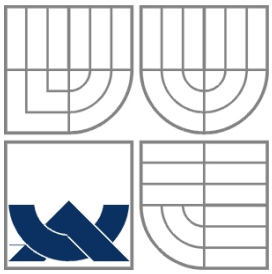
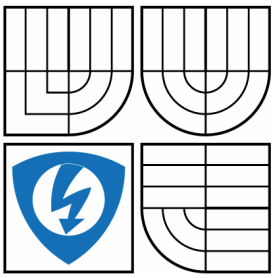


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

INTERNETOVÉ OVLÁDÁNÍ LABORATORNÍCH MODELŮ

INTERNET BASED CONTROL OF LABORATORY MODELS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Petr Dobrovolný

VEDOUCÍ PRÁCE
SUPERVISO

prof. Ing. František Zezulka, CSc.

BRNO 2011

Abstrakt

Toto zadání řeší návrh dvou modelů, které jsou určeny pro výuku na PLC a v budoucnu i na software Control Web. Oba modely mají být použity v systému LabLink, který umožňuje vzdálenou práci na laboratorních zadáních. První softwarový model simuluje pohyb vláčků po železnici v závislosti na řízení z PLC. Druhý model je fyzický model vrtačky. U obou modelů je řešeno ovládání a způsob provedení, který umožní provoz bez nutnosti fyzického zásahu.

Klíčová slova

LabLink, Control Web, Wago, CodeSys, OPC, ARM, stejnosměrný motor, vzdálená plocha, laboratorní model

Abstract

This award solves design of two models that are designed to teach on PLC and in the future on the Control Web. Both models should be used in LabLink system, which allows you to remotely working on laboratory tasks. The first software model simulates the movement of trains on rail in relation on control from the PLC. The second model is a physical model of drill. In both models are solid managing and type of construction, which allows operation without physical intervention.

Keywords

LabLink, Control Web, Wago, CodeSys, OPC, ARM, DC Motor, Remote Desktop, laboratory model

Bibliografická citace:

DOBROVOLNÝ, P. *Internetové ovládání laboratorních modelů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 54 s. Vedoucí diplomové práce byl prof. Ing. František Zezulka, CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Internetové ovládání laboratorních modelů jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2011** ……

podpis autora

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Děkuji vedoucímu diplomové práce prof. Ing. František Zezulka, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **23. května 2011** ……

podpis autora

Obsah

Seznam obrázků

1

ÚVOD

Cílem této práce je vytvoření dvou bezobslužných modelů, z nichž první bude softwarový a druhý fyzický. Oba mají sloužit na Vysoké škole polytechnické v Jihlavě k výuce na programovatelných automatech WAGO 750, popřípadě na SCADA systému Control Web.

Softwarový model je řešen jako zjednodušená simulace řízení vláčků v kolejišti a práce se zabývá tvorbou modelu, jeho nastavení a následně i jeho řízení. Dalším krokem je propojení modelu a PLC a poté i jeho řízení pomocí WAGO 750.

Fyzickým modelem je vrtačka, která slouží k výuce sekvenčního řízení. Zde se práce zabývá návrhem samotného modelu, snímacími prvky a pohony celého modelu. V konečné fázi se zabývá jeho řízením pomocí PLC.

Oba modely mají být připojeny k systému LabLink, což je software sloužící k připojení studentů za účelem zpracovávání laboratorních úloh ze vzdáleného pracoviště.

Tato práce by dále měla částečně sloužit i jako návod pro řešení úloh a specifikovat možná zadání úloh pro laboratorní cvičení.

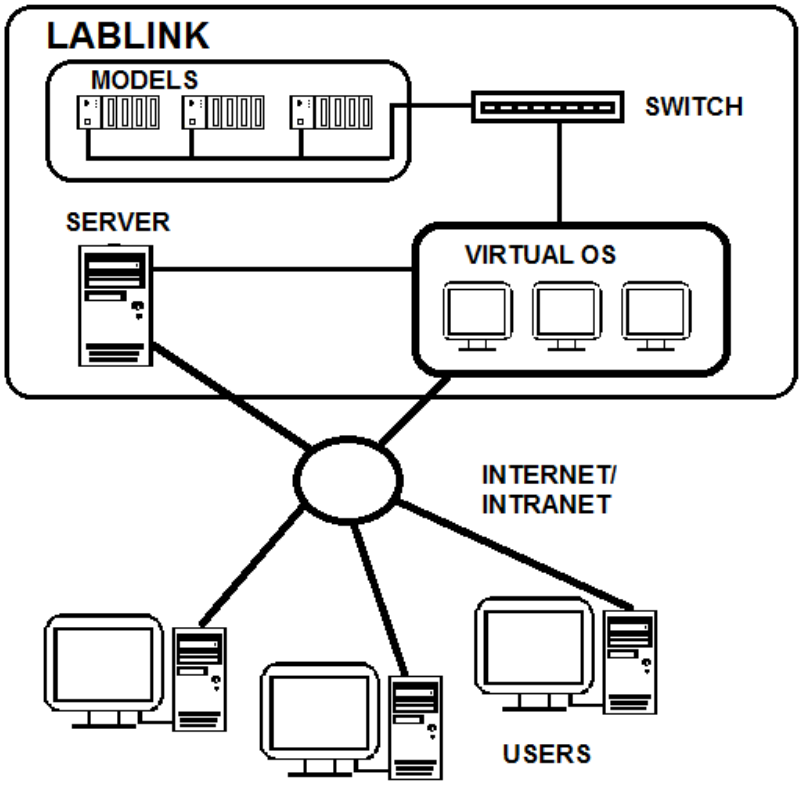
2 LABLINK [12]

Systém LabLink vznikl na Českém vysokém učení technickém v Praze, Fakultě elektrotechnické. Úkolem této aplikace je vyšší efektivita pro využití fyzických modelů. Ty bývají často vyrobeny v jediném kuse, a proto k nim může být fyzický přístup z časových důvodů omezen. Začlenění modelů do systému vzdáleného přístupu může systém dané modely studentům zpřístupnit po celý den a tím zefektivnit jejich využití. Další výhodou takového systému je přístup ke vzdálené ploše, na které jsou již nainstalována vývojová prostředí, která bývají zpravidla placenou službou. Touto metodou může být použita jedna licence na jednom zařízení, ke kterému má přístup větší počet uživatelů. Pro lepší využití jsou pro modely kladeny požadavky na bezúdržbovost modelu při běžném provozu, čímž je myšlena schopnost uvést vzdáleně model do defaultního stavu. Pozorování fyzických modelů je umožněno pomocí IP kamer, ke kterým se dá přistupovat pomocí internetového prohlížeče.

Systém LabLink zastává funkce:

- správa uživatelů
- spouštění prostředí virtuální operačních systémů
- přepínání uživatelů ke správnému modelu
- zabezpečení přístupu
- administraci a správu modelů

Uživatel si může zablokovat konkrétní úlohu na jemu vyhovující dobu a určí si zároveň čas, který pro práci potřebuje. Ve stanovenou dobu, na kterou si úlohu zablokoval, se pak může připojit k vytvořenému virtuálnímu operačnímu systému, který je vytvořen pomocí programu VMWare Workstation. Nově vytvořený operační systém má přístup k řídicímu systému daného úlohou a případně i web kamery, jež danou úlohu snímá. Tento systém je dopředu připraven pro konkrétní úlohy a je po každém přihlášení v defaultním stavu. Tím jsou odstraněny provozní problémy, které by mohli způsobit někteří uživatelé poškozením instalace operačního systému.



