TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Vytvoření programu pro obsluhu výrobního pracoviště laserového značení dílů

Diplomová práce

Studijní program:	N2612 – Elektrotechnika a informatika
Studijní obor:	1802T007 – Informační technologie
Autor práce:	**Bc. Leonid Kantalinskiy**
Vedoucí práce:	Ing. Pavel Tyl

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Vytvoření programu pro obsluhu výrobního pracoviště laserového značení dílů

Jméno a příjmení: Leonid Kantalinskiy

Osobní číslo: M19000154

Studijní program: B2612 Elektrotechnika a informatika

Studijní obor: Informační technologie

Zadávající katedra: Ústav mechatroniky a technické informatiky

Akademický rok: 2020/2021

Zásady pro vypracování:

1. Vymyslet algoritmus, který bude efektivně a s minimálním vstupem pracovníka připravovat vstupní data pro pracoviště laserového značení plastových dílů a dále bude kontrolovat načtené a zadané hodnoty a ověřovat jejich validitu před zápisem do databáze.
2. Vytvořit program, který umožní třídit data podle zadaných parametrů a zobrazí základní statistický přehled z dat uložených v databázi.
3. Vytvořit program, který zajistí obousměrné komunikační propojení mezi PC a řídícím PLC, vyhodnocovat datové vstupy a vizualizovat aktuální stav připojených zařízení, zabezpečeně přistupovat do připravených databází běžících na podnikovém serveru a umožní tvorbu nových šablon pro značení produktů.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 40–50
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] Matt Weisfled. The Object-Oriented Thought Process. Addison. Wesley, 2018. ISBN: 978-01-351-8166-6.

[2] Michal James Hernandez. Database Design for Mere Mortals. A Hands-On Guide to Relational Database Design. Grada, 2006. ISBN: 80-247-0900-7.

[3] Rudolf Pecinovský. Návrhové vzory. Computer Press, 2015. ISBN: 978-80-251-1528-4.

Vedoucí práce: Ing. Pavel Tyl
Ústav mechatroniky a technické informatiky

Datum zadání práce: 9. října 2020
Předpokládaný termín odevzdaní: 17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 9. října 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem. Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněn Technickou univerzitou v Liberci 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. Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

17. 05. 2021 Leonid Kantalinskiy

Abstrakt

Práce je zaměřena na vytvoření speciální aplikace pro obsluhu výrobního pracoviště laserového značení dílů. Tato aplikace automatizuje proces evidence označených dílů a ukládá příslušné informace k nim, generuje správně sériové číslo odpovídající určitým požadavkům a ukládá všechna potřebná data do databáze.

Byly testovány různé metody pro vytvoření algoritmu. S pracovníky firmy bylo projednáno, jaké nestandardní situace mohou vzniknout v procesu laserového značení dílů. Pak byl vytvořen algoritmus, který splňuje všechny požadavky firmy.

Bylo provedeno testování aplikace v různých pracovních situacích. Během tohoto testování bylo zjištěno, že aplikace funguje stabilně. Výsledkem provedené práce je aplikace *MarkingControl*, která je již prakticky používána pro účely obsluhy výrobního pracoviště laserového značení dílů.

Klíčová slova: Čtečka čárových kódů, čárový kód, laserové značení.

Abstract

The work is focused on the creation of a special application for the operation of the production workplace of laser marking of parts. This application must automate the process of registration of marked parts and related information, generate the correct serial number corresponding to certain requirements. Save all necessary information to the database.

During programming, various methods for creating the algorithm were tested. It was discussed with the company's employees what non-standard situations may arise in the process of laser marking of parts. Then an algorithm was created that meets all the requirements of the company.

The application was tested in various work situations. During this testing, the application was found to work stably. The result of the work is the application *MarkingControl*, which will be practically used for the purpose of operating the production workplace of laser marking of parts.

Key words: Bar code reader, bar code, laser marking.

Poděkování

Touto cestou bych rád poděkoval vedoucímu mé diplomové práce panu Ing. Pavlu Tylovi za jeho odborné rady, které výrazně přispěly k dokončení této práce. Zároveň bych chtěl poděkovat Mgr. Kamilu Nešetřilovi, Ph.D. a Ing. Jiřímu Jankelovi za cenné rady při konzultacích. V neposlední řadě bych rád poděkoval celé své rodině za podporu.

Obsah

Seznam zkratek

PLC	Programmable Logic Controller
DB	Database
PC	Personal Computer
USB	Universal Serial Bus
SQL	Structured Query Language
UI	User Interface
COM port	Communication port
Wi-Fi	Wireless Fidelity
CSV	Comma-separated Values

Seznam obrázků

Seznam schémat a diagramů

Úvod

Laserové značeni dílů je velmi důležitý proces ve výrobě. Tento proces umožňuje evidovat různé informace, umístit přímo do plastového dílu sériové číslo, logo nebo jinou potřebnou informaci. Pro ovládání laserového přístroje pro značení plastových dílů ve společnosti zákazníka se používá program *MaxMarking* od společnosti *Maxphotonics* [1]. Tento program umožňuje pouze samotný proces laserového značení plastových dílů, nikoliv evidenci informací, o již označených dílech. Z toho vyplývá cíl projektu – vytvořit program, který musí umožňovat generování sériového čísla a dalších potřebných informací pro značení, které pracovník musel dříve provádět ručně. Tento program tedy musí umožňovat zápis všech potřebných údajů do databáze. Uživatel musí mít možnost vytvořit nové šablony pro značení či prohlížet existující data. Cílem diplomové práce je návrh a implementace aplikace pro obsluhu výrobního pracoviště laserového značení dílů a další generování a ukládání sériových čísel i dalších potřebných informací do externí databáze běžící na podnikovém serveru.

Pro nový program byl zvolen název *MarkingControl*. Tento program musí umět řešit několik problémů. Popis těchto problémů zahrnuje jak situace v běžném režimu značení plastových dílů, tak i teoretické možné případy značení, které mohou vzniknout během využití programu na výrobním pracovišti.

Hlavní úloha, která byla vyřešena, je vyvinout spolehlivý, efektivní program, který bude užitečný pro obsluhu výrobního pracoviště laserového značení dílů. Program, který bude pomáhat pracovníkům v rutinní práci, při které pracovník může udělat náhodnou chybu.

Problémy, se kterými se může setkat pracovník, jsou neexistující číslo produktu, neexistující sériové číslo dílů podle databáze, nesprávnost tvaru čísla průvodky, nesprávnost tvaru čísla pracovníka, nesprávnost nastavění poloh pojezdového stolu, chyba připojení k databázi, chyba stavu čtečky pro načítání dat, chyba komunikace PLC a PC, problém manuálního značení dílů. Tyto problémy byly vyřešeny aplikací *MarkingControl*. Aplikace provádí kontrolu správnosti údajů a zrychluje práci při laserovém značení dílů.

Teoretická část zahrnuje popis historie vzniku značení produktů, cíle použití čárového kódu *Code 128*, popis použité čtečky čárových kódů a laseru, který je použit pro aplikaci *MarkingControl*, popis a principy fungovaní aplikace *MarkingControl*. Teoretická část také zahrnuje základní principy pro vytvoření ergonomického uživatelského rozhraní a popis úloh, které autor vyřešil. Praktická část zahrnuje implementaci aplikace v integrovaném vývojovém prostředí *Microsoft Visual Studio 2019* v jazyce *C#*, testování a dosažené výsledky.

První kapitola praktické části obsahuje vypracování algoritmu programu, který odpovídá požadavkům zákazníka a zároveň bude realizovatelný s pomocí standardních knihoven jazyka C#. Tento program musí obsahovat ergonomické rozhraní. Pracovník musí dokázat ovládat program v principu „jedním tlačítkem“. Rozhraní musí být dobře viditelné v podmínkách výrobního pracoviště.

Druhá a třetí kapitola praktické části popisuje vytvoření příslušné databáze, ve které jsou evidovány označené díly či uloženy šablony pro značení. Databáze také slouží pro účely poskytování informací pro řídicí PLC, jako je nájezd na správnou polohu pojezdovým stolem, do kterého je rozmístěn přípravek pro fixaci plastových dílů pro účel značení laserem. Také

druhá databáze slouží pro účely zobrazování všech potřebných údajů při laserovém značení dílů v aplikaci.

1 Značení

Tato část se týká historie vzniku značení produktů, historie vzniku čárových kódů, použití čárových kódů ve výrobě a popisu vybrané čtečky čárových kódů pro aplikaci *MarkingControl*. Také je zde stručně popsán proces laserového značení plastových dílů.

1.1 Značení dílů

V současné době je na trhu obrovské množství nabídek různých druhů produktů. Výrobci, kteří jsou v konkurenčním prostředí, jsou nuceni stále více bojovat za stabilní pozici na trhu a za dosažení nejvyššího zisku. Úkolem každého podniku je uspokojit poptávku lépe a efektivněji než konkurenti. Konkurenceschopnost je spojena se dvěma ukazateli – cenovou úrovní a úrovní kvality produktu. Druhý faktor se navíc postupně dostává do popředí, kde balení a označování je stimulem spotřebitele k jeho nákupu. V moderním světě je velmi důležitá svoboda volby produktu v preferované kvalitě na základě přístupu k úplným informacím o výrobku, které by měl spotřebiteli poskytnout výrobce nebo prodejce. Značení zboží má proto pro spotřebitele velký význam. Je hlavním nositelem příslušných informací o produktu. Může se jednat o informace vyžadované zákonem a další informace poskytované dobrovolně na základě jejich potřeby či užitečnosti pro výrobce, spotřebitele a další strany zapojené do procesu oběhu tohoto produktu. Rychlé a spolehlivé označování zboží je základním atributem moderního podniku. Označení je vyžadováno nejen přímo při výrobě zboží, ale také při jeho skladování a prodeji. Tím je dosaženo jasné regulace a včasné identifikace produktů, což značně usnadňuje a zrychluje celý výrobní proces.

Označením jsou text, symboly, jakož i další pomůcky určené k identifikaci výrobku nebo jeho jednotlivých vlastností, s cílem přinést spotřebiteli informace o produktu.

Poprvé se nápisy objevily v Egyptě ve III. tisíciletí před naším letopočtem. Byly to džbány se jmény vládnoucích faraonů. Kromě toho začali psát jména vlastníků, data výroby, a dokonce i značky výběrčích daní. Pouze obsah ještě nebyl zmíněn. K tomu došlo až v polovině II. tisíciletí před naším letopočtem. Informace v označení džbánků na víno vzrostly téměř do rozsahu moderní etikety na víno. Jednalo se o místo, kde byly hrozny sklizeny, jeho odrůda, chuť (kyselá, sladká), věk vína a místo výroby. Značení usnadňovalo prodej a přepravu, protože většina džbánů byla zpravidla stejného tvaru a zcela neprůhledná. Takže bez označení nebylo možné identifikovat obsah džbánu bez nutnosti otevření každého džbánu. [2, 3]

1.2 Čárový kód

Čárový kód je element pro automatizovaný sběr dat. Čárový kód je tvořen černě-tiskem vytištěnými pruhy (v některých novějších verzích kódu mozaikou) definované šířky, umožňující přečtení pomocí technických prostředků – čteček (pro jednorozměrné kódy) či skenerů (pro jedno i dvourozměrné kódy). Myšlenka je z roku 1949. Patent pro čárový kód byl poprvé udělen v roce 1952 Normanu Josephu Woodlandovi a Bernardu Silverovi. Podle způsobu, jakým se konkrétní znak kóduje do skupiny pruhů, se kódy dělí do skupin. [4, 5]

1.3 Čárový kód – Code 128

Formát čárového kódu *Code 128* se výrazně liší od široce uznávaných standardů čárového kódu, jako je například EAN. Rozdíly jsou především v možnosti kódování nejen čísel, ale také písmen latinské abecedy, jakož i specifických znaků. Kromě toho se digitální kód ve formátu *Code 128* stává velmi kompaktním, což je dosaženo díky „dvojímu zabalení“ dat, když jsou dvě číslice zapsány do jedné šablony čárového kódu. Abecední znaky jsou kódovány obvyklým „jednoduchým“ způsobem, což způsobuje, že abecední kód ve formátu *Code 128* má dvojnásobnou délku ve srovnání s digitálním. [4, 5]

1.4 Struktura čárového kódu – Code 128

Code 128 obsahuje 107 znaků, z toho 103 datových znaků, 3 stavy a 1 znak – stop symbol. Pro zakódování všech 128 znaků ASCII jsou zobrazeny tři znakové sady, jsou to: Code 128 A, B a C. Code 128 A – znaky ASCII od 00 do 95 (čísla od „0“ do „9“ a písmena od „A“ do „Z“), speciální znaky a znaky FNC 1-4. Code 128 B – znaky ASCII od 32 do 127 (čísla od „0“ do „9“, písmena od „A“ do „Z“ a od „a“ do „z“), speciální znaky a znaky FNC 1–4. Code-128 C – čísla od 00 do 99 (dvoumístné číslo je zakódováno jedním znakem) a FNC 1. Struktura čárového kódu Code 128 je poměrně jednoduchá. Čárový kód se skládá ze šesti částí: bílé pole, start symbol (*Start*), kódované informace, kontrolní číslice (zaškrtnutí); symbol stop (*Stop*) a bílé pole. Symboly čárových kódů 128 se skládají ze tří čar a tří mezer. Čáry a mezery jsou modulární. Šířka každé čáry a mezery je od 1 do 4 modulů (1 modul = 0,33 mm). Šířka symbolu je 11 modulů. Stop symbol (*Stop*) se skládá ze třinácti modulů a má čtyři čáry a tři mezery. [4, 5]

1.5 Ruční čtečka čárových kódů

Čtečka čárových kódů je elektronické zařízení, které dává možnost číst a vysílat data z čárového kódu do počítačové aplikace.

Ruční čtečka čárových kódů má rukojeť, konstrukce čtečky tak umožňuje používat čtečku manuálně. Ruční čtečka je nejčastěji používána v obchodech. Společnost zákazníka používá ruční čtečky čárových kódů pro identifikaci produktů.

