages. *FreeBSD* [online]. ©1995–2019 The FreeBSD Project [cit. 2019-03-21]. Dostupné z: https://www.freebsd.org/doc/handbook/printing-pdls.html

[11] Programming Guide: ZPL II ZBI 2 Set-Get-Do Mirror WML. In: *Zebra Technologies* [online]. Lincolnshire, USA, 2018 [cit. 2018-10-04]. Dostupné z: https://www.zebra.com/content/dam/zebra/manuals/printers/common/programming/zpl-zbi2-pm-en.pdf

[12] How is ZPL II different from ZPL scripts for Zebra printers? *Honeywell* [online]. 8 January, 2016 [cit. 2019-03-21]. Dostupné z: https://honeywellaidc.force.com/supportppr/s/article/How-is-ZPL-II-different-from-ZPL-scripts-for-Zebra-printers

[13] Printer Command Language. *Compart* [online]. ©2019 Compart [cit. 2019-03-21]. Dostupné z: https://www.compart.com/en-US/pcl

[14] HP's History Of Printer Command Language (PCL). *CSGNetwork* [online]. ©1973-2019 CSG, Computer Support Group [cit. 2019-03-21]. Dostupné z: http://www.csgnetwork.com/hppclhist.html

[15] Printer Control Language: Users Guide and Reference Manual. *Avery Dennison* [online]. 25 November, 2009 [cit. 2019-03-21]. Dostupné z: https://printers.averydennison.com/content/dam/averydennison/rbis/global/en/documents/Product%20Support/Apparel/ManualsOther/SNAP_PCL_Programming_Manual_English.pdf

[16] TANG, Chih-Yu. *Printer Command Language (PCL)* [online]. October 11, 2006 [cit. 2019-03-21]. Dostupné z: http://www.cs.brandeis.edu/~dilant/cs175/Talks_1/%5bTang_T1%5d.pdf

[17] TIMPLEDON, Miriam T., Susan F. MARSEKEN a Lambert M. SURHONE. *Zebra Technologies: Zebra Technologies, Barcode, RFID, Smart label, Fortune 500, ISO 9001, JabilCircuit*. Betascript Publishing, 2010, 96 s. ISBN 978-613-0-55262-6.

[18] EPL2 (page mode) Sample Label Format. *Zebra Technologies* [online]. 11 January, 2003 [cit. 2019-03-21]. Dostupné z: https://support.zebra.com/cpws/docs/eltron/common/epl2_samp.htm

[19] EPL Programming: Guide. *Zebra Technologies* [online]. Illinois U.S.A.: ZIH, 2007 [cit. 2019-03-21]. Dostupné z: https://web.archive.org/web/20120417002930/http://www.zebra.com/id/zebra/na/en/documentlibrary/manuals/en/epl2_manual__en_.DownloadFile.File.tmp/14245L-001rA_EPL_PG.pdf

[20] PHILLIPS, James Lee. What Is "Bi-Directional" for Printers? *Small Business - Chron.com* [online]. 1 May, 2013 [cit. 2019-03-21]. Dostupné z: https://smallbusiness.chron.com/bidirectional-printers-59754.html

[21] KENNEDY, John a Michael SATRAN. Single-byte Character Sets. *Microsoft* [online]. May 31, 2018 [cit. 2019-04-05]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/Intl/single-byte-character-sets

[22] SEJPM. Encoding vs. Compression vs. Encryption. *Cryptography StackExchange* [online]. 24 September, 2016 [cit. 2019-04-05]. Dostupné z: https://crypto.stackexchange.com/questions/40215/encoding-vs-compression-vs-encryption

[23] Lossless Compression vs Lossy Compression. *GIS Geography* [online]. Feb 22, 2018 [cit. 2019-04-05]. Dostupné z: https://gisgeography.com/lossless-compression-vs-lossy-compression/

[24] HORDĚJČUK, Vojtěch. Kompresní algoritmus LZ77. *Vojta Hordějčuk aka voho – Software Engineer and Bedroom Music Producer* [online]. April 7, 2017 [cit. 2019-04-05]. Dostupné z: http://voho.eu/wiki/algoritmus-lz77/

[25] ČERNIČKO, Sergij. *Slovníkové metody komprese dat* [online]. Brno, 2011, 52 s. [cit. 2019-04-05]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117723. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií, Ústav počítačové grafiky a multimédií. Vedoucí práce Ing. David Bařina.

[26] COLLINS, Wayne. Raster Image Processing. *BC Open Textbooks* [online]. cca. 15. března, 2019 [cit. 2019-04-05]. Dostupné z: https://opentextbc.ca/graphicdesign/chapter/5-2-raster-image-processing/

[27] Various Authors. *Tecnologia grafica. Per gli Ist. Professionali per l'industria e l'artigianato*. 5. edizione. Verona: Ist. Salesiano, 2007, 952 s. ISBN 978-8889112014.