Obr. 1: Schéma zapojení systému LabLink

Na serveru může být nainstalována libovolná linuxová distribuce. Tvůrci používají operační systém SUSE Enterprise Server, v případě VŠP Jihlava je použita distribuce openSuse, která tvoří platformu pro instalaci aplikace VMware Workstation. Tato aplikace nabízí řadu funkcí. Velkou výhodou je podpora operačních systémů pro Windows, Linux, NetWare a FreeBSD, včetně nejnovějších Windows 7. Architektura zahrnuje 32 i 64 bitovou podporu a až 10 síťových karet. Vlastností, která je využita v systému LabLink, je schopnost spouštět virtuální stroje na pozadí bez uživatelského rozhraní aplikace VMware Workstation. Uživatel tedy provede registraci úlohy a času na serveru, který vytvoří virtuální operační systém s nastavením pro konkrétní úlohu a uživatel se v daném čase přihlásí přímo na nově vytvořenou plochu, pomocí které má prostřednictvím switche přístup jak k řídicímu systému úlohy, tak i web kamery, která úlohu snímá.

Přepínač (switch) musí umožnit připojení do příkazové řádky (CLI). Tím dochází k dynamické tvorbě a rušení VLAN a přidávání portů do vytvořených VLAN. Vlastní skripty pro správu přepínače (switch) jsou posílány do přepínače ze systému LabLink.

3 ŘÍZENÍ MODELŮ

Řízení modelů bude prováděno dvěma způsoby. První z nich je řízení modelu pouze pomocí PLC od firmy WAGO a druhým způsobem je využití vstupních a výstupní karet PLC a řízení softwarem Control Web.

Během dynamického vývoje, který provázel systémy od různých výrobců, vznikla nutnost spolupráce mezi jednotlivými systémy. Vývojem vznikly systémy, které se staly standardy v komunikaci. Jednotlivé standardy zahrnují technické prostředky, jako jsou sběrnice PCI, USB či Ethernetu. Zároveň vznikly i programové rozhraní a protokoly, například síťový standard TCP/IP, HTTP, FTP a formáty dokumentů HTML a XML.

Klíčovou úlohu v řídicích systémech přebírá programové vybavení. Technické vybavení je ve většině případů jednotné a záměnné. Až software je tou vrstvou celého systému, která jej činí unikátním a odpovídajícím specifickým potřebám zákazníků. Ti pomocí software se svými aplikacemi komunikují. Software implementuje řídicí algoritmy a oživuje celý systém, dává mu vlastní inteligenci. A je to právě software, který je nutno vždy znovu vytvořit nebo alespoň upravit. [1]

3.1 WAGO-I/O-Systém 750 [13]

Pro řízení modelů má být používán WAGO-I/O-Systém 750. Systém umožňuje připojit k jedné distribuované stanici libovolné kombinace analogových a digitálních vstupních/výstupních modulů. Každý z modulů má 1, 2, 4 nebo 8 kanálů. Na výběr jsou moduly pro práci s napěťovými úrovněmi pro stejnosměrné i střídavé napětí. Například vstupní digitální moduly pro stejnosměrné napětí jsou 5V, 24V, 42V, 48V a 110V. Pro střídavé napětí pak moduly pro 24V, 42V, 120V a 230V. Vedle vstupních digitálních modulů se vyrábí i výstupní digitální moduly, na trhu jsou dále k dispozici speciální moduly pro PWM a analogové moduly. Vstupní analogové moduly mohou zpracovávat napětí ±10V, 0-10V nebo proudovou smyčku 0-20 mA nebo 4-20mA.

Výrobce dodává velkou nabídku komunikačních modulů s programovatelným kontrolérem, které umožňují připojení na sběrnice Ethernet TCP/IP, PROFINET IO, ETHERNET Powerlink, Profibus, Interbus, DeviceNet, CANopen, CAL, Modbus a další. Tyto moduly musí být umístěny, v sestavě PLC, vždy na prvním místě.

Použité PLC, v této práci, je v sestavě:

Objednací číslo	**Popis**
750-841	Programovatelný komunikační modul Ethernet TCP/IP
750-530	8 digitálních výstupů 24V DC
750-430	8 digitálních vstupů 24V DC
750-530	8 digitálních výstupů 24V DC
750-430	8 digitálních vstupů 24V DC
750-600	Zakončovací modul vnitřní sběrnice

3.1.1 CODESYS [3]

CoDeSys je vývojové prostředí pro tvorbu řídicích programů pro systémy PLC podle normy IEC 61131-3, které bylo vytvořeno firmou 3S. Funkcí prostředí je:

- textové a grafické editory pro naprogramování
- ladění
- vizualizace řízeného procesu
- tvorba dokumentace vytvořených programů

Prostředí umožňuje programování v několika jazycích, které lze v rámci projektu kombinovat. Implementované jazyky se nazývají:

LD - Ladder Diagram (reléové schéma)

FBD - Function Block Diagram (jazyk funkčních bloků)

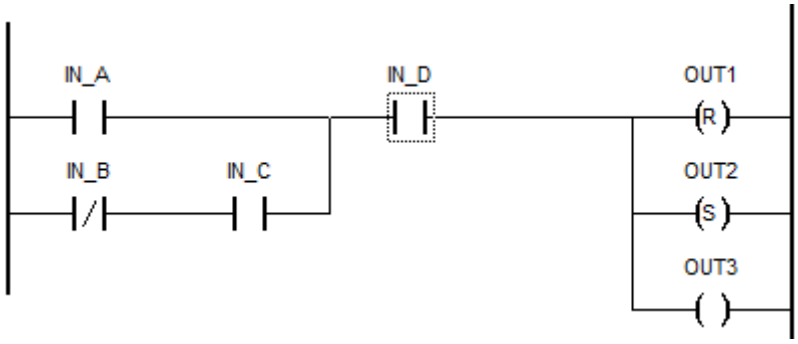
IL - Instruction List (sada instrukcí)

ST - Structured Text (strukturovaný text)

SFC - Sequential Function Chart (jazyk sekvenčního programování)

3.1.1.1 LD

Jedná se o grafický programovací jazyk, který umožňuje používat relé a cívky.



Obr. 2: Příklad LD

Na Obr. 2 je příklad reléového schématu. V levé části jsou vstupní proměnné (relé) a v pravé výstupní (cívky). Každé relé je znázorněno dvěma krátkými svislými čarami. Pokud je relé proškrtnuto, pak je negované. Cívky jsou znázorněny uzavřenými závorkami. Znak S značí set (nastaví hodnotu TRUE) proměnné a R označuje reset proměnné (nastaví hodnotu FALSE). Prázdné závorky zapisují do proměnné hodnotu, která odpovídá hodnotě výstupu vstupních relé.

Celé schéma je možno představit tak, že na postraní čáry je připojeno napájení z baterie, relé jsou vypínače a výstupy žárovky. Pokud sepneme vypínač IN_A a zároveň i vypínač IN_D, pak se napájení dostane až k výstupům, tedy žárovkám. V případě přivedení napájení se provede akce. Na výstupu jsou znázorněny v závorkách 3 možné výstupy, tedy 3 různé akce. Prázdná závorka pracuje stejně jako běžná žárovka

v osvětlení. V případě připojení napájení se rozsvítí, jinak je zhasnutá. Z pohledu programu bude v době svitu žárovky nastavena hodnota proměnné OUT3 na logickou 1. Závorka se znakem S (Set) způsobí pouze nastavení proměnné OUT2 na hodnotu logické 1. Opakem této akce je R (Reset), která pouze nastaví hodnotu proměnné na logickou 0. Pokud budou proměnné z příkladu opravdu žárovky a logická 1 je rozsvíceno a logická 0 zhasnuto, pak bude OUT1=0 a OUT2=1.

IN_D je v tomto příkladu rozhodující. V případě, že bude IN_D rozepnuto (logická 0), pak na výstupech nedojde k žádné akci.