1.6 Čtečka Eclipse MS5145 LS

Honeywell je jednou z velkých mezinárodních společností v USA. Byla založena v roce 1906. Honeywell Laser MS5145 Eclipse je jedno-paprsková ruční laserová čtečka čárových kódů, disponuje USB portem (obrázek 1) pro připojení, automatickým rozlišením standardních čárových kódů a aktivačním tlačítkem CodeGate, které spouští režim snímání čárových kódů. Čtečka umožňuje snímat jak standardní, tak i husté čárové kódy, které snímá rychlostí 72 sejmutí za sekundu. Její technické parametry jsou následující: laserová dioda – viditelné spektrum (650 nm ± 10 nm), výkon 675 mW, hloubka pole: 0–140 mm (100 % EAN), rozlišení: 0,102 mm, kontrast: minimum 35 %, úhel rotace: 42°, rozteč 68°, vychýlení 52°. Maximální délka snímaného údaje: až 80 znaků. Signalizace: zvuková (7 druhů tónů) nebo bez zvuku. Rozměry: 169 × 40 × 31 mm (rukojeť) / 63 × 35 mm (hlavice). Hmotnost: 100 g. [6]

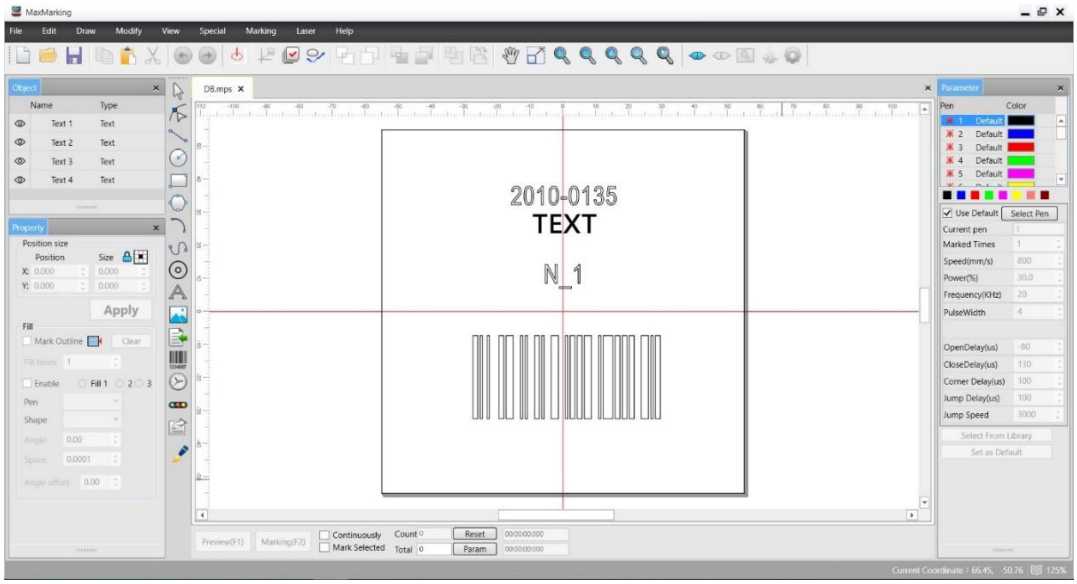
Obrázek 1: Ruční čtečka čárových kódů Honeywell, Eclipse MS5145 LS

1.7 Laserové značení plastových dílů

Značení laserem je proces, během kterého jsou jednotlivé díly označeny různými znaky jako například sériové číslo nebo logo. Laserové značení dílů je nekontaktní optický proces, během kterého plast absorbuje laserový paprsek. Výsledkem tohoto procesu je díl označený potřebnou informací.

1.8 MaxMarking

Pro vytvoření designu šablon a pro účel laserového značení slouží speciální program, který je poskytován spolu se přístrojem pro laserové značení (*HXM20 – COMPACT III*). Tento software má název *MaxMarking* (obrázek 2). Software byl vytvořen společnosti *Maxphotonics*. [1]



Obrázek 2: Uživatelské rozhraní aplikace MaxMarking verzi 2.7.10

1.9 Laser HXM20 – COMPACT III

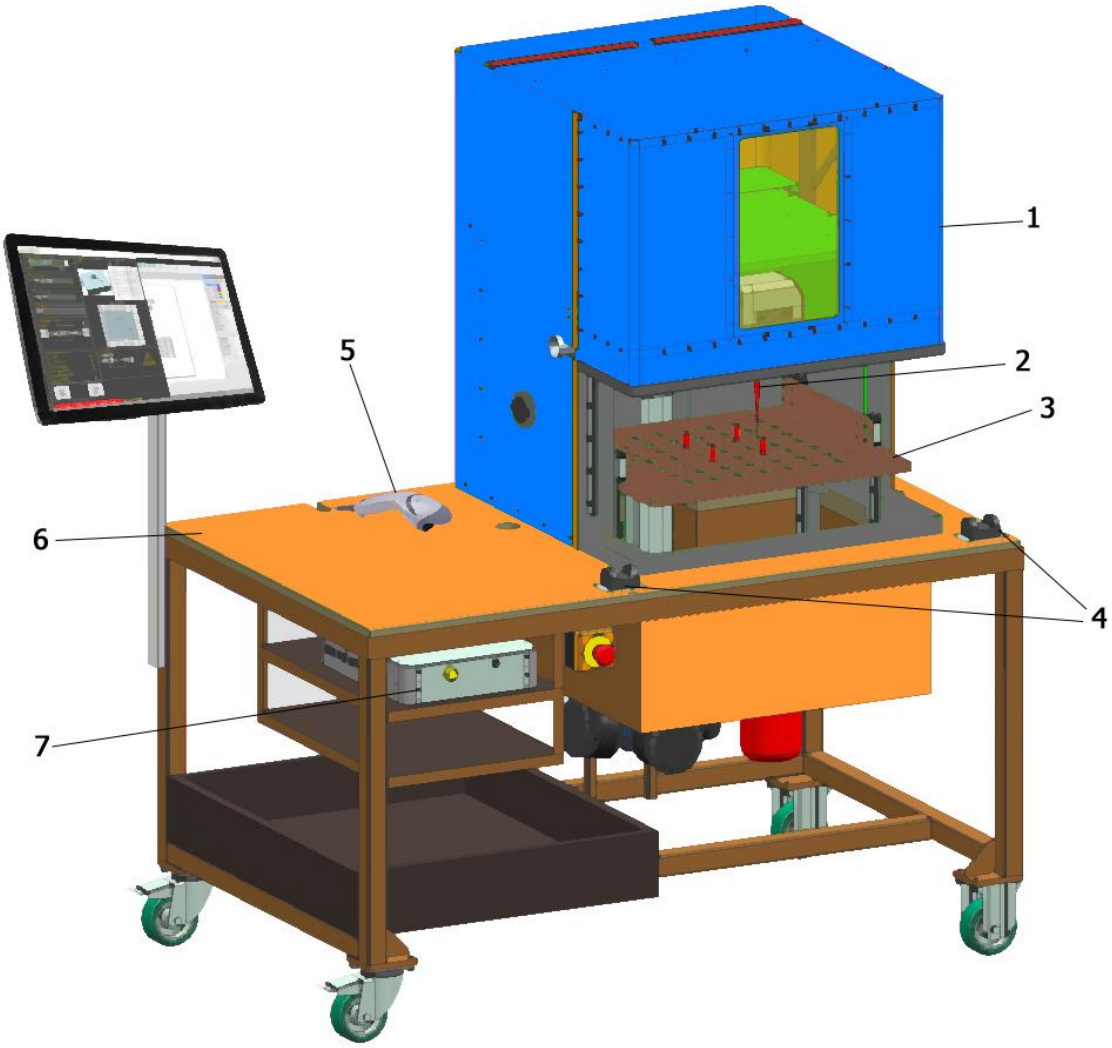
Pro účel značení plastových dílů firma již měla k dispozici přístroj, který má název *HXM20 – COMPACT III* (obrázek 3) od společností *MEPAC CZ*. Tento produkt má v sobě několik elementů, jsou to: držák na laser, který pak byl nahrazen vytvořeným strojem, počítač řídící laser, laser pro značení.

Tento laser má následující parametry: nominální síla – 20 W, provozní režim – pulzní, délka vlny – 1064 ± 5 nm, energie pulzu – 0,68 mJ, délka pulzu – 80~120 ns, frekvence – 20–100 kHz, napájecí napětí – 220 V, provozní teplota – od +10 do +35 °C, chlazení – vzduchové, hmotnost – 10 kg. [7]

Obrázek 3: HXM20 – COMPACT III
Zdroj: https://www.profilaser.eu/project/hxm20-compact-iii

1.10 Stroj pro laserové značení plastových dílů

Firma zákazníka pro účel laserového značení plastových dílů vytvořila speciální stroj (obrázek 4). Autor práce vytvořil pro tento stroj aplikaci *MarkingControl*. Tento stroj je mobilní, což je velká výhoda. Na podvozku stroje jsou instalovány všechny elementy. Připojení k firemní síti může být zajištěno bezdrátově či síťovým kabelem. Na stroji jsou umístěné dva počítače, jeden od společnosti *MEPAC CZ* pro účel řízení laseru, druhý pro aplikace *MaxMarking*, *MarkingControl*, a tedy pro komunikaci s řídicím PLC.



Obrázek 4: Stroj pro laserové značení plastových dílů

1 – Bezpečnostní pojezdový kryt.

2 – HXM20 – COMPACT III – laser od společností MEPAC CZ instalovaný uvnitř krytu.

3 – Pojezdový stůl pro přípravek, kam se vkládá plastový díl, pro který proběhne značení.

4 – Bezpečnostní ovládací tlačítka – spustí proces značení, jen když budou přiloženy oba prsty obou rukou. Pracovník musí počkat, pokud pojezdový stůl nebude nastaven na správnou pozici. Když pracovník odejme prsty, proces pojezdu stolu z bezpečnostního hlediska bude zastaven.

5 – Ruční čtečka čárových kódu Eclipse MS5145 LS od společnosti Honeywell.

6 – Deska stroje.

7 – Počítač ovládající laser.

2 Uživatelské rozhraní

Tato část se týká vývoje ergonomického rozhraní pro uživatele aplikace *MarkingControl*. Tento vývoj proběhal v souladu s požadavky společnosti zákazníka, pro kterou byla vytvořena tato aplikace.

2.1 Ergonomie

Ergonomie (z řečtiny *ergon* – práce a *nomos* – zákon) je vědecká disciplína zabývající se poznáním a pochopením interakcí mezi lidmi a dalšími prvky systému a profesí, která aplikuje teorie, principy, data a metody navrhování systémů tak, aby optimalizovala pohodu člověka a celkový výkon systému. Samotný pojem ergonomie poprvé použil polský vědec a profesor zemědělsko-lesnického institutu ve Varšavě Wojciech Jastrzebowski ve své práci „Rys ergonomii czili nauky o pracy“ roku 1857, ve které vymezil ergonomii jako vědu o práci.

2.2 Ergonomické rozhraní

Uživatelské rozhraní se skládá ze dvou hlavních částí. Jednak je to návrh uživatelského zážitku (*UX – User eXperience*), jednak návrh samotného uživatelského prostředí (*UI – User Interface*). UX designéři mají za úkol především navrhnout architekturu uživatelského prostředí tak, aby bylo srozumitelné a dobře se používalo. Úkolem UI designérů je, aby grafická podoba uživatelského prostředí byla přehledná, dobře vypadala a byla rozeznatelná od konkurenčních produktů.

Mezi primární metody používané při návrhu rozhraní patří prototypování a simulace. Typický design uživatelského rozhraní se skládá z následujících fází: specifikace interakce, specifikace softwaru rozhraní a prototypování:

- Mezi běžné postupy pro specifikaci interakce patří design zaměřený na uživatele, osobnost, design zaměřený na činnost, návrh založený na scénáři a návrh odolnosti.
- Běžné postupy pro specifikaci softwaru rozhraní zahrnují případy použití a omezují vynucování interakčními protokoly (zamezuje se chybám použití).
- Běžné postupy pro vytváření prototypů jsou založeny na knihovnách prvků rozhraní (ovládací prvky, dekorace atd.).

Zásady kvality:

- Jasnost: Rozhraní se vyhne nejednoznačnosti tím, že vše vyjasní prostřednictvím jazyka, toku, hierarchie a metafor pro vizuální prvky.
- Stručnost: Přehnaně detailní vysvětlení a označení všeho, co obsahuje rozraní není správná cesta. To vede k nepřehlednosti rozhraní, protože je na obrazovce současně příliš mnoho věcí najednou. Pokud je na obrazovce příliš mnoho prvků, je obtížné najít požadovaný prvek, a proto je používání rozhraní únavné. Skutečnou výzvou při vytváření rozhraní je zajistit, aby bylo stručné a jasné zároveň.
- Známost: I když někdo používá rozhraní poprvé, určité prvky mohou být stále známé.

- Schopnost reagovat: Dobré rozhraní by nemělo být pomalé. To znamená, že rozhraní by mělo poskytovat uživateli dobrou zpětnou vazbu o tom, co se děje a zda se vstup uživatele úspěšně zpracovává.
- Konzistence: Udržování konzistentního rozhraní napříč vaší aplikací je důležité, protože umožňuje uživatelům rozpoznat vzorce používání.
- Efektivita: Čas jsou peníze a perfektní rozhraní by mělo zvýšit produktivitu uživatele pomocí zkratek a dobrého designu.
- Shovívavost: Dobré rozhraní by nemělo trestat uživatele za jejich chyby, ale mělo by místo toho poskytnout prostředky k jejich nápravě.