[28] PRITCHARD, Gordon. Rosettes – everything you didn't realize you needed to know. *The Print Guide* [online]. April 23, 2009 [cit. 2019-04-05]. Dostupné z: http://the-print-guide.blogspot.com/2009/04/rosettes-everything-you-didnt-realize.html

[29] Screening technologies in flexographic printing. *Gallus Ferd* [online]. April, 2011 [cit. 2019-04-05]. Dostupné z: https://www.gallus-group.com/archiv/en/desktopdefault_aspx/tabid-336/503_read-1132.html

[30] How to Select the Right Label Printer: Understanding the Convenience, Cost and Quality Considerations for Laser, Inkjet and Thermal Printers. *Intermec Technologies Corporation*[online]. 2010 [cit. 2019-04-05]. Dostupné z: https://www.codeone.pt/catalogos/CodeOne-Choosing-Thermal-Printer.pdf

[31] Laserová tiskárna. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-05]. Dostupné z: https://cs.wikipedia.org/wiki/Laserov%C3%A1_tisk%C3%A1rna

[32] WASTRADOWSKI, Matt. Thermal Printers vs. Inkjet Printers. *Graphic Products* [online]. January 19, 2017 [cit. 2019-04-05]. Dostupné z: https://www.graphicproducts.com/articles/thermal-printers-vs-inkjet-printers/

[33] Thermal Transfer vs. Direct Thermal: Five Key Considerations. *SATO* [online]. SATO America, [cca. 2018] [cit. 2019-04-05]. Dostupné z: https://www.satoamerica.com/resources/learning-center/white-papers/thermal-transfer-vs.-direct-thermal-five-key-considerations

[34] Difference Between Direct Thermal and Thermal Transfer Printing. *Online Labels* [online]. Online Labels, Inc, [cca. 2018] [cit. 2019-04-05]. Dostupné z: https://www.onlinelabels.com/articles/printing_direct_thermal_transfer.htm

[35] *Barcode, Labeling, Printing & Imaging components for .NET & PHP by Neodynamic* [online]. ©2019 Neodynamic SRL [cit. 2019-04-05]. Dostupné z: https://www.neodynamic.com

[36] *GS1 Czech Republic* [online]. ©2017 GS1 Czech Republic [cit. 2019-04-05]. Dostupné z: https://www.gs1cz.org/

[37] MINGLE, Katie. Barcodes. *99% Invisible* [online]. April 5, 2014 [cit. 2019-04-05]. Dostupné z: https://99percentinvisible.org/episode/barcodes/

[38] What is a barcode? *DENSO WAVE* [online]. DENSO WAVE INCORPORATED, [b.r.] [cit. 2019-04-05]. Dostupné z: https://www.denso-wave.com/en/adcd/fundamental/barcode/barcode/index.html

[39] Barcode Checksums. *Accusoft* [online]. ©2016 Accusoft Corporation [cit. 2019-04-05]. Dostupné z: https://help.accusoft.com/BarcodeXpress/v11.0/nodejs/Barcode%20Checksums.html

[40] BEN. What is the difference between 1D (Linear) and 2D barcodes? *IDAutomation Barcode Support Forum* [online]. June 18, 2012 [cit. 2019-04-05]. Dostupné z: https://support.idautomation.com/Barcode-Learning/_412

[41] EAN. *KEYENCE America* [online]. ©2019 KEYENCE CORPORATION [cit. 2019-04-08]. Dostupné z: https://www.keyence.com/ss/products/auto_id/barcode_lecture/basic/jan/

[42] EAN 13 a EAN 8: Nejznámější čárový kód pro zboží v obchodní síti. *Kodys* [online]. August 24, 2017 [cit. 2019-04-08]. Dostupné z: https://www.kodys.cz/technologie/carovy-kod/ean-13-ean-8

[43] QR Code: Structure, Usage and Applications. *Shield UI* [online]. February 9, 2014 [cit. 2019-04-08]. Dostupné z: https://www.shieldui.com/blogs/qr.code

[44] QR Code Basics. *Egoditor* [online]. ©QRCode-Generator.de 2019 [cit. 2019-04-08]. Dostupné z: https://www.qr-code-generator.com/qr-code-marketing/qr-codes-basics/

[45] QR code. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-04-08]. Dostupné z: https://en.wikipedia.org/wiki/QR_code

[46] QR Code Error Correction. *QR Stuff* [online]. December 14, 2011 [cit. 2019-04-08]. Dostupné z: https://blog.qrstuff.com/2011/12/14/qr-code-error-correction

[47] AXELSON, Jan. USB complete: the developer's guide. Fifth edition. Madison, WI: Lakeview Research, [2015]. ISBN 978-1931448284.

[48] MAKOFSKE, David B, Michael J DONAHOO a Kenneth L CALVERT. TCP/IP sockets in C#: practical guide for programmers. Boston: Elsevier, c2004. ISBN 9780124660519.

[49] Zebra Xi4 / RXi4 User Guide. *Zebra Technologies* [online]. 2013 [cit. 2019-04-18]. Dostupné z: https://www.zebra.com/content/dam/zebra/manuals/printers/industrial/xi4/xi4-ug-en.pdf

[50] CreateFileA function. *Windows Dev Center* [online]. May 12, 2018 [cit. 2019-04-18]. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/api/fileapi/nf-fileapi-createfilea

[51] Remote Serial Console HOWTO: Select a serial speed and parameters. *Linux Documentation Project* [online]. June 16, 2002 [cit. 2019-04-18]. Dostupné z: https://www.tldp.org/HOWTO/Remote-Serial-Console-HOWTO/preparation-setspeed.html

# PŘÍLOHY

Příloha A – Projekt Enterprise Architect

Veškeré případy užití, scénáře a požadavky jsou namodelovány v projektu Enterprise Architect, který je přiložen na CD.

Příloha B – Uživatelská příručka Editoru štítků

Uživatelská příručka Editoru štítků popisuje celé návrhové prostředí editoru a seznamuje uživatele s jeho obsluhou. Příručka je sepsána do dokumentu ve formátu PDF a přiložena na CD.