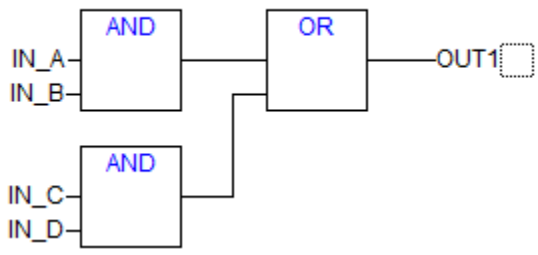
Zároveň jsou v příkladu vstupy IN_B a IN_C. U nich dojde k popisu s logickými hodnotami TRUE a FALSE. K přivedení napájení dojde, pokud bude splněna podmínka, že IN_B = FALSE, IN_C = TRUE a IN_D = TRUE. Hodnota FALSE je v tomto případě nastavena z důvodu, že přeškrtnutý znak relé je negovaný.

Relé i cívky se vkládají na plochu z lišty nástrojů.

3.1.1.2 FBD

I v tomto případě se jedná o grafický programovací jazyk. Umožňuje práci s proměnnými bool i s analogovými výrazy. Na Obr. 3 je znázorněna funkce:

OUT1 = IN_A * IN_B + IN_C * IN_D



Obr. 3: Příklad FBD

V CoDeSys se schéma vykresluje pomocí myši a příkazů na liště nástrojů. Vývojové prostředí nabízí vytváření boxů, přidávání vstupů a výstupů boxů, negaci signálů, vytváření skoků a návrat z funkce. Vlastnosti jednotlivých boxů jsou přiřazovány názvem boxu. V příkladu je použito AND a OR, ale existují i další. Pokud budeme chtít časovač, zapíšeme do boxu „TON“ a box se nám sám překreslí. Při návrhu je třeba dobře rozlišovat, co je ve schématu právě označeno. Pokud budeme chtít

připojit na vstup boxu ještě jiný box, pak musíme mít označenou nožičku, nikoliv pole pro název proměnné. Taktéž nejde připojit na nožičku výstupu boxu další box, ale je třeba mít označen přímo box, ke kterému má být nový připojen.

3.1.1.3 IL

Tento programovací jazyk umožňuje snadné programování na akumulátor, je tedy podobný programovacímu jazyku assembler.

Mezi základní instrukce patří [14]:

LD - Načte operand do akumulátoru

LDN - Načte negovanou hodnotu operandu do akumulátoru

ST - Obsahu akumulátoru zapíše do proměnné nebo na adresu

S - Nastavuje operand na hodnotu TRUE

R - Nastavuje operand na hodnotu FALSE

A - Bitový součin z akumulátoru a operandu

OR - Bitový součet akumulátoru a operand

ADD - Sčítání akumulátoru a operandu

SUB - Odčítání akumulátoru a operandu

MUL - Vynásobení akumulátoru a operandu

DIV - Vydělení akumulátoru a operandu

JMP - Nepodmíněný skok

JMPC - Podmíněný skok. Pokud je akumulátor = TRUE, pak dojde ke skoku

CAL - Výzva programu nebo FUNCTION_BLOCK

Příklad pro logický součin proměnné A a B a zápis do proměnné C by vypadal takto:

```
LD A

A B

ST C
```

3.1.1.4 ST

Tento jazyk je obdobou jazyků Pascal a C. V konstrukcích se používají příkazy jako IF, WHILE, CASE a FOR. Tyto příkazy jsou ideální pro podmíněné programování a pro smyčky.

Pro příklad konstrukce uvedu posloupnost příkazů prvně pro IF:

```
IF a < 5 THEN
    c := 10;
ELSIF a = 5 THEN
    c := 5;
ELSE
    c := 0;
END_IF
```

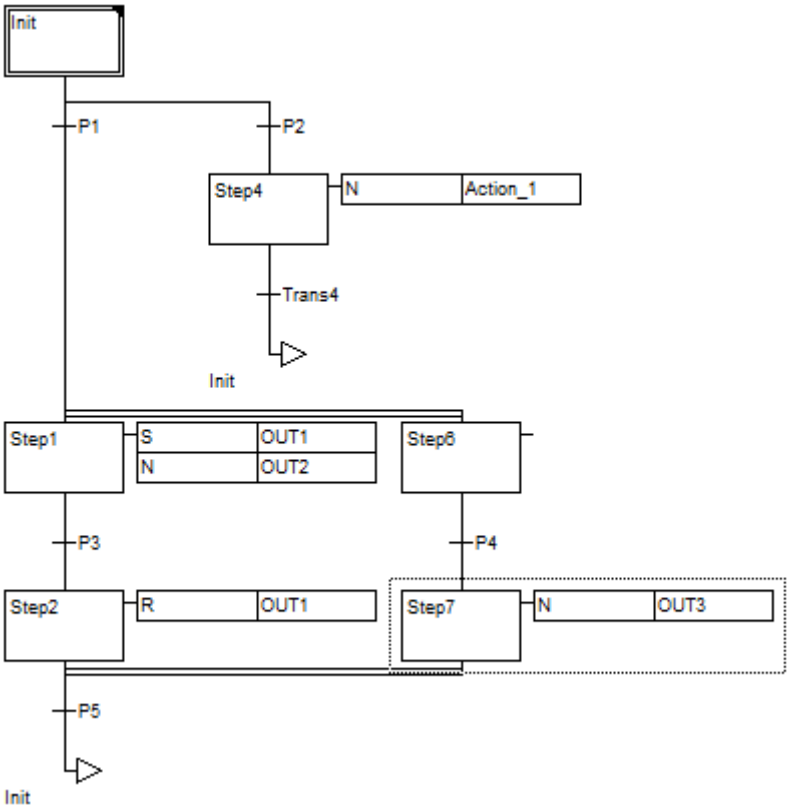
Smyčka FOR má takovouto syntaxi.

```
FOR i := 1 TO 100 BY 10 DO
    x[i]:=i;
END_FOR
```

3.1.1.5 SFC

Jedná se o sekvenční funkční grafy. Vhodné pro programy, které fungují v jednotlivých krocích. Diagram se skládá z kroků a přechodů. Kroky jsou stavy řízeného systému. Ke každému stavu je přiřazen blok akcí. Jednotlivé stavy jsou odděleny přechody, na které jsou vázány podmínky pro přechod do dalšího stavu. Tyto stavy nabývají hodnot TRUE nebo FALSE. Mohou zde být zapsány i podmínky (<, >, =). Každému bloku lze naprogramovat v libovolném, zde uvedeném, jazyce akce, které mohou být provedeny před krokem nebo za krokem. Obě akce mohou být provedeny pouze jednou. Příkazy pro nastavení akce daného kroku jsou Add Entry-Action a Exit-Action.

Touto metodou lze provést i větvení a to buďto na alternativní větve, kdy se program vydá jen jednou větví, nebo paralelní souběh jednotlivých větví a jejich synchronizace na konci větvení.



Obr. 4: Příklad SFC

Na Obr. 4 je příklad SFC. Přechody P1 a P2 se rozhoduje, kterou větví program bude pokračovat. Pokud program pokračuje přechodem P1, pak se rozvětví do dvou souběžně běžících větví. Jakmile je v obou větvích dosaženo posledního stavu, pak teprve pokračuje program pouze v jednom vláknu. Pro blok akcí platí následující seznam.

- N (Non-stored) - Akce je aktivní tak dlouho, jako je aktivní krok .
- R (overriding Reset) - Akce je deaktivována.
- S (Set / Stored) - Akce se aktivuje a zůstává aktivní.
- L (time Limited) - Akce se aktivuje na určitou dobu.
- D (time Delayed) - Akce je aktivní na určitou dobu. V případě, že krok bude i nadále aktivní, pak zůstane akce aktivní do konce aktivity kroku.
- P (Pulse) - Akce se provede pouze jednou, pokud je krok aktivní.
- SD (Stored and time Delayed) - Akce se aktivuje po určité době a zůstává aktivní, dokud není deaktivována.
- SL (Stored and time Limited) - Akce se aktivuje na určitou dobu.