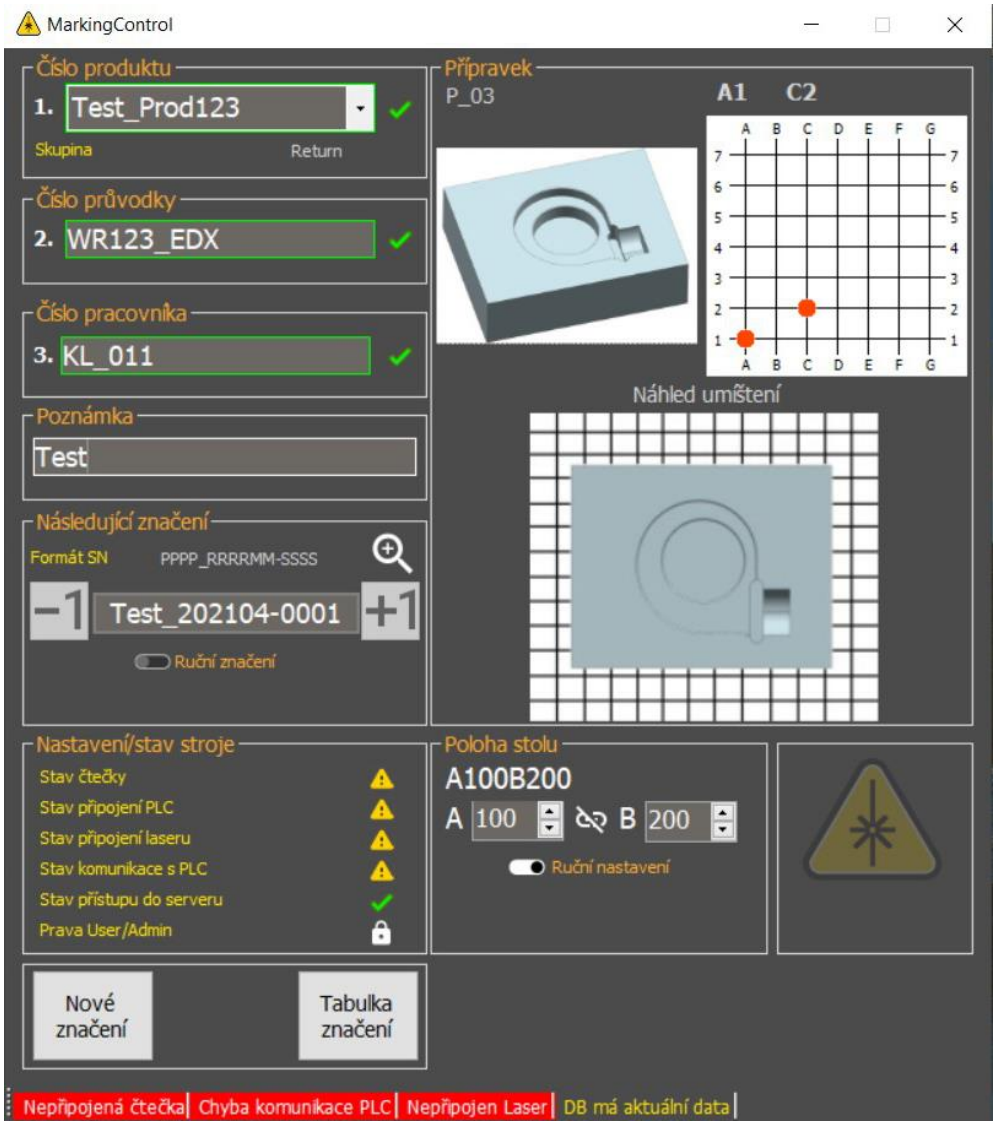
Kapitoly 2.1 a 2.2 byly zpracovány s použitím materiálů [1, 2, 8].

2.3 Vytvoření uživatelského rozhraní pro aplikaci

V rámci procesu vytvoření uživatelského rozhraní probíhala jednání se zákazníkem. Na základě vznikajících požadavků zákazníka na to, jaké elementy musí mít uživatelské rozhraní aplikace, autor práce postupně vylepšoval program *MarkingControl*. Tyto požadavky vznikaly zpětnou vazbou během použití této aplikace při procesu značení plastových dílů. Nejdůležitějším bylo dodržovat princip *„jediné tlačítko“*. To znamená, že aplikace *MarkingControl* provádí všechny operace pro načítání dat a pro následné značení dílů. Pracovník ovládá aplikaci pouze pomocí čtečky čárových kódů. Započetí procesu značení je z důvodu bezpečnosti provedeno s použitím dvou bezpečnostních tlačítek umístěných na desce strojního zařízeni.

Princip načítání dat do aplikaci spočívá na třech základních bodech. Jsou to: číslo produktu, číslo průvodky a číslo pracovníka. Číslo produktu je nejzákladnější údaj. Z čísla produktu aplikace generuje nové sériové číslo pro značení, otevírá správný soubor pro značení, nastavuje správnou pozici pojezdového stolu a posílá tuto pozici na určitý požadavek od PLC. Při náhodné chybě načítání dat nebo špatném manuálním nastavení musí aplikace informovat pracovníka. Tudíž aplikace *MarkingControl* musí požádat, aby pracovník tuto chybu opravil a pak pokračoval v procesu značení. K nejčastějším chybám patří načtení neexistujícího čísla produktu například v případě značení nového produktu, který ještě nemá šablonu pro značení uloženou do databáze. V takovém případě pracovník dostane hlášení, že takový produkt neexistuje a bude mít možnost vytvořit novou šablonu pro tento produkt. Pro novou šablonu pracovník uvede všechny potřebné informace – jsou to: číslo produktu, formát sériového čísla, pozice pojezdového stolu a speciální soubor pro aplikaci *MaxMarking*.

Při spolupráci se zákazníkem bylo postupně vyvinuto ergonomické rozhraní, které odpovídá požadavkům zákazníka a splňuje, základní požadavek *„princip jediné tlačítko“*. Kromě toho uživatelské rozhraní má snadno srozumitelný interface, dobře čitelná tlačítka – ikonky, které autor částečné nakreslil sám. Pro zbytek byla požita veřejně dostupná bezplatná knihovna [9]. V aplikaci je použitá sada firemních barev například: šedá (*75; 75; 75*), (*107; 103; 99*), žlutá (*255; 228; 0*), oranžová (*255; 170; 50*). Jako písmo bylo použito *Tahoma* (obrázky 5, 6).



Obrázek 5: Uživatelské rozhraní MarkingControl

Obrázek 6: Uživatelské rozhraní MarkingControl – kompaktní tvar

3 Princip fungování aplikace

Tato část diplomové práce se zabývá popisem základních principů fungování aplikací a rešeršemi spolehlivého komunikačního algoritmu PLC – PC. Zabývá se algoritmem, který umožňuje vyhodnocovat datové vstupy a zobrazovat aktuální stav připojených zařízení. Tato část diplomové práce se zabývá algoritmem, který má možnost zabezpečeně přistupovat do databáze běžící na podnikovém serveru, a podle potřeby umožnit tvorbu nových šablon pro značení produktů. Zabývá se rešerší spolehlivého, efektivního algoritmu pro ovládání aplikace s minimálním vstupem pracovníka do programu s možností kontroly dat načtených do aplikace, a v případě potřeby, možností opravy zvolených dat. Zabývá se rešerší algoritmu pro třídění dat podle zvolených požadavků pracovníka, pro které pracovník má možnost zobrazit statistické informace.

3.1 Rešerše spolehlivého komunikačního algoritmu PLC – PC

Jedna z důležitých, základních úloh byla vytvořit spolehlivý komunikační algoritmus pro spojení PLC – PC. Při spuštění celého stroje PLC pošle přes komunikační protokol *RS-232* požadavek speciální symbol *„r“* na nastavení polohy pojezdového stolu. Uživatel s pomocí čtečky čárových kódů načte číslo produktu, číslo průvodky a číslo pracovníka. Aplikace *MarkingControl* běžící na PC pracovníka je připojena k databáze, která obsahuje tabulky *LaserCFG* a *LaserDB* (schéma 1) a načte všechny potřebné informace včetně polohy stolu (schéma 2). Aplikace *MarkingControl* na dotaz PLC symbol *„r“* odpoví polohou stolu ve tvaru *AXXX*, kde *XXX* je poloha pro ukládání přípravku do pojezdového stolu a *BYYY*, kde *YYY* je poloha pro značení dílů. Polohy nastavení mohou mít stejnou pozici (např. *A222B222*), tak i různé (např. *A100B200*). Existuje omezení pozic nastavení pojezdového stolu. Maximální možná pozice je *512*, kde poslední čísla od *502* jsou servisní polohy. Možnost nastavení polohy začíná od 0 a to je nulová poloha. Poloha stolu nemůže být záporná. Všechna omezení nastavení poloh stolu jsou realizována v aplikaci *MarkingControl* podle toho, jak byly do aplikace načteny z databáze všechny potřebné informace. PC odpoví PLC polohou stolu. PLC pak musí poslat speciální symbol *„s“*, který představuje dotaz, zda je vše připraveno pro značení. Pokud uživatel načetl číslo průvodky, číslo pracovníka, tak PC odpoví speciálním symbolem *„S“*. Podle toho bude PLC čekat na stisk speciálních bezpečnostních startovacích tlačítek umístěných na desce stroje. Když pracovník přiloží prsty na tlačítka, PLC začne proces laserového značení. Po ukončení značení PLC pošle do PC speciální symbol *„x“* – to je konec značení. PC po obdržení zprávy *„x“* uloží informace o označeném dílu do bufferu. Pokud je přístup do databáze umožněn, PC zapíše data do databáze, přičemž to udělá v jiném vlákně. Jakmile proběhne zápis dat do bufferu, aplikace provede inkrementaci sériového čísla, a odpoví PLC speciálním symbolem *„X“*. Pak se cyklus opakuje.

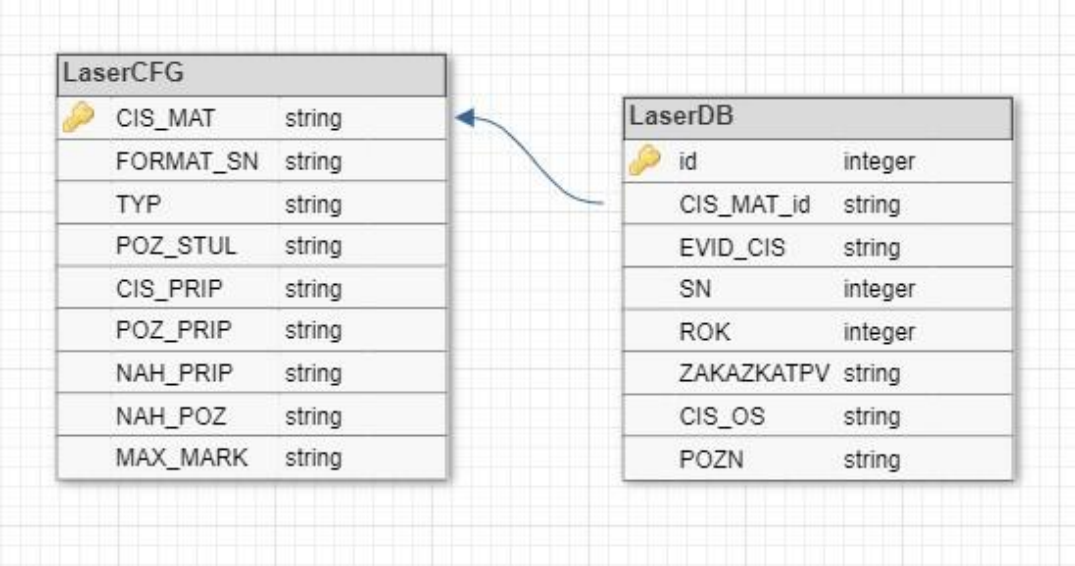


Schéma 1: Tabulky LaserCFG a LaserDB

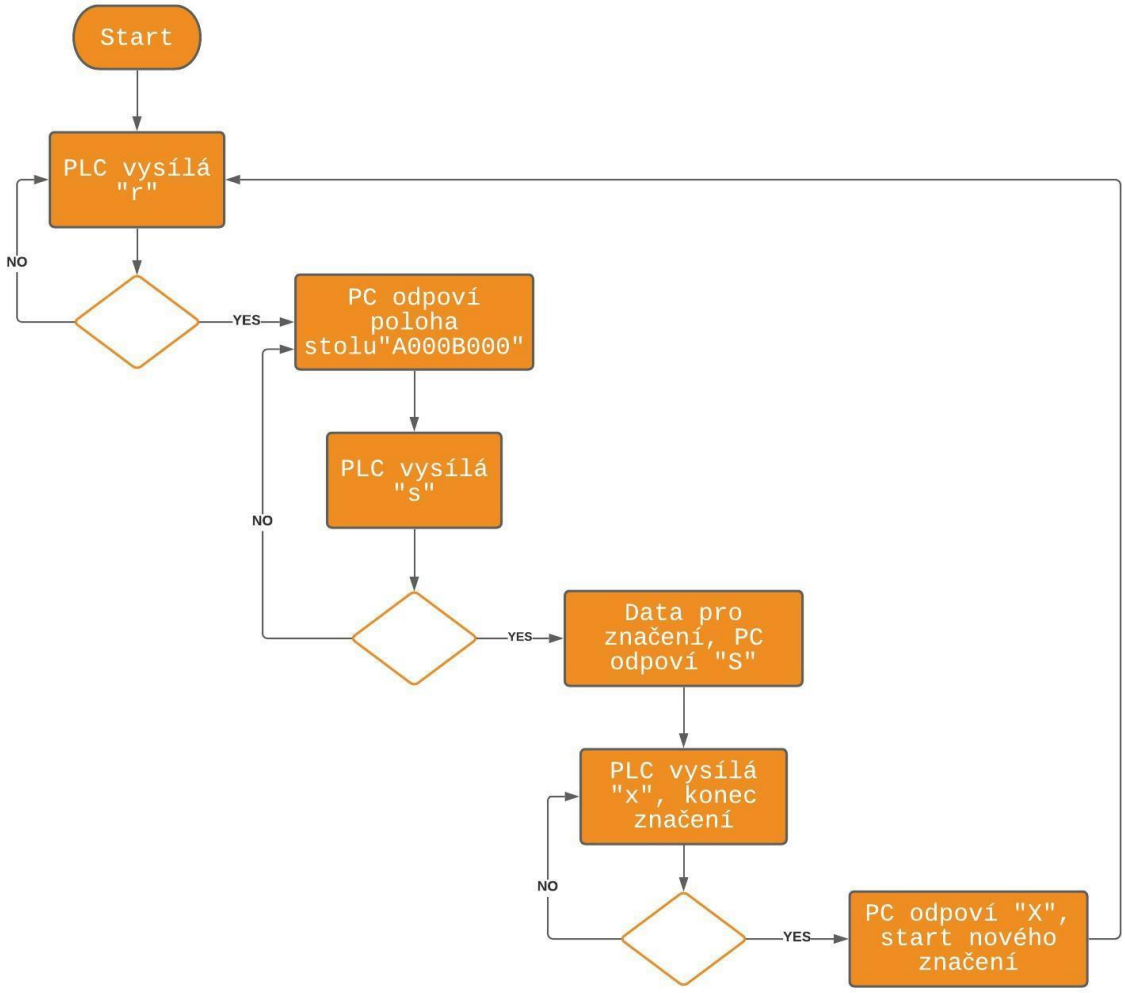
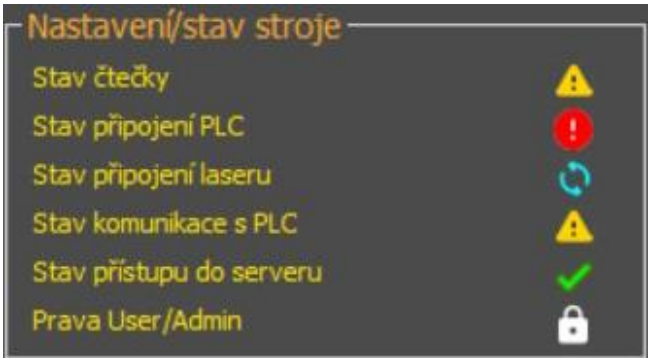


Schéma 2: Cyklus komunikace mezi PLC a PC

Tato komunikace byla požadována na straně zákazníka. Cílem bylo vyvinout algoritmus pro komunikaci PC a PLC spolehlivým způsobem, odpovídajícím požadavkům zákazníka. Princip opakování speciálního symbolu slouží pro zabezpečení komunikace proti chybám a poruchám. Zpráva bude opakována, dokud nebude přijata. Když je zpráva přijata, algoritmus pokračuje v dalších krocích cyklu. Popis tohoto algoritmu vznikl v průběhu spolupráce s ostatními členy týmu společnosti zákazníka. Pro autora to znamenalo prokázat schopnost pracovat v týmu a vytvořit spolehlivý a funkční algoritmus. Tento algoritmus je základem aplikace *MarkingControl*.

3.2 Vyhodnocení datových vstupů a zobrazení aktuálního stavu připojených zařízení



Obrázek 7: Uživatelské rozhraní MarkingControl – zobrazení aktuálního stavu připojených zařízení

Jedna z úloh diplomové práce je vytvořit algoritmus, který bude vyhodnocovat datové vstupy a bude zobrazovat aktuální stavy připojeních zařízení. K takovým zařízením patří: čtečka čárových kódů, počítač, který ovládá laser, PLC, Wi-Fi adaptér pro připojení do podnikové sítě.

3.2.1 Připojení a komunikace se čtečkou čárových kódů

Kvůli tomu, že zákazník měl určitý požadavek na princip ovládání aplikace, vznikla potřeba najít způsob načítání informací do textových polí jak čtečkou čárových kódů, tak i ručně s pomocí klávesnicí podle potřeby pracovníka. Úloha, kterou autor měl řešit, spočívala vtom, že program musí zjistit, kdy je načtení dat ukončeno a přeskočit na další textové pole automaticky. Tato úloha měla být řešena jak v případě klávesnice, tak i čtečky čárových kódů. Autor provedl analýzu parametrů různých čteček čárových kódu ohledně schopnosti odeslat nějaký speciální symbol na konci každého čárového kódu načteného čtečkou. Bylo zjištěno, že čtečka čárových kódů *Eclipse MS5145 LS* od společnosti *Honeywell* má možnost na konci každého načteného čárového kódu odeslat speciální symbol *„Enter“*. Tím pádem byl zvolen tento speciální symbol, který měl znamenat ukončení načítání dat do textového pole, a to i v případě klávesnice. V případě, že bude textové pole opuštěno ručně s pomocí myši, bude odeslán speciální symbol *„Enter“*.

Stav čtečky je v aplikaci zobrazen pomocí sady tří ikonek „V pořádku“, „Pozor“ a „Chyba“.

Stav připojení čtečky je aktualizován v reálném čase. V případě odpojení čtečky uvidí pracovník aktuální chybový stav.

3.2.2 Připojení a komunikace s PLC

PLC je připojen do aplikace pomocí kabelu a komunikačního protokolu *RS-232*. Tento komunikační protokol byl zvolen podle požadavku zákazníka. Tento protokol umožňuje propojení a vzájemnou sériovou komunikaci dvou zařízení. Na straně PC je použit adaptér USB. Na straně PLC je použit *9pinový D-Sub konektor DE-9 M*. Komunikace mezi PLC a PC probíhá s pomocí algoritmu uvedeného v odstavci 3.1. Pro zobrazení aktuálního stavu podle požadavku byly použity dvě ikonky. Jeden ze stavů je stav připojení k PLC. Tento stav zobrazí jen připojení PLC k PC. Ten stav neumožňuje kontrolovat, jestli probíhá nějaká komunikace mezi PC a PLC. Druhá ikonka naopak zobrazí stav komunikace PC a PLC. Pokud PLC vysílá správný speciální symbol a PC ho přijímá, bude zobrazen stav – v pořádku. Pokud dojde k nějaké chybě, PC dostane nějaký jiný symbol nebo nedostane žádný a bude zobrazena ikonka chybového stavu.

3.2.3 Stav připojení laseru

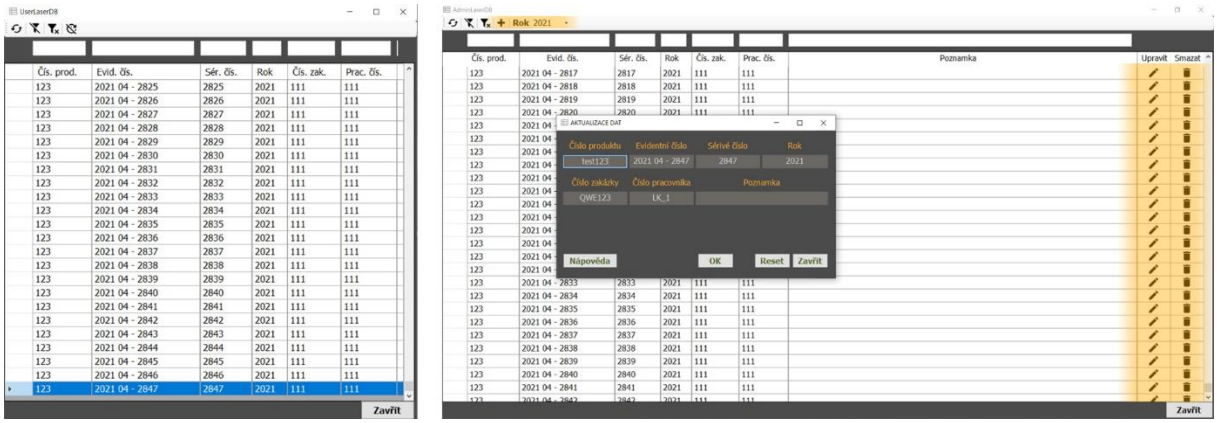
Mezi počítačem, do kterého byla nainstalována aplikace *MaxMarking*, a počítačem, který řídí laser, byla zajištěna komunikace s pomocí protokolu *EtherNet/IP*. Tato komunikace také musí být kontrolována s pomocí aplikace *MarkingControl*. Program kontroluje, jestli je připojení zajištěno. Pro zobrazování stavu je použita stejná sada ikonek jako pro komunikaci s PLC.

3.2.4 Stav přístupu k serveru

Komunikace se serverem probíhá kvůli načítání a ukládání dat do databáze. Aplikace *MarkingControl* musí zjistit pomocí pingu IP adresy SQL serveru, zda je server přístupný nebo není. V případě, že dojde k odpojení od serveru během značení, aplikace to zjistí a bude ukládat data do speciálního bufferu. Tento buffer je vytvořen pro to, aby zabránil ztrátě dat. Tak bude proces značení laserem probíhat dál, dokud jej pracovník nezastaví. Tento algoritmus povolí dokončeni zahájeného značení pro jednotlivé druhy produktu. Stav komunikace se serverem dokáže pracovník zjistit s pomocí standardní sady ikonek.

3.2.5 Stav User/Admin

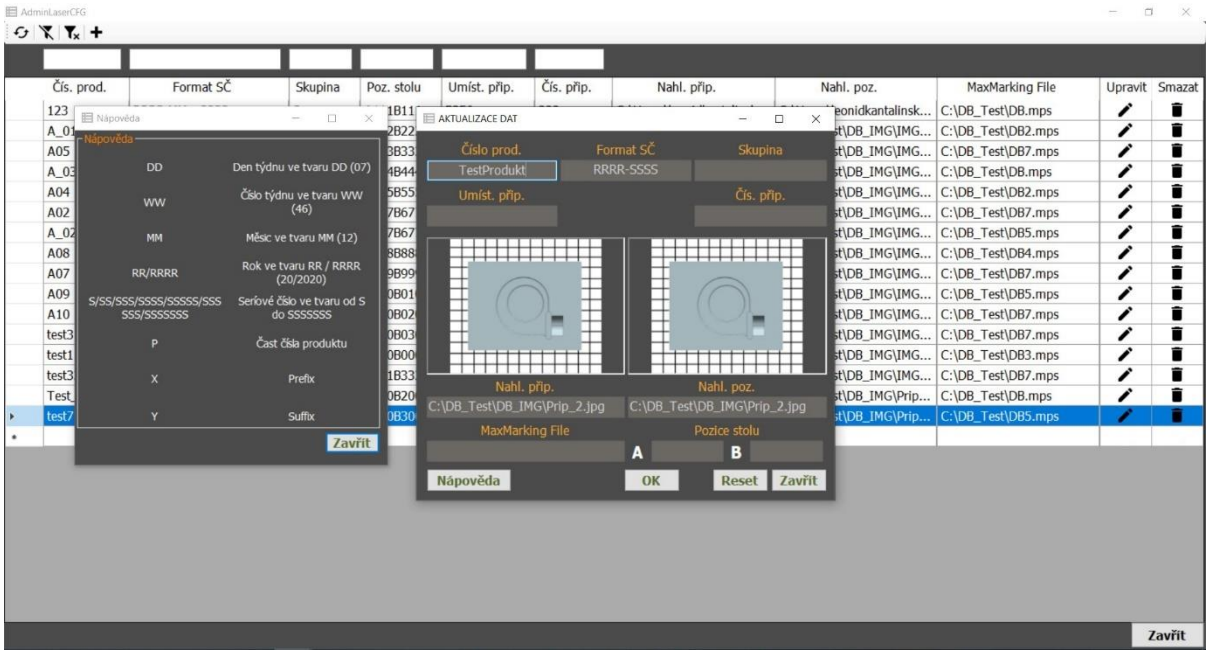
Stav User/Admin upozorňuje pracovníka na to, jaký přístup do aplikace má uživatel. Běžně je pro každého pracovníka povolen přistup User. Přistup Admin umožňuje změnu existujících a vytvoření nových šablon pro tabulky *LaserCFG* a *LaserDB*. Proto je z důvodu bezpečnosti přístup Admin zaheslovaný. Rozdíly mezi *UserLaserDB* a *AdminLaserDB* jsou možnost aktualizace a vytvoření nových řetězců (obrázek 8). Přístup Admin povolí provést změnu existujících a vytvoření nových řetězců v tabulkách *LaserDB* a *LaserCFG*. Přístup Admin umožňuje spouštět režim manuálního značení plastových dílů.



Obrázek 8: Uživatelské rozhraní MarkingControl – rozdíly mezi grafickým rozhraním User a Admin v části prohlížení a opravy dat. Oranžovou barvou jsou označeny ovládací prvky uživatele Admin

3.2.6 Tvorba nových šablon

Pro uživatele s přístupem Admin existuje možnost vytvořit a uložit novou šablonu pro značení plastových dílů. Při vytvoření šablony uživatel musí zadat několik povinných údajů. K těmto údajům patří: číslo produktu, formát sériového čísla, pozice stolu, soubor pro značení pro aplikaci *MaxMarking*. Ostatní položky je možné změnit během aktualizací. K takovým patří: skupina, umístění přípravku, číslo přípravku, náhled přípravku a náhled umístění. Pro pohodlnost uživatele existuje tlačítko *„Nápověda“*, které zobrazí všechny potřebné informace pro vytvoření nových šablon (obrázek 9).

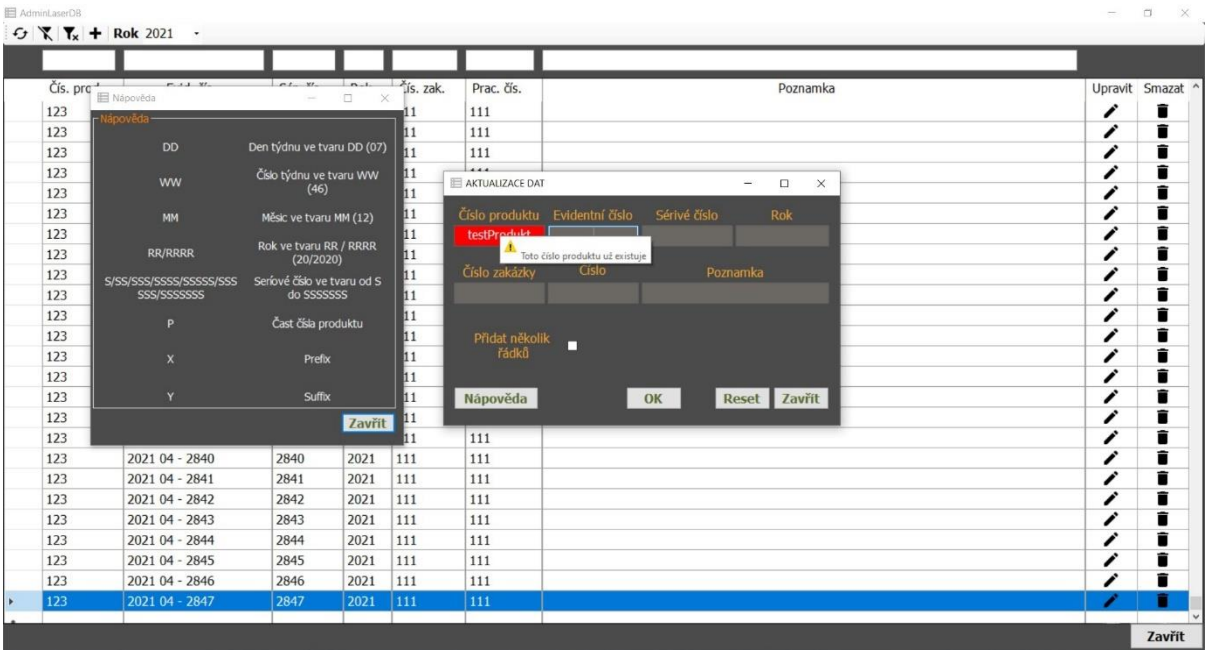


Obrázek 9: Uživatelské rozhraní MarkingControl – vytvoření nové šablony pro značení

Pro uživatele s přístupem Admin tedy existuje možnost vytvořit nový řetězec nebo změnit řetězec již zapsaný do tabulky *LaserDB*. Při vytvoření nového řetězce, pro který ještě nebylo

uvedeno v databázi číslo produktu, bude uživatel na tuto skutečnost upozorněn hláškou. Pak bude textové pole „Číslo produktu“ rozsvíceno červeně. Aplikace při zadání čísla produktu zkontroluje v databázi, jestli uvedený produkt již byl někdy označen (obrázek 10).