3.1.2 OPC [7]

Nejprve je třeba zodpovědět otázku, co je to OPC. Jedná se o zkratku, která označuje OLE for Process Control (OLE-Object Linking and Embedding). Jde o standardizované rozhraní, které slouží k přístupu zpracovávaných dat. Založeno je na standardu COM a DCOM2 od firmy Microsoft. Tyto protokoly byly rozšířeny o přístup k datům v průmyslové automatizaci. Zařízení se rozdělují na server a klient. Vizualizace jsou obvykle označovány jako klient, zatímco programy sloužící pro záznam provozních dat jsou označovány jako server. Nutnou podmínkou je dodání OPC serveru k PLC systému.

Každý OPC server definuje nějaký prostor jmen, jimiž identifikuje všechny datové položky (kanály), které má k dispozici. Prostor jmen může být jednoúrovňový (též označovaný jako plochý) nebo hierarchicky strukturovaný (podobně jako např. strom).

OPC server může (ale nemusí) poskytovat rozhraní, které umožní prohledávat jeho jmenný prostor. To je velmi užitečné, protože jakákoliv komunikace s OPC serverem vyžaduje zařazení každé položky (kanálu) do nějaké skupiny a každou položku je nutno

přesně pojmenovat. Pokud server nemá rozhraní poskytující jména položek, je na uživateli a výrobci daného serveru aby zvolili způsob zadávání jmen. [7]

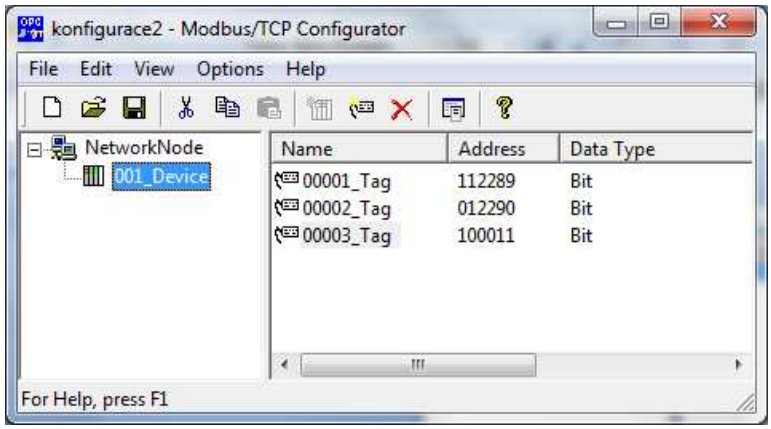
Výhody OPC:

- Odpadá nutnost návaznosti programového a technického vybavení, které využívají vlastní protokoly zavedených firem. Jedná se hlavně o firmy, které brání své produkty a především protokoly.
- OPC standard je spravován neziskovou organizací OPC Foundation (http://www.opcfoundation.org/). Použití standardu nevyžaduje žádné licenční poplatky.
- Snižují datovou náročnost na síti pro větší řízené systémy. OPC servery pravidelně načítají data z PLC a řídicích modulů, které můžou OPC klienti pravidelně získávat ze serveru.

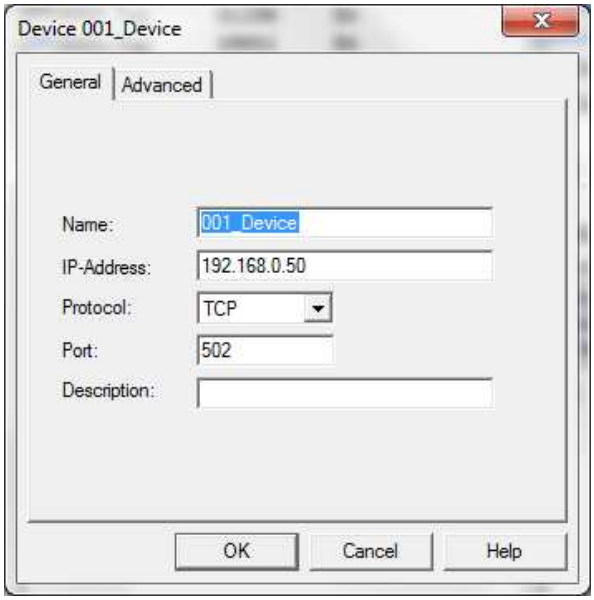
3.1.3 WAGO OPC Server [6]

Společnost WAGO dodává WAGO OPC Server Modbus TCP. Spolu s tímto softwarem dodává společnost program ModbusTCP Configurator. V levém sloupci na Obr. 5. je seznam stanic, se kterými má OPC server komunikovat. Na obrázku je pouze jedna stanice pojmenována 001_Device. Na Obr. 6 je vidět nastavení komunikace s touto stanicí, kterou je PLC WAGO 750. Stačí pouze nastavit IP adresu PLC, vybrat protokol mezi TCP a UDP a nastavit port, který je pro WAGO nastaven 502, pokud použijeme protokol TCP. Je ale možné změnit jej. Jak je z názvu OPC serveru zřejmé, komunikace mezi OPC serverem a PLC probíhá s použitím protokolu Modbus TCP/IP.

V pravém sloupci jsou definovány tagy určené ke komunikaci mezi zařízeními.

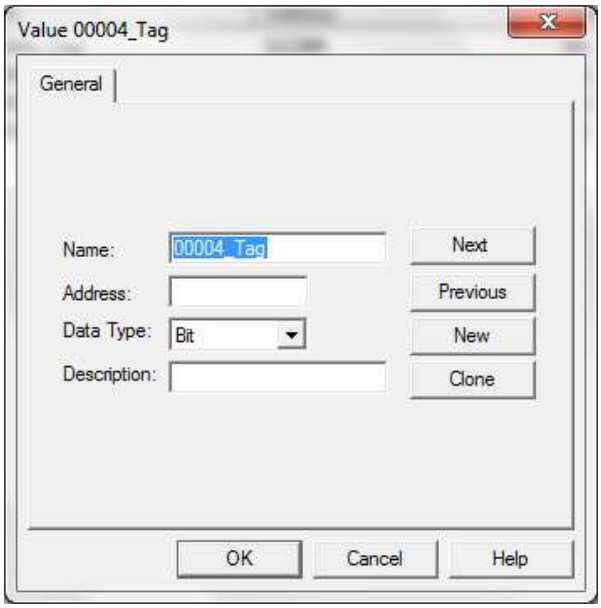


Obr. 5: Konfigurátor pro OPC server



Obr. 6: Nastavení komunikace OPC serveru s PLC

Pro vytvoření nového tagu je potřeba mít označené zařízení, jehož tag budeme požadovat. Poté v záložce Edit vybereme volbu Insert Tag a otevře se nové okno.



Obr. 7: Insert Tag

První kolonka slouží pouze jako unikátní jméno tagu pro OPC. V druhé kolonce ale říkáme, jaká data chceme číst z PLC. Třetí kolonka je s druhou svázána. Pokud půjde pouze o bit, bude tento datový typ automaticky vybrán. Pokud by ale nešlo o bit, je možné vybírat mezi Byte, Word, DWord, Char, Short a Long. Pro řízení obou modelů bude stačit pouze bit.

3.1.3.1 Adresa proměnných [5]

Adresa je skládána ze šesti znaků, z nichž první udává typ komunikace po Modbus TCP/IP. Jednotlivé typu udává Tab. 1.