Další požadavek, který měl autor splnit, je možnost přidání několika již označeních produktů do tabulky *LaserDB*. Tato možnost byla realizovaná s pomocí zaškrtávacího políčka *„Přidat několik řádků“*. Když uživatel zvolí tuto možnost, objeví se textové pole, kde uživatel může vybrat: zadat počet řádků či zvolit do jakého sériového čísla uživatel chce přidat řádky.



Obrázek 10: Uživatelské rozhraní MarkingControl – vytvoření nového řetězce v tabulce LaserDB pro generovaní sériového čísla

Pro pohodlí uživatele byla vytvořena možnost načítání dat pro již existující produkt do zadávacího okna. Když uživatel zvolí číslo produktu, který již existuje, aplikace načte všechna data do zadávacího okna, uživatel má možnost pro tento produkt něco změnit a pak údaje uložit do databáze.

3.3 Algoritmus ovládající aplikaci s minimálním vstupem pracovníka

Jedna ze základních úloh diplomové práce a zároveň požadavek zákazníka byla rešerše spolehlivého algoritmu, který by umožnil ovládání aplikace s minimálním vstupem pracovníka. Tento algoritmus má v praxi určité výhody. Například algoritmus urychluje proces značení dílů, zvyšuje spolehlivost procesu značení a automatizuje proces evidování již označených dílů. Tento algoritmus je tedy část systému, který zvyšuje bezpečnost pracovníka.

3.3.1 Princip fungování algoritmu

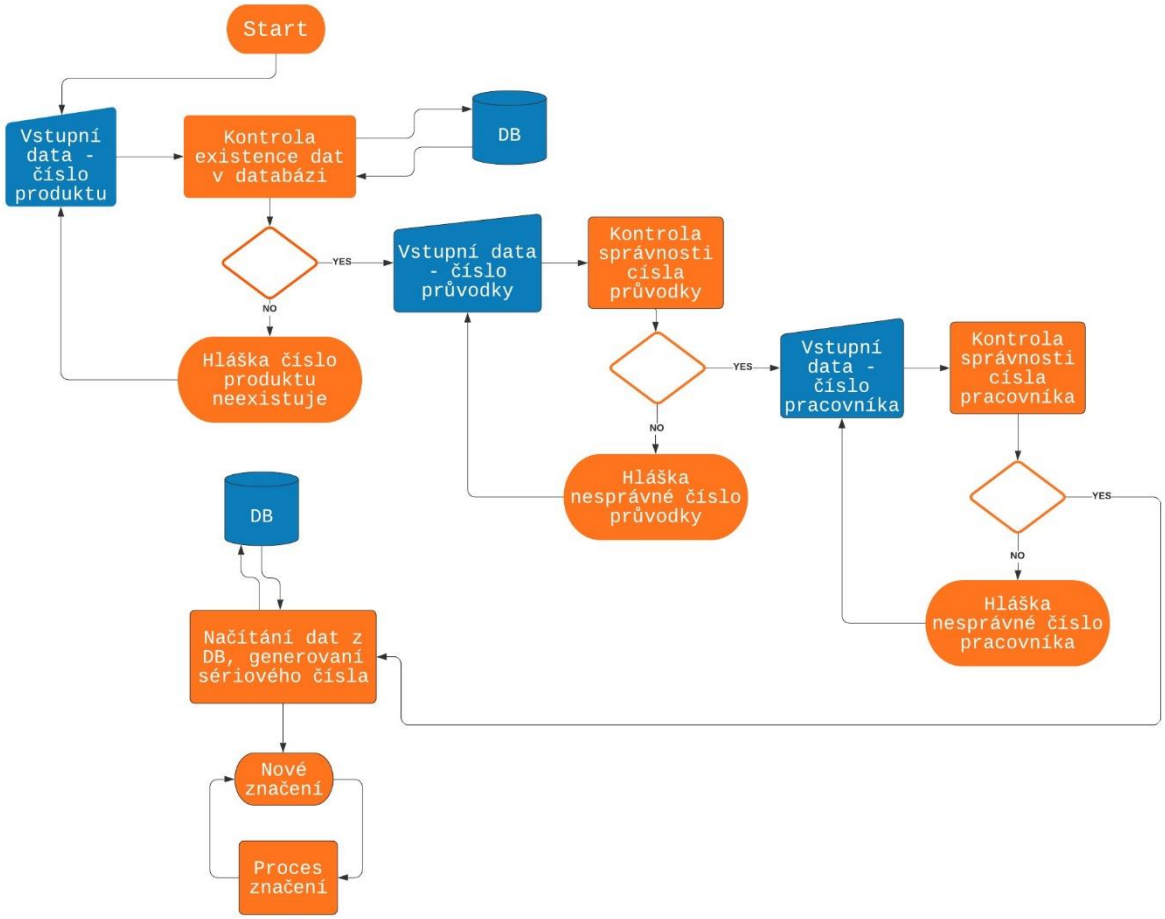


Schéma 3: Algoritmus načítání dat do aplikace

Princip fungovaní algoritmu spočívá v načítání (čtečkou čárových kódů), třech základních údajů: čísla produktu, čísla průvodky a čísla pracovníka. Generování a načítání dat z databáze je zajištěno načtením existujícího čísla produktu. Číslo produktu je základem pro generování sériového čísla pro značení dílů, nastavení polohy pojezdového stolu a zobrazení potřebné grafické informace pro značení. Pro číslo průvodky a číslo pracovníka probíhá kontrola správnosti podle přednastavených parametrů. Po ukončení značení dílu budou všechna data o již označeném kusu uložena do tabulky *LaserDB* (schéma 3).

Po označení prvního plastového dílu začíná fungovat další část ovládacího algoritmu. Tento algoritmus inkrementuje o jedna sériové číslo pro značení. Na konci procesu značení proběhne zápis do tabulky *LaserDB*. Pokud dojde k odpojení od serveru a možnost zápisu do databáze bude zastavena, budou data o již označených dílech shromažďována do bezpečnostního bufferu, který spojuje aplikaci a databáze. Když dojde k připojení k databáze data z bufferu budou přepsána do tabulky *LaserDB* (schéma 4).

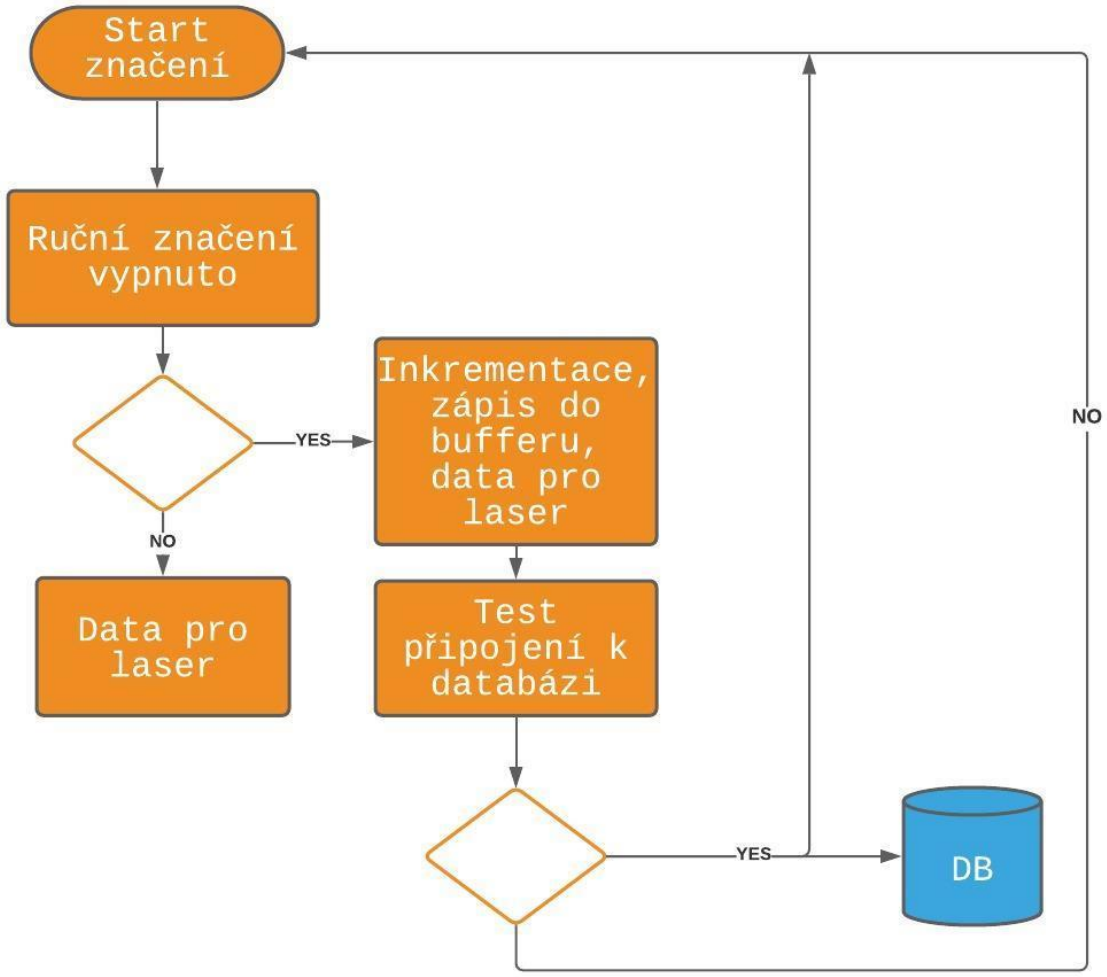
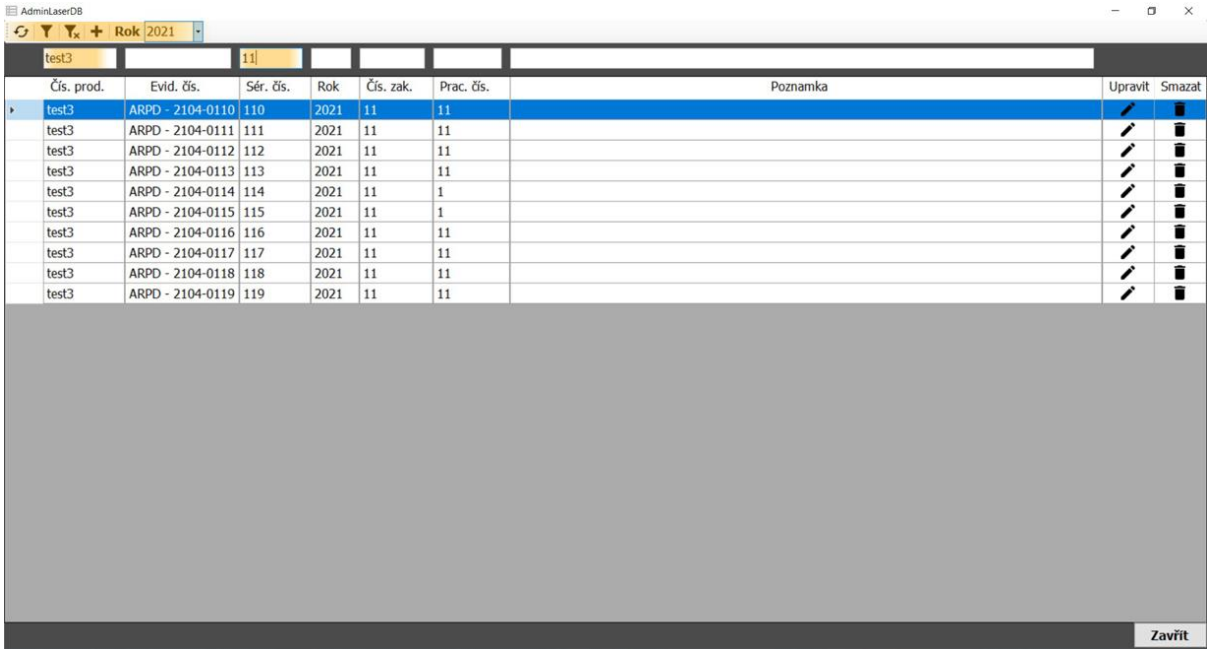


Schéma 4: Algoritmus inkrementace a zápisu do databáze

V případě potřeby má tedy pracovník s přístupem Admin možnost provést ruční značení. V takovém režimu má pracovník možnost manuálně nastavit sériové číslo. Jako další možnost byl realizován zápis produktu s manuálním zapsáním sériového čísla do databáze.

3.3.2 Princip fungování třídění dat podle požadavku pracovníka

Možnost třídění zobrazovaných dat je zajištěno za pomoci nastavení vyhledávacích filtrů. Při zadání dat do textového pole je filtr zapnut po stisku klávesy *Enter*. Filtraci je tak možno zapnout či vypnout pomocí příslušné ikonky (obrázek 11).



Obrázek 11: Uživatelské rozhraní MarkingControl – rozdíl mezi grafickým rozhraním User a Admin v částí prohlížení a opravovaní dat. Oranžovou barvou jsou označeny ovládací prvky uživatele Admin

Jako další možnost pro třídění zobrazovaných dat je seznam roků. Tento způsob třídění je možno použít s filtrem nebo je možno použít jen samostatné třídění podle zvoleného roku. Tento způsob je velmi užitečný při zobrazování velkého objemu dat.

Třídění dat s pomocí filtru je umožněno uživatelům s úrovní oprávnění User i Admin.

4. Praktická realizace aplikace

Tato část diplomové práce se zabývá praktickou realizací zadání. Praktická část se skládá z programování a pak testovaní vytvořené aplikace. Poslední odstavec praktické částí se zabývá popisem dosažených výsledků.

Vytvoření spolehlivé aplikace bylo hlavním cílem této diplomové práce. Tento proces byl rozdělen na několik etap:

- Definice problému.
- Analýza požadavků a vývoj algoritmu aplikace.
- Implementace.
- Testování aplikace.
- Zkoumaní výsledků.

Jedním z úkolu praktické části bylo zjistit požadavky zákazníka, projednat se zákazníkem tyto požadavky, promyslet různé situace, které mohou vzniknout na pracovišti během použití aplikace a navrhnout další možný vývoj aplikace. Druhým úkolem praktické části bylo vypracovat efektivní algoritmus aplikace, který bude odpovídat požadavkům zákazníka a zároveň bude realizovatelný s pomocí standardních knihoven *.NET*. Dalším úkolem praktické části byla realizace programu, zahájení testovaní a možné opravy kódu, které budou zjištěny během etapy testování.

Jako programovací jazyk byl zvolen jazyk *C#*, tento jazyk byl požadován zákazníkem. Jako vývojová platforma byla zvolena platforma *.NET* a programovací prostředí *Microsoft Visual Studio 2019*. Pro vývoj aplikace byly použity jen standardní knihovny, žádné speciální. To byl také požadavek firmy.

Během vývoje bylo vyřešeno několik dalších úloh jako například ergonomické rozhraní. Dále byla vyřešena možnost ovládání aplikace pouze jedním tlačítkem, (například čtečkou čárových kódů). Jedna ze základních úloh diplomové práce byla vytvořit efektivní algoritmus s minimálním vstupem pracovníka. Byl vytvořen rychlý a spolehlivý algoritmus pro načtení dat z databáze a ukládání dat do databáze. Pro potřeby uživatele byla vytvořena možnost ručního ovládání s pomocí klávesnice. V manuálním režimu byla vyřešena možnost zadání údajů do textového pole *„sériové číslo“* v libovolném tvaru. Byl realizován algoritmus kontroly správnosti ukládaných dat do databáze. Byla zajištěna bezpečnost ukládání dat do databáze běžící na podnikovém serveru.

Programování aplikace bylo rozděleno na několik částí:

- Vytvoření spolehlivého, efektivního algoritmu pro komunikaci PLC – PC.
- Vytvoření spolehlivého algoritmu pro komunikaci mezi PC a ostatními zařízeními.
- Vytvoření algoritmu pro načítání dat s pomocí ruční čtečky čárových kódů, vyhodnocení konečného speciálního symbolu.

- Vytvoření speciálního bufferu, jehož cílem, je zabezpečení dat proti ztrátě při komunikaci s databází během ukládání informací o nových označených dílech.
- Řešení grafického zobrazování dat načtených z databáze, vytvoření nových šablon, aktualizací již existujících dat.
- Vytvoření komunikace s programem *MaxMarking*.
- Vytvoření algoritmu pro logování informací o probíhající komunikaci mezi PLC a PC.

Konečným výsledkem celé diplomové práce je aplikace *MarkingControl*. Níže uvedený UML diagram (schéma 5) zobrazuje všechny základní třídy, metody a vztahy mezi nimi pro tuto aplikaci.

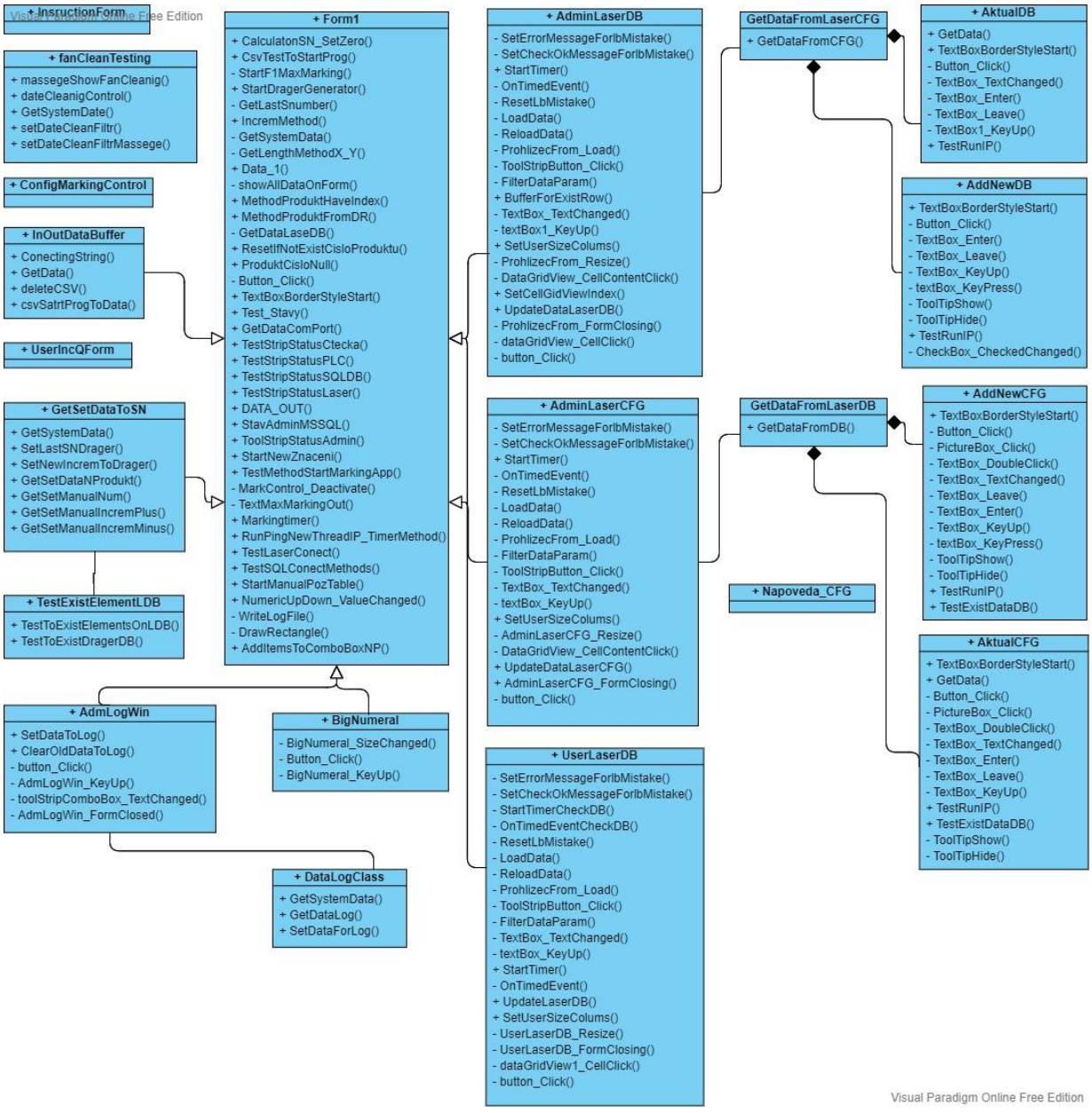


Schéma 5: UML diagram aplikace MarkingControl

4.1 Algoritmus pro komunikaci PLC – PC

Základní částí realizace kódu bylo vytvořit spolehlivý algoritmus pro komunikaci PLC a PC. Důležité bylo zabezpečit komunikaci proti poruchám či chybám. Pro řešení problému komunikace byl zvolen princip komunikace s pomocí speciálních symbolů. Opakování speciálního symbolu probíhá, pokud nebude vrácen odpovídající symbol [10, 12]. Byl implementován algoritmus:

```
try{
Invoke((MethodInvoker)delegate{
if (comunicStringPLC.Contains("r")){
      ComPortStart.Write(PLC_final_poloha);
                testStavComunicPLC = true;
      Console.WriteLine(string.Concat("Spojeni se PLC je v pořádku, poloha
      stolu je: " + PLC_final_poloha));
}
if (comunicStringPLC.Contains("s")){
      testStavComunicPLC = true;
     StartLaserZnaceni();
     ComPortStart.Write("S");
      Console.WriteLine("Udaje jsou v pořádku, začínam značit");
     ChangeLabMarkingTimer();
if (comunicStringPLC.Contains("x")){
     ComPortStart.Write("X");
     DATA_OUT();
     StartNewZnaceni();
     ComPortStart.DiscardInBuffer();
     ComPortStart.DiscardOutBuffer();
     ResetMarkingtimer();
}
}
```

Tato část kódu ukazuje princip pro komunikaci PLC – PC. Tento princip realizace algoritmu vyřešil problém spolehlivosti aplikace. Tento princip zajišťuje bezpečnost algoritmu proti různým chybám či poruchám, což je velmi důležité v podmínkách výroby.

4.2 Komunikace mezi PC a ostatními zařízeními

Aplikace *MarkingControl* kontroluje komunikaci mezi několika zařízeními připojenými k PC. Tato zařízení jsou: ruční čtečka čárových kódů, převodník *RS-232* pro komunikaci s PLC a síťová karta pro připojení laseru do počítače. Každá metoda určená pro kontrolu připojení zařízení běží ve svém vlákně.

4.2.1 Komunikace mezi PC a čtečkou čárových kódů

Pro připojení čtečky čárových kódů byl použit port USB. Čtečka čárových kódů je rozpoznávána počítačem jako klávesnice *USB HID (human interface device)*. Kontrola stavu čtečky čárových kódů (zda je čtečka připojena do počítače) probíhá s pomocí knihovny `System.Management` a třídy `ManagementObjectSearcher` [10, 12, 13, 14, 15]. Byl implementován algoritmus:

```
try{
string query = "SELECT * FROM Win32_Keyboard";
ManagementObjectSearcher searcher = new ManagementObjectSearcher(query);
```

```
foreach (ManagementObject item in searcher.Get()){
string deviceId = item["DeviceID"].ToString();

      if (deviceId.Contains("VID_0C2E&PID")){
     Invoke((MethodInvoker)delegate{
     pictureBox3.Image = Properties.Resources.Ok_1;
     });
     testDeviceId_Ctecka = false;
     testStatusCtecka = true;
     }
     if (testDeviceId_Ctecka == true){
     Invoke((MethodInvoker)delegate{
     pictureBox3.Image = Properties.Resources.Pozor;
     });
     testStatusCtecka = false;
     }
}
```

Komunikace mezi čtečkou čárových kódů a aplikací zajišťuje standardní knihovna `System.Windows.Forms` – třída `KeyEventArgs` s proměnnou `e` a metodou `KeyCode`. Po ukončení načítání čárového kódu čtečka pošle (speciální kontrolní symbol) klávesu `Enter`. Symbol `Enter` byl zvolen jako speciální symbol pro ukončení aktivity načítání dat. Pak ukončení aktivity a speciální symbol spouští další aktivity. Algoritmus byl implementován:

```
private void comboBoxSetProdukt_KeyUp(object sender, KeyEventArgs e)
{
      InputLanguage.CurrentInputLanguage =
      InputLanguage.FromCulture(new CultureInfo("en-GB"));
      comboBoxSetProdukt.Text = comboBoxSetProdukt.Text.Trim(' ');
      lbLastKey.Text = e.KeyCode.ToString();

if (e.KeyCode == Keys.Enter)
{
      comboBoxSetProdukt_Leave(sender, e);
}
}
```

V případě opuštění textového pole bude vygenerován speciální symbol `Enter`. Takovým způsobem je řešen problém načítání dat s pomocí klávesnice:

```
private void comboBoxSetProdukt_Leave(object sender, EventArgs e)
{
if (comboBoxSetProdukt.Items.Contains(comboBoxSetProdukt.Text))
{
      getNewNumProduct = comboBoxSetProdukt.Text;
      GetLastSN.GetAllFromLaserCFG();
      GenerationNewSN.IsGenerationSN();
      GetAllDate();
      txCisloPruvod.Focus();
}
if (!comboBoxSetProdukt.Items.Contains(comboBoxSetProdukt.Text)){
      pictureBox10.Image = Properties.Resources.Chyba;
}
}
```

4.2.2 Komunikace mezi PC a převodníkem RS-232

PLC je do počítače připojen s pomocí adaptéru. Pro komunikaci je použit protokol *RS-232*. Komunikace mezi PLC a PC je obousměrná. PLC posílá dotaz PC, ten na dotaz odpoví a čeká potvrzení od PLC o přijaté odpovědi.

Pro kontrolu připojení převodníku *RS-232* byla použita knihovna `System.Management` a třída `ManagementObjectSearcher`. Pro zjištění přítomnosti zařízení je používán kód:

```
public void GetDataComPort(){
      try{
      ManagementObjectSearcher ComPortName = new ManagementObjectSearcher
      ("root\\CIMV2", "SELECT * FROM " + "Win32_PnPEntity
      WHERE ClassGuid=\"{4d36e978-e325-11ce-bfc1-08002be10318}\"");
      try{
      foreach (ManagementObject queryObj in ComPortName.Get()){
      string deviceId = queryObj["DeviceID"].ToString();
            if (deviceId.Contains("USB\\VID_067B&PID_2303") ^
                  deviceId.Contains("FTDIBUS\\VID_0403+PID_6001")){
            }
```

Když aplikace zjistí, že zařízení je připojeno, otevřen COM port a začne probíhat komunikace mezi PLC a PC:

```
if (!ComPortStart.IsOpen){
      ComPortStart.PortName = NameComPort3;
      ComPortStart.BaudRate = 9600;
      ComPortStart.DataBits = 8;
      ComPortStart.Parity = Parity.None;
      ComPortStart.StopBits = StopBits.One;
      ComPortStart.Open();
      toolStripStatusLabelFirst = string.Concat("Com_Port: "
      + NameComPort3 + " is started");
      StatusToolStripStatusLabel_Text();
      testStatusPLC = true;
      }
```

V případě splnění podmínky je spouštěn komunikační algoritmus mezi PLC a PC.

4.2.3 Komunikace s laserem

Pro kontrolu komunikace mezi aplikace a počítačem řídící laser byl použit komunikační protokol EtherNet/IP. Laser má přednastavenou IP adresu. S pomocí ethernetu je připojen do počítače. Na straně počítače má nastavenou statickou adresu. Podle seznamu připojeních síťových zařízení je možno zjistit, zda je laser připojen

S pomocí knihovny `System.Net` a třídy `Dns` byl realizován kód:

```
public void TestLaserConnect(){
      try{
      int infoDNS_lenght = Dns.GetHostAddresses(Dns.GetHostName()).Length;
            for (int i = 0; i < infoDNS_lenght; i++){
            testIP_LaserConnect =
            Dns.GetHostAddresses(Dns.GetHostName())[i].ToString();
                  if (testIP_LaserConnect.Contains("192.168.170")){
```

```
                        testActiveConnectLaser = true;
                    }
            }
      }
      }
```

Tento algoritmus běží ve svém vlákně, které je spouštěno při spuštění programu.