Tab. 1: První znak z adresy pro Modbus TCP/IP

Příkazy	Hodnota
Output Coil	0
Input Coil	1
Input Register	3
Output Register	4

Vstupy a výstupy jsou chápány z pohledu klienta. Hodnoty 0 a 1 jsou výstup nebo vstup cívky, tedy datový typ bool, kde se přenáší jeden bit. Hodnoty 3 a 4 jsou pro čtení 8 – 32 bitů, podle vybraného datového typu.

Zbylých 5 znaků určuje konkrétně hodnotu v paměti PLC. Pro čtení a zápis dat v paměti PLC musí být data uložena v paměti merkerů. Pro datový typ bool můžeme takový merker zapsat jako:

Promena1 AT %MX0.0:BOOL;

Promena2 AT %MX0.1:BOOL;

,nebo pro datový typ Word takto:

Teplota1 AT %MW100: WORD;

Teplota2 AT %MW101: WORD;

Adresa %MXx.y, zapsaná v PLC se vypočítává jako:

3000hex+(x*16)+y+1=12288+(x*16)+y+1.

Pokud k výsledné hodnotě přidáme i hodnotu pro Output Coil, která je 0, pak můžou adresy zadávané v okně na Obr. 7 vypadat následovně.

Tab. 2: Příklady nastavení adresy pro cívku

Adresa Wago	Adresa
%MX0.0	012289
%MX0.1	012290
%MX1.0	012305

Podobně se vypočítává i adresa pro jiné datové typu. Uvedeme příklad pro datový typ Word. Adresa %MWx se vypočítá jako:

$$3000hex+x+1=12288+a+1.$$

Zde pro příklad přidáme k adrese hodnotu pro Input Register, který má hodnotu 3 a adresy vypadají následovně.

Tab. 3: Příklad nastavení adresy pro register

Adresa Wago	Adresa
%MW0	312289
%MW1	312290
%MW100	312389

Je možné přistupovat i přímo k fyzickým vstupům a výstupům, které nejsou adresovány od adresy 3000hex, ale od adresy 0 s offsetem 1.

Po ukončení konfigurace a jejím uložením je možné danou konfiguraci spustit. Před spuštěním samotného OPC serveru se v programu pro konfiguraci v záložce File zvolí položka Set OPC Configuration. Až poté je možné zapnout OPC server. Pokud se budou provádět změny v natavení, je třeba OPC server vypnout. Nenajdete jej ale na žádné liště, je třeba jej vyhledat ve správci úloh systému Windows. V záložce Procesy jej najdete pod názvem MBTOPC.exe.

3.2 Control Web [1]

Systém Control Web, od české firmy Moravské přístroje a.s., je nástroj pro vývoj aplikací pro vizualizaci, řízení a aplikací pro sběr, ukládání a vyhodnocování dat. Tento systém je schopný realizovat logické a sekvenční řízení, což jsou funkce PLC, které může systém zastoupit při řízení pomalejších dějů. Aplikace je objektově orientovaná

komponentová architektura, která umožňuje nasazení i pro vizualizační a řídicí aplikace reálného času. Podle těchto vlastností můžeme tento systém označovat jako SCADA.

Control Web pracuje jako aplikace operačního systému v programovém rozhraní Win32. Rychlost celé vytvořené aplikace tedy není závislá pouze na rychlosti CPU, ale také na operačním systému.

Systém je možné využít jako centralizovaný nebo decentralizovaný:

Centralizovaný systém [15]: Do PC, na kterém běží program Control Webu je možné instalovat V/V karty, ke kterým jsou připojeny čidla a pohony řízeného procesu. Tento systém je vhodný pro menší aplikace a je omezen V/V kartami, které je možné instalovat k PC. Firma Moravské přístroje nabízí systém DataLab IO, který je možné připojit pomocí USB. Nevýhodou je omezená délka propojovacího kabelu na 5m. Novější jednotky už mají ale 10/100 Mbit Ethernet rozhraní, které umožňuje připojení na větší vzdálenosti.

Decentralizovaný systém: Tento systém je vhodný pro větší aplikace. Program Control Webu se připojuje pomocí síťových rozhraní ke vzdáleným vstupním a výstupním modulům. Obvykle se používají PLC.

3.2.1 Připojení Control Webu k PLC [1]

Protože výrobci mají vlastní organizaci dat s různými pravidly, je potřeba, aby byl přenos dat podporován softwarově. Připojení systémů k Control Webu řeší Moravské přístroje vytvářením ovladačů, které se do vytvářené aplikace přidávají.

Moravské přístroje nabízejí ovladače přímo ke konkrétním PLC, nebo modulům, od různých výrobců, jako je Allen Bradley, AMiT, Doyo, Mitsubishi, Omron, Siemens a Tecoma. Tyto ovladače zaručují znalost uspořádání dat a přístup k nim.

Druhou možností je využití ovladačů, které využívají komunikačních standardů MODBUS, MODBUS TCP/IP, OPC. Tyto protokoly bývají jednotlivými výrobci podporovány a lze je využít pro připojení modulů a PLC, pro které nemají Moravské přístroje ovladač. V této úloze může být využit, pro komunikaci s WAGEM 750, MODBUS TCP/IP i OPC. Pro tuto aplikaci je zvoleno OPC.

3.2.1.1 Ovladač OPC klient pro Control Web [7]

Před připojením ovladače OPC klienta k programu v Control Webu je třeba nastavit dva soubory.

Prvním z nich je parametrický soubor (.PAR), který říká, s jakým zařízením a s jakým datovým prostorem bude aplikace pracovat. Tento soubor se dá rozdělit na 2 části. První identifikuje server takzvaným CLSID (Class Identifier). Druhá část definuje kanály v podobě:

číslo_kanálu = identifikátor_položky_OPC_serveru

V tomto případě vypadají první položky:

1=001_Device/00001_Tag

1=001_Device/00002_Tag

Druhým souborem je mapovací soubor (.DMF), který aplikaci říkám, jaké kanály ovladač nabízí. Definovány jsou kanály, datový typ a směr komunikace. V práci vypadají kanály takto:

1 boolean input

2 boolean bidirectional

Pro ulehčení práce je spolu s Control Webem dodáván konfigurační nástroj OpcDrvCf. S tímto nástrojem je velmi jednoduché vytvořit oba výše popsané soubory. Program vyhledá OPC servery a všechny nalezené nabídne k výběru. Po vybrání dojde k připojení. Pokud je program úspěšně připojen k serveru, pak máme k dispozici jeho jmenný prostor. Z něj můžeme vybrat položky, které budou Control Webem používány jako kanály. Čísla kanálů jednotlivých položek je možné ručně nastavit, nebo nechat vygenerovat. Následně je možné nechat vygenerovat hotové konfigurační soubory.

3.2.2 Práce s programem Control Web

Spolu s Control Webem je dodávána i podrobná dokumentace pro práci s aplikací. Dokumentace je psána formou tutoriálu, což je velmi nápomocné začínajícím vývojářům. Dokumentaci lze nalézt v adresáři, ve kterém je program nainstalován, nebo přímo v programu a to v Nápověda/Obsah. Případně stačí stisknout klávesu F1. V okně jsou 4 záložky. Obsah, Rejstřík, Hledat a Oblíbené položky. Pod záložkou Obsah lze nalézt dokumentaci i s návody.

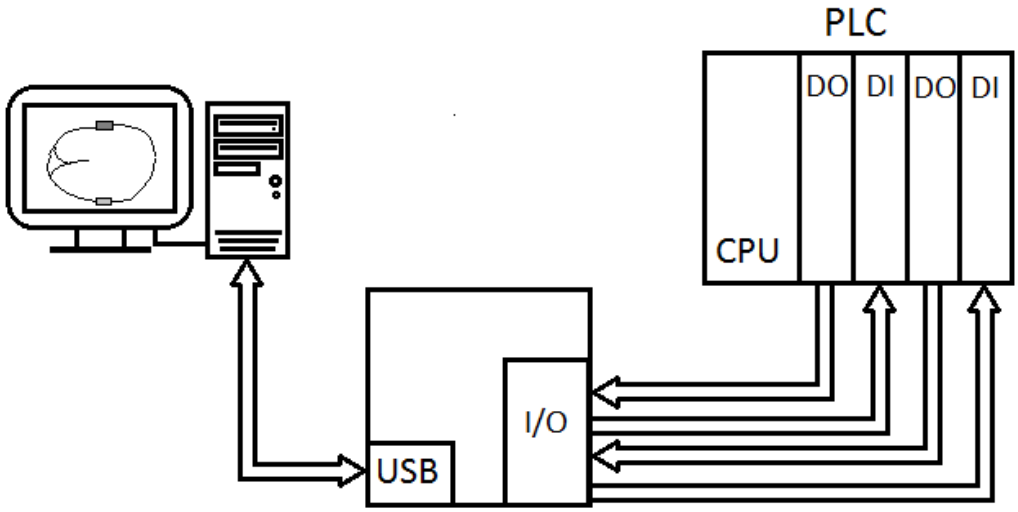
Projekt je možné vytvářet v textovém editoru, nebo v grafické nástavbě. Obě tyto části jsou provázány, proto se změna v textovém editoru promítne v grafické nástavbě a naopak. Bohužel této vlastnosti tvůrci využili pro řešení chyb při vytvářeném projektu.

Pokud vývojář například smaže proměnou, která je někde v projektu použita, zaznamená program chybu a odkáže vývojáře na textový editor a do grafické nástavby jej nepustí. Je proto třeba nespoléhat pouze na popsaná tlačítka a úhledné záložky, ale i strukturu zdrojového kódu projektu, aby bylo možné projekt opravit. Toto ale není v žádném případě velkou překážkou. I generované zdrojové kódy jsou strukturované do bloků a k přehlednosti pomáhá i zvýrazněná syntaxe.

Naproti tomu má velmi dobře propracovaný grafický editor. Z knihoven nástrojů je možné přetahovat nástroje na plochu a provádět jejich nastavení. Je možná vkládat pozadí, textury, nebo vytvářet nové přístroje. Je velmi variabilní pro využití v různých odvětví průmyslu.

4 MODEL VLÁČKOVIŠTĚ

Vláčkoviště je úloha, která slouží k výuce na programovatelných automatech WAGO. Na vstupy a výstupy řídicího automatu je připojeno zařízení, které slouží jako převodník ze stavů vstupních a výstupných modulů na USB. K tomuto převodníku se v počítači připojuje aplikace, která se snaží částečně simulovat provoz na jednoduché trati.



Obr. 8: Zapojení modelu „Vláčkoviště“

4.1 Vývojové prostředí Qt Creator [16]

Pro vývoj nové aplikace jsem použil vývojové prostředí Qt Creator, které používá toolkit Qt. Qt vytvořila společnost Trolltech v roce 1999 a prodala jej v roce 2008

společnosti Nokia. Slouží jako multiplatformní vývojové prostředí pro tvorbu programů, které mohou být doplněny grafickým prostředím. Vývojové prostředí neslouží pouze k vývoji GUI aplikací a konzolových aplikací, ale také aplikací pro mobilní telefony. Stejný kód lze přeložit pro platformy[10]:

- Linux (32 a 64-bit)
- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows Vista 64bit
- Microsoft Windows 7
- Apple Mac OS X 10.6 "Snow Leopard"
- Apple Mac OS X 10.5 "Leopard" x86_64 (Cocoa 32 a 64bit)
- Embedded Linux QWS (ARM)
- Windows CE 5.0 (ARMv4i, x86, MIPS)
- Symbian (Symbian/S60 5.0)

Qt je knihovna sloužící pro programování nejen v jazyce C++, ale také pro jazyky Ada (QtAda), C#, C, Pascal, Python (PyQt), Ruby (QtRuby), Perl (PerlQt), Java (Qt Jambi) [9]. Mimo to podporuje zpracování XML, práci s SQL, webovými aplikacemi, řetězci, vlákny, grafikou a multimédii. Další podporovaný jazyk je OpenGL (QtOpenGL), což je standart specifikující API pro tvorbu počítačové grafiky.

Výhodou využití Qt Creatoru je zvýrazněná syntaxe, automatické doplňování kódu, grafický designer (možnost skládat vizuální část pouhým přetahováním objektů do prostoru widgetu). Uložení grafického adapteru je ve formátu XML. Zároveň je možné do widgetu vepsat funkce i pomocí metod v C++. Další vlastností Qt Creatoru je správa projektů, debuger, možnost ladění, krokování programu pomocí breakpointů a sledování stavů proměnných. Součástí je také prohlížeč dokumentace Qt. V neposlední řadě je výhodou, že software i s knihovnami je volně ke stažení. Pro používání je třeba jen dodržování podmínek uvedených v GNU LGPL 2.1. Existuje také Qt komerční licence pro vývojáře, která je určena pro ty, kteří nechtějí sdílet kód v souladu s licencemi GPL nebo LGPL. Více informací viz [8].

Odlišností Qt od ostatních, je v knihovně QObjekt, díky které je možné vytváření signálů a slotů, které slouží pro komunikaci mezi objekty. V programu určujeme, ze kterého objektu půjde jaký signál a která metoda (slot) a ve kterém objektu bude tímto signálem volána. Asi nejjednodušší příkladem může být vytvořené tlačítko na ploše (widget). Při stisknutí takového tlačítka je vyslán signál CLICK() z objektu OBJEKT1. Řekněme, že kliknutím na tlačítko má být inkrementována číslice. Pro tuto funkci si

vytvoříme slot INKR() v objektu OBJEKT2. Pro určení, který signál je určen pro který slot, slouží metoda connect(). Celé propojení může vypadat takto:

connect(OBJEKT1, SIGNAL(CLICK()), OBJEKT2, SLOT(INKR()));

Sloty jsou vytvářeny stejně, jako běžné metody, jediným rozdílem je, že jsou v hlavičkovém souboru v sekci public slots, pokud se jedná o přístupnou metodu z jiného objektu. Syntaxe v hlavičkovém souboru by vypadala následovně.

public slots:

void INKR();

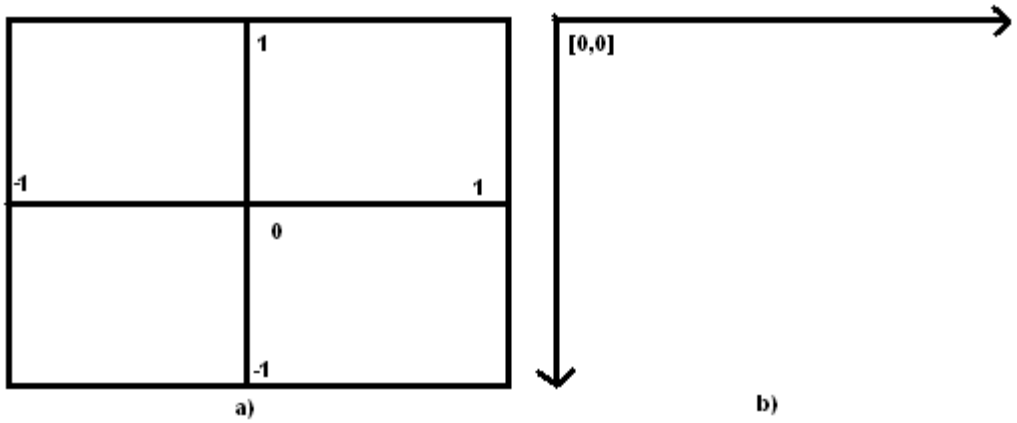
Podobně vypadá i syntaxe v hlavičkovém souboru i pro signály. Volání signálů je stejné, jako volání funkce.