4.3 Buffer

Pro zabezpečení procesu ukládání dat do databáze běžící na podnikovém serveru byl vytvořen speciální buffer. Tento buffer má v sobě všechny položky, které má tabulka *LaserDB*. Buffer představuje soubor ve formátu *CSV*. V případě odpojení od serveru bude soubor uložen do počítače, dokud připojení do databáze nebude zajištěno. V okamžiku spuštění aplikace algoritmus zkontroluje, zda buffer obsahuje alespoň jeden řádek, pokud ano, přepíše ten řádek do databáze a smaže ho z bufferu:

```
public void CsvTestToStartProg() {
      if (testIP_RUN == true) {
            if (testStatusSQLDB == true) {
            InOutDataBuffer inOutDataBuffer = new InOutDataBuffer();
            inOutDataBuffer.csvSatrtProgToData();
            InOutDataBuffertoolStripStatusLabel_Text();
            }
      }
      }
```

Tato část kódu kontroluje buffer při spuštění aplikace.

Metoda `inOutDataBuffer.GetData()` realizuje zápis dat do bufferu:

```
public void GetData(){
ConnectingString();
      Cis_Mat = MarkControl.Data_BUFF_Cis_Mat;
      EVID_CIS = MarkControl.Data_BUFF_EVID_CIS;
      SN = MarkControl.Data_BUFF_SN;
      ROK = MarkControl.Data_BUFF_ROK;
      ZAKAZKATPV = MarkControl.Data_BUFF_ZAKAZKATPV;
      CIS_OS = MarkControl.Data_BUFF_CIS_OS;
      POZN = MarkControl.Data_BUFF_POZN;

            string Cis_Mat_CSV = Cis_Mat;
            string EVID_CIS_CSV = EVID_CIS;
            string SN_CSV = SN.ToString();
            string ROK_CSV = ROK.ToString();
            string ZAKAZKATPV_CSV = ZAKAZKATPV;
            string CIS_OS_CSV = CIS_OS;
            string POZN_CSV = POZN;

            string csvRow = string.Format("{0},{1},{2},{3},{4},{5},{6}",
            Cis_Mat_CSV, EVID_CIS_CSV, SN_CSV, ROK_CSV, ZAKAZKATPV_CSV,
                  CIS_OS_CSV, POZN_CSV);

            var csv = new StringBuilder();
            var newCsvRows = csvRow;
            csv.AppendLine(newCsvRows);
            var listCsvRows = new List<string>();
```

```
            listCsvRows.Add(newCsvRows);
            try{
                File.AppendAllText(filePath, csv.ToString());
            }
      }
```

Pro zápis do souboru *CSV* byla použita knihovna `System.IO` a metoda `File.AppendAllText()`. Soubor *CSV* může obsahovat sedm sloupců a prakticky neomezený počet řádků. Po ukončení každého značení proběhne zápis dat do databáze pomocí sledujícího algoritmu:

```
try{
      connect_DB_OUT.Open();
      data_out_tb.Parameters.Add("@CIS_MAT", SqlDbType.Text).Value =
      Data_c_produktu;
      data_out_tb.Parameters.Add("@EVID_CIS", SqlDbType.Text).Value =
      SN_ROK;
      data_out_tb.Parameters.Add("@SN", SqlDbType.Int).Value = Data_SN;
      data_out_tb.Parameters.Add("@ROK", SqlDbType.Int).Value = Data_Rok;
      data_out_tb.Parameters.Add("@ZAKAZKATPV", SqlDbType.Text).Value =
      Data_Pruvodka;
      data_out_tb.Parameters.Add("@CIS_OS", SqlDbType.Text).Value =
      Data_Pracovnik;
      data_out_tb.Parameters.Add("@POZN", SqlDbType.Text).Value =
      Data_poznam;
      data_out_tb.ExecuteNonQuery();
      }
finally{
      connect_DB_OUT.Close();
      executeCSVtoData = "DB aktualizovaná";
}
```

Buffer je vždy spouštěn při spuštění aplikace ve svém vlákně. Tento způsob realizace umožňuje zajistit rychlou a bezpečnou komunikaci mezi databází a bufferem.

4.4 Zobrazování dat z databáze, vytvoření nových šablon, aktualizace dat

Velmi důležitou úlohou bylo vytvoření spolehlivého algoritmu, který umožní načítání potřebných informací z databáze. Tento proces musí proběhnout po ukončení načítání dat do textového pole *„Číslo produktu“* čtečkou čárových kódů. Po načtení čísla produktu musí algoritmus načíst několik základních informací, které umožní proces značení plastových dílů. Načtení dat probíhá ze dvou tabulek: *LaserCFG* a *LaserDB*. Z tabulky *LaserCFG* získává aplikace potřebné informace o produktu. Jsou to: formát sériového čísla, skupina produktu, poloha pojezdového stolu, umístění přípravku, číslo přípravku, náhled přípravku, náhled umístění a soubor pro aplikaci *MaxMarking*. Z tabulky *LaserDB* algoritmus zjistí poslední sériové číslo pro konkrétní produkt.

Pokud konkrétní produkt patří do existující skupiny produktů, bude sériové číslo tříděno podle roku a skupiny produktů. Například je možno nastavit jediné sériové číslo, které bude inkrementováno pro celou skupinu produktů, a tak evidovat tímto sériovým číslem celou skupinu.

Realizace algoritmu byla splněna pomocí knihoven `Data.SqlClient`, `SqlConnection` a tříd `SqlCommand` a `SqlDataReader` [11, 16].

Tato část kódu ukazuje načítání dat z tabulky *LaserCFG*:

```
public static void GetAllFromLaserCFG(){
if (MarkControl.testIP_RUN == true){
using (SqlConnection connectionCFG = new SqlConnection(connectStr)){
try{
connectionCFG.Open();
      string sqlExpressionCFG = "SELECT * FROM LaserCFG "
      + "WHERE CIS_MAT IN ('" + MarkControl.getNewNumProduct + "')";
SqlCommand commandCFG = new SqlCommand(sqlExpressionCFG, connectionCFG);
SqlDataReader readerCFG = commandCFG.ExecuteReader();
while (readerCFG.Read()){
      isNumProd = readerCFG.GetString(readerCFG.GetOrdinal("CIS_MAT"));
      isFormatSN = readerCFG.GetString(readerCFG.GetOrdinal("FORMAT_SN"));
      isGroup = readerCFG.GetString(readerCFG.GetOrdinal("TYP"));
      isPosTable = readerCFG.GetString(readerCFG.GetOrdinal("POZ_STUL"));
      isPosForms = readerCFG.GetString(readerCFG.GetOrdinal("CIS_PRIP"));
      isNumForms = readerCFG.GetString(readerCFG.GetOrdinal("POZ_PRIP"));
      isPicFormsSet = readerCFG.GetString(readerCFG.GetOrdinal("NAH_POZ"));
      isPicForms = readerCFG.GetString(readerCFG.GetOrdinal("NAH_PRIP"));
      isMaxMarkingFile =
      readerCFG.GetString(readerCFG.GetOrdinal("MAX_MARK"));
}
connectionCFG.Close();
GetAllGroupAndSNFromLaserCFG();
}
catch (Exception) { }
finally { connectionCFG.Close(); }
}
}
Else { return; }
}
```

Potom, co algoritmus načte data z tabulky *LaserCFG*, metoda `GetLastSNfromLaserDB()` spustí načítání posledního sériového čísla pro konkrétní produkt z tabulky *LaserDB*:

```
public static void GetLastSNfromLaserDB(){
      if (MarkControl.testIP_RUN == true){
      using (SqlConnection connection = new SqlConnection(connectStr)){
      try{
      connection.Open();
      for (int i = 0; i < isAllNumProFromGroup.Count; i++){
      sqlExpDBSearch = "SELECT * FROM LaserDB "
      + "WHERE CIS_MAT IN ('" + isAllNumProFromGroup[i] + "')"
      + "AND SN = (SELECT MAX(SN) FROM LaserDB WHERE ROK IN ('" +
      isAktualYear + "') "
      + "AND CIS_MAT = '" + isAllNumProFromGroup[i] + "')";
      if (isGroup.Contains("DR")){
      sqlExpDBSearch = "SELECT * FROM LaserDB "
      + "WHERE CIS_MAT IN ('" + isAllNumProFromGroup[i] + "')"
      + "AND EVID_CIS LIKE '" + isAktualDRPref + "%'";
}
SqlCommand commandDB = new SqlCommand(sqlExpDBSearch, connection);
SqlDataReader readerDB = commandDB.ExecuteReader();
while (readerDB.Read()){
isMaxSNFromNumProList.Add(readerDB.GetInt32(readerDB.GetOrdinal("SN")));
```

```
}
      readerDB.Close();
}
      if (isMaxSNFromNumProList.Count == 0) {isMaxSNFromNumProList.Add(0);}
      isLastSN = isMaxSNFromNumProList.Max();
      connection.Close();
      isMaxSNFromNumProList.Clear();
      isAllNumProFromGroup.Clear();
}
      catch (Exception){}
      finally { connection.Close(); }
}
}
      else {return;}
}
```

Tak bude zjištěno poslední sériové číslo pro zvolený produkt. Načítání posledního sériového čísla z tabulky *LaserDB* probíhá rychle (během několika desítek milisekund).

4.4.1 Grafické zobrazování dat z databáze

Pro fungovaní aplikace byly vytvořena databáze, která obsahuje dvě tabulky. Obě tabulky běží na podnikovém serveru. Mají názvy *LaserDB* a *LaserCFG*. Pro přístup a grafické zobrazení dat uloženích do těchto tabulek bylo vytvořeno grafické ovládací rozhraní. Z bezpečnostních důvodů má databáze dvě úrovně přístupu (Admin a User). Pro sledování již uložených informací o označených dílech stačí použít úroveň oprávnění User. Pro vytvoření nových šablon nebo aktualizaci existujících je třeba použít úroveň oprávnění Admin.

Pro zobrazení již označených dílů, uživatel s přístupem User stiskne na aplikačním rozhraní tlačítko – *„Tabulka značení“*, které spouští metodu `btUserDB_Click()`. Tato metoda má následující kód, který otevře okno *„UserLaserDB“*:

```
Thread thread = new Thread(() =>{
      Application.Run(new UserLaserDB());
      });
      thread.SetApartmentState(ApartmentState.STA);
      thread.Start();
      }
```

Pro grafické zobrazení informací lze použít objekt `DataGridView`. Knihovna `Data.SqlClient` slouží pro připojení k databázi a načítání dat do objektu `DataGridView`.

Pro tabulku (DataGridView) existuje možnost nastavení šířky sloupců podle jejich obsahu, takže uživatel může nastavit rozměr okna podle potřeby. Kvůli uživatelské přívětivosti po stisku tlačítka *„Tabulka značení“* bude okno *UserLaserDB* vždy otevřeno vedle hlavního okna aplikace a nebude jej překrývat.

4.4.2 Vytvoření nových šablon a aktualizace dat

Vytvoření nových šablon pro nové produkty bylo realizováno v rozhraní *AdminLaserCFG*. Po stisku tlačítka bude otevřeno okno, uživatel má možnost zadat do příslušných textových polí všechny potřebné informace pro vytvoření nové šablony. Druhé tlačítko dává možnost aktualizovat vybranou šablonu. Takže existuje možnost smazat vybranou šablonu. Pro

vytvoření a aktualizaci šablony existují speciální třídy: `AddNewCFG`, `AktualCFG`. Při vytvoření nové šablony bude zkontrolováno, zda číslo produktu již existuje v databázi. V případě, že takový produkt je již uložen v databázi, uživatel je na to upozorněn, ale má mít možnost produkt do databáze uložit. Tím pádem při načítání produktu při procesu značení bude zvolen novější produkt. Což je také užitečné, pokud uživatel chce mít v databázi několik různých šablon pro jeden produkt.

Pro pohodlnost uživatele je možno obrázky, které zobrazují přípravek a umístění přípravku, přidat do šablony a vybrat je ze složky. Pro obrázky jsou podle požadavku zákazníka omezeny formáty pro načítání. Pro načítání byla použita třída `OpenFileDialog`:

```
private void TextBox7_DoubleClick(object sender, EventArgs e){
      OpenFileDialog openFileDialog_NAH_PRIP = new OpenFileDialog{
      Filter = "Image files (*.jpg, *.png)|*.jpg; *.png;",
      InitialDirectory = stringFileDialog_NAH_PRIP
      };
      if (openFileDialog_NAH_PRIP.ShowDialog() == DialogResult.OK){
      textBox7.Text = openFileDialog_NAH_PRIP.FileName;
      pictureBox1.Image =
      Image.FromFile(Convert.ToString(openFileDialog_NAH_PRIP.FileName));
      }
}
```

Když uživatel bude chtít změnit nějakou šablonu, po stisku tlačítka mu bude zobrazeno okno, kam do příslušných textových polí již budou načteny informace o zvolené šabloně. Po změně nějakých údajů šablona nahradí starou a bude zapsána do databáze:

```
private void Button1_Click(object sender, EventArgs e){
      AdminLaserCFG adminLaserCFG = new AdminLaserCFG{
      cisloProduktuGetAktual = textBox1.Text,
      formatSN_GetAktual = textBox2.Text,
      typPoduktGetAktual = textBox3.Text,
      poziceStoluGetAktual = textBox4.Text,
      umisteniPripravkuGetAktual = textBox5.Text,
      cisloPripravkuGetAktual = textBox6.Text,
      nahledPripravkuGetAktual = textBox7.Text,
      nahledPoziceGetAktual = textBox8.Text,
      maxmarkingFileGetAktual = textBox9.Text
      };
adminLaserCFG.UpdateDataLaserCFG();
Close();
}
```

4.5 Komunikace s programem MaxMarking

Program *MaxMarking* je program, který ovládá laser. Program slouží pro vytvoření designu pro značení, tvaru a rozměru sériového čísla pro značení. Umožňuje nastavit vzhled značení na plastovém dílu. *MaxMarking* byl používán během výroby ve firmě již nějakou dobu před vznikem aplikace *MarkingControl*, přičemž samotný proces laserového značení a ukládání dat byl ruční. Pracovník musel ručně zapisovat všechny údaje do databáze. Pracovník musel ručně nastavit polohu stolu pro laserové značení. Kvůli tomu ve firmě vznikla potřeba uvedený proces automatizovat. S pomocí vytvořeného programu *MarkingControl* byl tento problém vyřešen.

Vytvořená aplikace *MarkingControl* dává možnost otevřít soubor nutný pro laserové značení dílu v programu *MaxMarking*. Tato možnost je zajištěna s pomocí tabulky *LaserCFG* a dynamické knihovny `user32.dll`. S pomocí třídy `DllImport` je možné připojit dynamickou knihovnu `user32.dll`. Třída, která povoluje spuštění zvoleného souboru, je `ProcessStartInfo`. Kód této části aplikace je:

```
[DllImport("user32.dll", EntryPoint = "FindWindow", SetLastError = true)]
static extern IntPtr FindWindowByCaptionMaxMarking
(IntPtr Set, string nameLaser);
private void START_LASER(){
IntPtr target_hwnd = FindWindowByCaptionMaxMarking(IntPtr.Zero,
"MaxMarking");
      if (target_hwnd == IntPtr.Zero){
                ProcessStartInfo start_info = new
                ProcessStartInfo(Data_laser){
                    FileName = Data_laser
                };
                proc.StartInfo = start_info;
                proc.Start();
                TestMethodStartMarkingApp();
            }
            else{
                return;
            }
        }
```

V aplikaci byla realizována možnost udržení programu *MarkingControl* v popředí. Tato situace vznikne, když bude zvolen požadovaný soubor pro značení laserem. V běžném provozu Windows převádí ovládací fokus na spuštěný program. Když bude spuštěn nový program, zaměření bude přemístěno na něj. Když bude spuštěn program *MarkingControl*, zaměření na něm bude zachováno. To je velmi důležitá operace, protože to odpovídá koncepci jednoho tlačítka. Program *MarkingControl* musí udržovat zaměření na sobě.

Jedna z vlastností fungovaní programu *MaxMarking* spočívá v tom, že pro načítání informací o sériovém čísle pro značení dílů je použit textový soubor. Právě ta vlastnost aplikace *MaxMarking* umožnila vzájemnou integraci programů *MaxMarking* a *MarkingControl*.

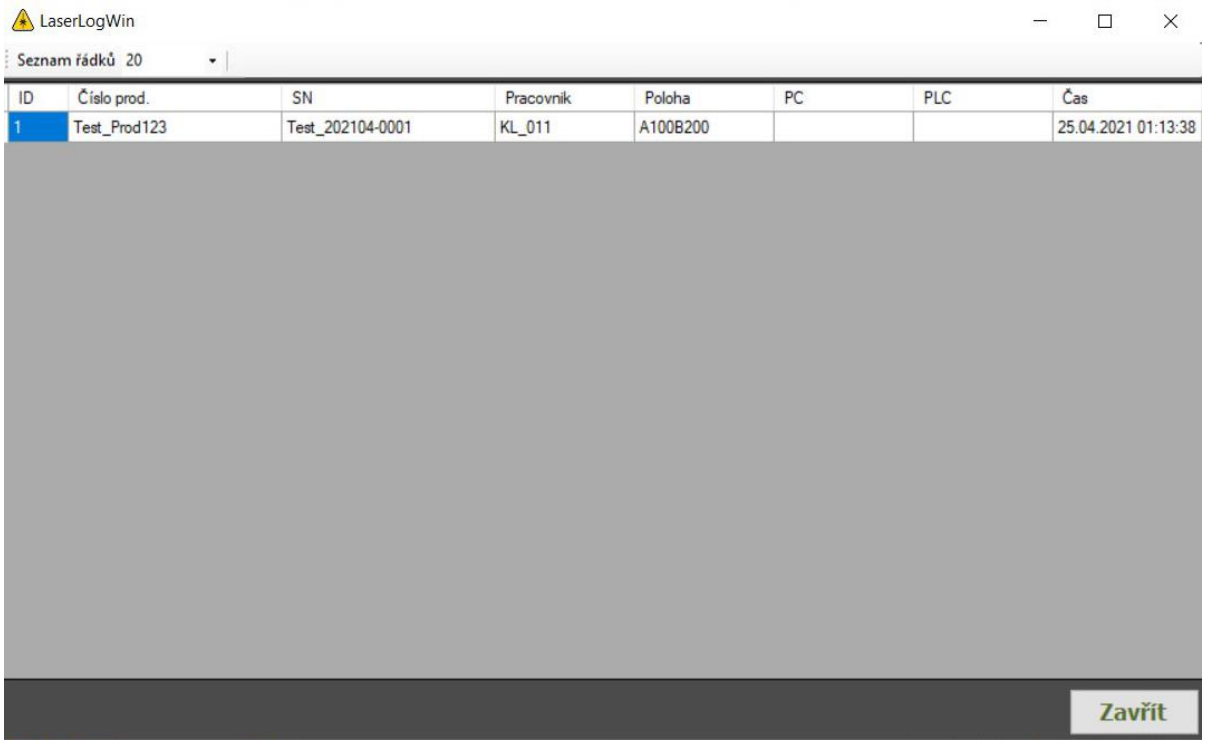
Aplikace *MarkingControl* vygeneruje sériové číslo pro značení a uloží jej do textového souboru. Pak při značení aplikace *MaxMarking* načte tento textový soubor do své šablony. Sériové číslo bude mít tvar podle nastaveného designu v aplikaci *MaxMarking*.

Pro ukládání sériového čísla do textového souboru byl použit kód:

```
private async void TextMaxMarkingOut(){
      string writePath = @"C:\Temp\Temp.txt";
      string text = SN_ROK;
            if (testStatusToogleManualMarking == true){text = txSN.Text;}
            try{
                using (StreamWriter sw = new StreamWriter
                  (writePath, false, Encoding.Default)){
                  await sw.WriteLineAsync(text);
                }
            }
            catch (Exception){}
        }
```

4.6 Rozhraní pro logování informací o komunikaci PLC – PC

Pro logování pobíhající komunikace mezi PC a PLC bylo vytvořeno speciální rozhraní (obrázek 12).



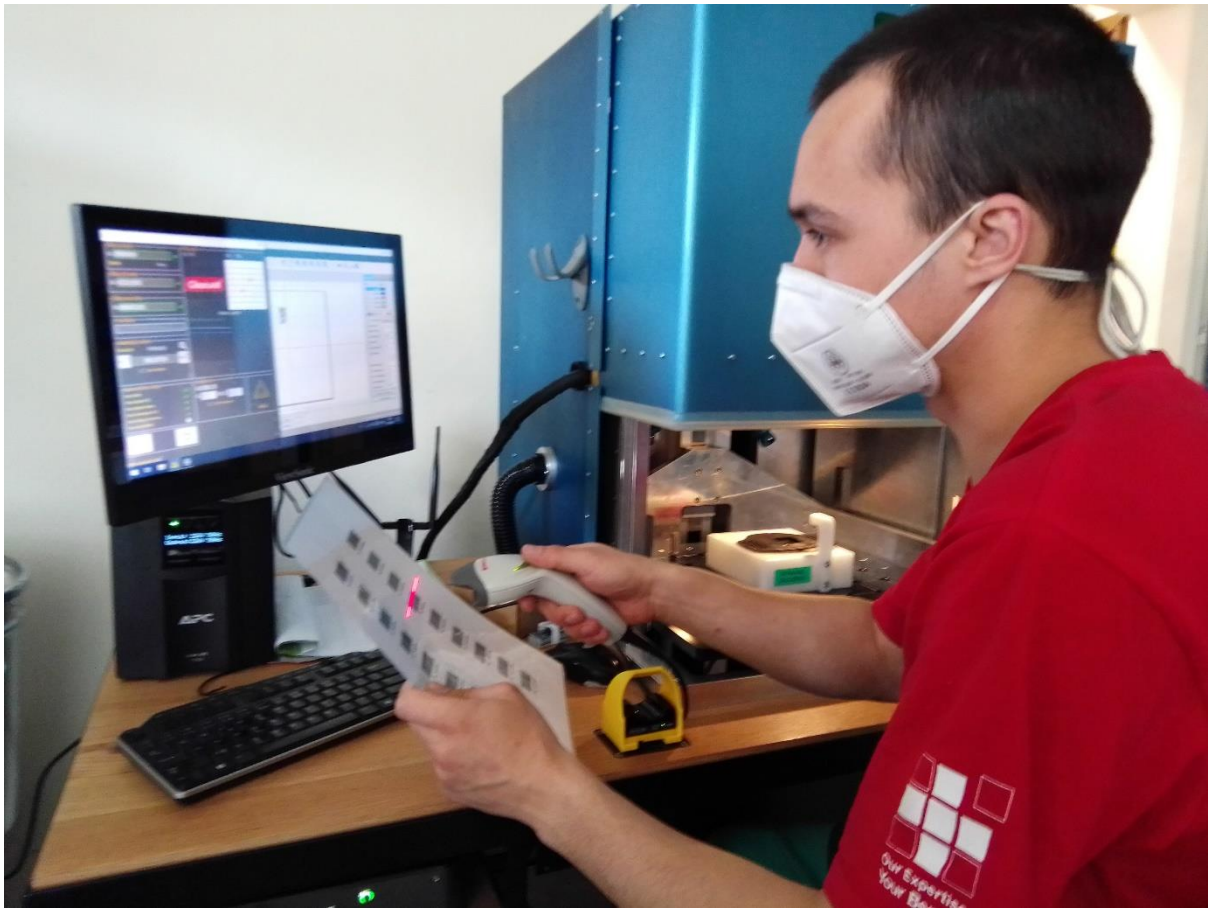
Obrázek 12: Uživatelské rozhraní MarkingControl – LaserLogWin

Toto rozhraní umožňuje zobrazit probíhající komunikaci mezi PLC a PC. Zobrazuje celý řádek, který obsahuje ID, číslo produktu, sériové číslo, číslo pracovníka, pozici stolu, zprávu PC, PLC a čas. Podle potřeby je možno zvolit, kolik řádků bude zobrazeno.

5. Dosažené výsledky

Hlavním výsledkem této diplomové práce je aplikace *MarkingControl*. Aplikace odpovídá požadavkům zákazníka a splňuje zadání diplomové práce. Hlavní dosažené výsledky jsou:

- Doba značení pro každý plastový díl klesla podle informaci zákazníka o 10–15 %.
- Evidence již označených dílů je nyní automatizovaná.
- Od nasazení programu bylo již označeno kolem 50 000 dílů.
- Pro přednastavení laseru pro značení je nyní používána šablona.



Obrázek 13: MarkingControl v provozu

Závěr

Cílem diplomové práce bylo vytvoření programu pro obsluhu výrobního pracoviště laserového značení dílů. Tento program byl vytvořen. Aplikace implementuje algoritmus, který umožňuje připravovat vstupní data pro pracoviště laserového značení plastových dílů, a to efektivně a s minimálním vstupem pracovníka. Tento algoritmus zajišťuje bezpečnou a spolehlivou komunikaci mezi PC a PLC, zabezpečuje spolehlivý přístup k databázi běžící na podnikovém serveru a splňuje podmínky pro ochranu před ztrátou dat kvůli odpojení sítě.

Během zpracování diplomové práce bylo řešeno velké množství rozmanitých problémů spojených s vytvořením aplikace. Nejdůležitější z nich se týkaly algoritmu pro komunikaci PLC – PC, komunikace s připojenými zařízeními, komunikace s aplikací *MaxMarking*; načítání, ukládání a třídění dat, realizace grafického rozhraní.

Během řešení těchto úloh autor zlepšil své znalosti v programovacím jazyce C# a o principech komunikace s Microsoft SQL Serverem. Autor implementoval algoritmus pro komunikaci mezi různými zařízeními, které byly použity v této práci.

Výsledkem této práce je splnění všech požadavků na vyvíjenou aplikaci *MarkingControl*. Tento program již funguje několik měsíců ve firmě zákazníka a do jisté míry usnadňuje proces laserového značení dílů. Během fungovaní aplikace ve výrobě již bylo označeno a evidováno kolem 50 tisíc plastových dílů. Pro tyto účely bylo vytvořeno kolem 30 šablon. Do budoucna je plánován další vývoj aplikace *MarkingControl*. Například je to vytvoření seznamu chybových kódů pro komunikace s PLC a jejich grafické zobrazení.

Seznam použité literatury

1. MAX, [online, cit. 17.10.2020]. Dostupné z: http://en.maxphotonics.com.

2. ACKEE. Vývoj a design úspěšných aplikací, [online, cit. 10.04.2021]. Dostupné z: https://www.ackee.cz/vyvoj-aplikaci/

3. SEMANTICSTUDIOS. Uživatelské rozhraní, [online, cit. 10.04.2021]. Dostupné z: https://semanticstudios.com/user_experience_design.

4. WHP TECHNIK. Čárové kódy. Značení výrobků. EAN 13, EAN 8, Code 128, Code 39. Barcode, [online, cit. 10.10.2020]. Dostupné z: https://www.whp.cz/carovy-kod-ean.html.

5. MOROVIA. KB10625 – GS1 128 specification, [online, cit. 10.10.2020]. Dostupné z: https://www.morovia.com/kb/GS1-128-Specification-10625.html.

6. HONEYWELL productivity and workflow solutions, [online, cit. 17.10.2020]. Dostupné z: https://www.honeywellaidc.com/en.

7. MEPAC CZ, [online, cit. 20.04.2021]. Dostupné z: https://www.profilaser.eu/popisovaci-lasery.

8. THE INTERNATIONAL ERGONOMICS ASSOCIATION (IEA): 2016, [online, cit. 15.11.2020] Dostupné z: http://www.iea.cc/whats/index.html.

9. GOOGLE FONTS. [online, cit. 03.02.2021]. Dostupné z: https://fonts.google.com.

10. WEISFELD Matt. The Object-Oriented Thought Process. Addison. Wesley, 2018. ISBN: 978-01-351-8166-6.

11. HERNANDEZ Michael James. Database Design for Mere Mortals. A Hands-On Guide to Relational Database Design. Grada, 2006. ISBN: 80-247-0900-7.

12. PECINOVSKÝ Rudolf. Návrhové vzory. Computer Press, 2015. ISBN: 978-80-251-1528-4.

13. MICROSOFT. Visual Studio product family documentation, [online, cit. 25.10.2020]. Dostupné z: https://docs.microsoft.com/en-us/visualstudio/?view=vs-2019.

14. WAGNER Bill. C# docs – get started, tutorials, reference, [online, cit. 15.10.2020]. Microsoft. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp.

15. MICROSOFT. C Sharp, [online, cit. 10.10.2020]. CC 2020. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-9.

16. METANIT. MS SQL Server 2017 и T-SQL, [online, cit. 03.11.2020]. Dostupné z: https://metanit.com/sql/sqlserver.

Příloha I – Zdrojové kódy

- Zdrojové kódy pro aplikace MarkingControl – MarkingControl.zip.