Signals:

void CLICK();

4.2 OpenGL [17]

Před dalším popisem je třeba dodat, že zobrazení simulace je naprogramováno v OpenGL, což je standart specifikující multiplatformní rozhraní pro tvorbu počítačové grafiky. Použití OpenGL má vliv na pohled souřadnicového vykreslování. Zatímco vykreslování pomocí Qt, a dalších programech, má začátek souřadnic x=0 a y=0 umístěn v levém horním rohu obrazovky, OpenGL má tento počátek umístěn ve středu. Situace je znázorněna na Obr. 9. Oba programy mají osu x na horizontální ose a osu y na vertikální.



Obr. 9: Souřadnice a) v OpenGL, b) Qt

Na obrázku pro souřadnice v OpenGL je plocha ohraničena v horizontální ose -1≤x≤1a stejně tak i vertikální ose -1≤y≤1. Jedná se pouze o defaultní nastavení, které je možné změnit příkazem gluOrtho2D(-x/2,x/2,-y/2,y/2). Pro tento model bylo dokonce nutné tento příkaz použít. Pokud bychom nechali poměr stran 1:1 a zobrazili obraz na monitoru, nebo displeji, pak by se objekty začaly deformovat a místo čtverců by začaly vznikat kosočtverce a místo kružnic by vznikaly elipsy. Je tedy nutné nastavit vhodný poměr mezi stranami.

Funkcí OpenGL je vytvořit 2D nebo 3D scénu, která je poté přenesena jako 2D obraz na monitoru nebo displeji. Obraz nemusí být snímán pouze z jedné perspektivy, ale je možné hýbat kamerou a tím měnit pohled na celou scénu. Scéna je tvořena objekty ze základních geometrických prvků. Základní prvky, které jsou obsaženy v OpenGL je bod (GL_POINTS), úsečka (GL_LINES), trojúhelník (GL_TRIANGLES), čtyřúhelník (GL_QUADS), polygon (GL_POLYGON), … Každý objekt je vykreslován v bloku, který je otevřen příkazem glBegin(Název_objektu) a uzavřen příkazem glEnd(). Mezi těmito bloky jsou uvedeny vrcholy vykreslovaných objektů příkazem glVertex3f. Číslovka 3 uvádí, že se uvádí bod v třírozměrném souřadnicovém systému x, y a z. Poslední písmeno f uvádí, že jsou parametry příkazu udávány v datovém formátu s pohyblivou desetinnou čárkou. Celý blok by mohl vypadat pro čtverec ve 2D takto:

```
glBegin(GL_QUADS);
        glVertex2f(-a,a);
        glVertex2f(a,a);
        glVertex2f(a,-a);
        glVertex2f(-a,-a);
glEnd();
```

Pomocí OpenGL je možné vytvořené objekty posouvat a rotovat od počátku souřadného systému pomocí příkazů:

```
glTranslatef( Δx , Δy , 0);
    glRotatef(uhelf, 0.0f, 0.0f, 1.0f);
```

U obou příkazů se zadávají parametry v datovém typu float. Příkaz glTranslatef slouží k posunu počátku osy o Δx a Δy. Tímto příkazem budou všechny vykreslované

objekty posunuty o Δx a Δy od počátku. Třetím parametrem by byla osa z, která zde není využita, protože se jedná o 2D simulaci. Druhý příkaz glRotatef slouží k rotaci objektu. První prvek uhelf je úhel, o který se objekt pootočit. Zbylé tři parametry určují, podle které osy. U tohoto modelu docházelo k rotaci pouze podél osy z, proto je u 4. parametru hodnota „1.0“. Stejně jako u translace dojde k pootočení celé osy.

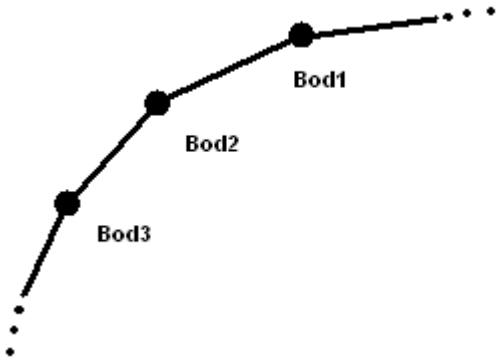
4.3 Řešení trasy vláčků

Pro určení trasy, po které se budou vlaky pohybovat, je potřeba vytvořit databázi souřadnic, po kterých se budou vlaky pohybovat. Tu je potřeba napsat tak, aby je využívaly oba vlaky a nezáleželo na tom, zda vlak jede po úseku například doprava nebo doleva. Pro jednoduchou manipulaci jsem zvolil systém párů souřadnic dvou bodů, mezi kterými se bude vlak pohybovat po přímce.

Čím více těchto bodů, tím více bude pohyb plynulý a méně ostrý. Naopak pro dlouhé rovné úseky stačí méně párů bodů. Je tedy zřejmé, že velikost databáze bude závislá na složitosti dráhy a vyžadované jemnosti pohybu vlaku.

Samotná databáze je uložená v textovém souboru souradnice.txt. Je tedy možné trasu upravovat v textovém režimu, čehož se dá využít například pro změnu názvů úseků, výhybek, případně na drobnou korekci souřadnic a dodávání aktivních prvků, jako jsou semafory.

4.3.1 Přímý pohyb modelu



Obr. 10: Znázornění bodů tratě

Na výše zobrazeném Obr. 10 je vidět zdiskretizovaná dráha zatáčky trati. Na Obr. 9 je zobrazen souřadnicový systém, který je použit. Čísla, udávající souřadnice, jsou desetinná typu float. Jak je na obrázku vidět, tak hraniční body zobrazovaného obrazu jsou defaultně pro obě osy x a y v rozmezí -1 a 1. Datový formát, ve kterém je zobrazen jeden pár souřadnic, je zobrazen jako:

#[Úsek],[Bod1.x/Bod1.y],[Bod2.x/Bod2.y],

Celý řetězec je dělen programem na jednotlivé segmenty, které používá pro svoji orientaci v databázi a další podpůrné nastavení, pomocí oddělovacích znaků „ , “. Druhou důležitou poznámkou je, že program ignoruje všechny mezery a entery. Úvodním znakem, nejen každého úseku tratě, je mřížka, za kterou bez jakéhokoliv znaku následuje název úseku, po kterém se jede. Následují dva body, mezi kterými se právě bude pohybovat vláček. Hodnota souřadnic x a y je rozdělena pomocí znaku „/“.

Na příkladu si ukážeme, v jakém formátu je tento tvar uložen v databázi s konkrétními čísly:

Budeme uvažovat body na Obr. 10, kde řekneme, že jejich hodnoty jsou:

Bod1= 0.3x0.3

Bod2= 0.2x0.2

Bod3= 0.1x0.1

První číslice je souřadnice x a druhá souřadnice y. Celý úsek bude označován jako S0. Znak “S” udává, že se jedná o úsek a číslovka 0 slouží k rozlišení jednotlivých úseků. Z důvodu komunikace je třeba dodržet číslování úseků, aby v databázi byla uvedena všechna čísla z řady, začínající číslem 0. To znamená, aby bylo označení úseku tratě pro 4 úseky S0, S1, S2, S3. Nemělo by ale dojít k situaci, kdy by byly úseky označeny S0, S2, S4, S5, kde by došlo k výpadku úseků S1 a S3. Pro samotnou simulaci by tento výpadek neměl vliv, ale na komunikaci mezi převodníkem a PC, a následně i samotným PLC, by nedošlo k odesílání informace o obsazení úseku S2, S4 i S5. V samotné databázi mohou být jednotlivé úseky promíchány, ale jedná se především o jejich zastoupení v celé databázi.

Výsledná databáze pro tyto 3 body potom je:

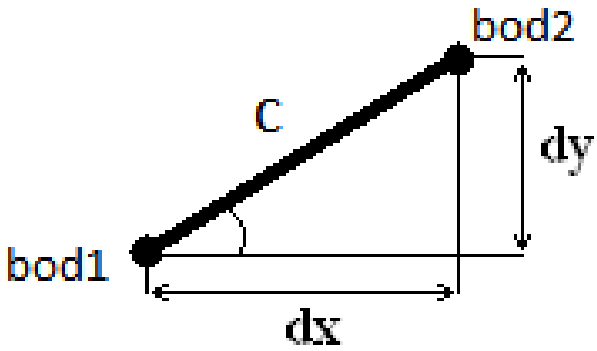
#S0,0.1/0.1,0.2/0.2,

#S0,0.2/0.2,0.3/0.3,

Tímto způsobem se ale dá popsat jen uzavřená trať bez jakýchkoliv výhybek. Pokud by ale nebyla křivka uzavřená, pak by se na koncovém bodu vlak otočil a jel zpět.

4.3.1.1 Výpočet posunu objektu

Samotný pohyb je vytvořen postupným generováním obrazu, který je tvořen v pravidelných časových intervalech. Velikost posunu je počítána ze vzdálenosti dvou bodů mezi, kterými se má právě objekt pohybovat.



Obr. 11: Pohyb mezi dvěma body

Pohyb v každém časovém intervalu je potřeba počítat tak, aby objekt plynule dosáhl požadovaného bodu jak souřadnicí x, tak y a neměla by nastat situace, kdy dosáhne objekt požadované souřadnice x, ale na souřadnicích y bude objekt v půlce cesty. Je proto potřeba vypočítat pro každou souřadnici přírůstek zvlášť. Společným číslem použitým pro výpočet je přímá vzdálenost mezi oběma body, v matematice bychom ji nazvali velikostí přepony trojúhelníku, zobrazeném na Obr. 11 (označíme si velikost přepony jako c).

Velikost přírůstku by se vypočítala jako:

-pro x: $\left|\frac{bod2.x - bod1.x}{c}\right|$

-pro y: $\left|\frac{bod2.y - bod1.y}{c}\right|$

Rychlost posunu se dá nastavit buď vynásobením vypočteného čísla, nebo změnou časového intervalu, který by měl být menší než 10ms. To z důvodu, ale lidské oko

nezaznamenalo jednotlivé časové intervaly, při kterých se vlak posune. Při dodržení tohoto času bude pro oko probíhat vizualizace plynule.

Zároveň je potřeba objektem otáčet, kvůli správné interpretaci pohybu vlaku po kolejích. Tento úhel se dá jednoduše vypočítat ze vztahu:

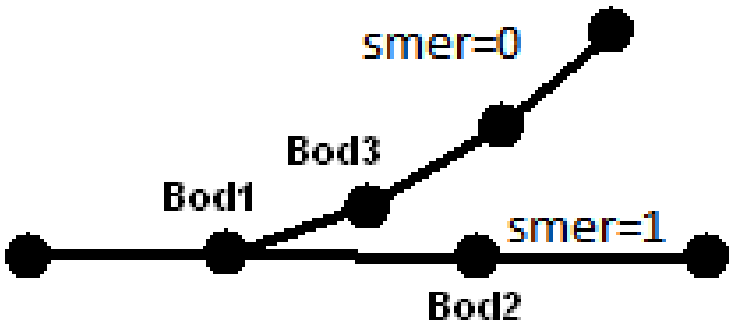
$$Uhel=arctg\ (dx/dy)$$

4.3.2 Řešení výhybek

V simulátoru je možné vytvořit pohyb po vyhybce podobně, jako je vytvořen pohyb po přímé trati. Zápis vyhybky je:

#[cislo_vyhybky],[Bod1],[Bod2],[směr],

#[cislo_vyhybky],[Bod1],[Bod3],[směr],



Obr. 12: Znázornění bodů pro řešení výhybky

Uvození žebříčkem zůstává, ale místo jména trati máme název výhybky, které je dáno označení ‘V‘. Opět následují souřadnice dvou bodů a posledním prvkem je směr, který určí, po které kolejnici se vlak vydá, pokud přijíždí na ukázaném příkladu zleva. Pokud by vlak přijížděl z pravé strany a jel ze směru, který není povolený, dojde k ukončení simulace z důvodu vykolejení vlaku. Program si též hlídá, aby vlak podle Obr. 12 neprojížděl výhybkou v pořadí Bod2, Bod1, Bod3 nebo v opačném pořadí. Po projetí 2 bodů výhybky vlak opouští výhybku. Směr se rozlišuje číslicemi 0 a 1. Číslování výhybek je obdobné úsekům, kdy musí být zastoupena všechna čísla od 0 do posledního použitého.

Upozornil bych na zápis, kde se prvně píše souřadnice bodu Bod1, tedy společný bod pro obě větve.

Pro příklad uvedu zápis s čísly:

Bod3= 0.1x0.05

Bod2= 0.2x0.0

Bod1= 0.0x0.0

, pro které bude zápis vypadat:

#V1,0.0/0.0,0.1/0.05,0,

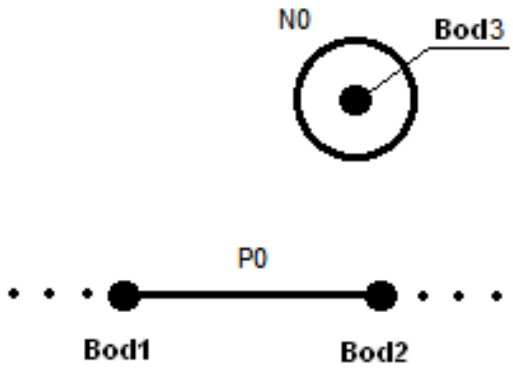
#V1,0.0/0.0,0.2/0.0,1,

Pokud bude směr vyhybky 0, vydá se vlak na Obr. 12 nahoru, pokud bude směr vyhybky 1, pak pokračuje rovně.

4.3.3 Zastávky

Zastávka určuje místo, kde se vlaky po zapnutí nebo restartu objeví. Zároveň je možné říci vlaku, projíždějícím zastávkou, jestli může jet, či jestli má zastavit. Zastávka je v databázi zapsána zápisem:

#[Zastávka],#[Úsek],[Bod1.x/Bod1.y],[Bod2.x/Bod2.y],



Obr. 13: Znázornění bodů pro zastávku

Zastávka má označení P spolu s číslem, které zastávku identifikuje a je třeba přiřazovat číslo zastávky od 0 až do poslední použité. Jinak se jedná o stejný zápis jako u úseků.

Další vlastností zastávky je, že pořadí bodů udává směr, kterým se vlak po povolení výjezdu ze zastávky vydá. Pokud by body byly zapsány tak, jako v příkladu nad Obr. 13, pak by se vlak vydal doprava, tedy směrem od Bodu1 do Bodu2. Pro směr doleva stačí v databázi přehodit pozici obou bodů.

V Obr. 13 je mimo úseku vidět i návěstí, které je umístěno v Bodu3. Návěstí je úzce propojeno se samotnou zastávkou. Je-li povolen výjezd nebo pouze průjezd zastávkou, pak bude svítit zelená. V opačném případě svítí červená. Pro databázi přijalo návěstí znak ‘N‘ a stejně jako předchozí části v databázi má i své číslo. Toto číslo ale musí přímo odpovídat zastávce, pro kterou je návěstí určeno. Návěstí N0 reaguje na dění zastávky P0. Stejně tomu je i u následujících zastávek a návěstích. V databázi je návěstí reprezentováno zápisem:

#[Návěstí],[Bod3.x/Bod3.y]

Na příkladu s konkrétními hodnotami uvedu zápis zastávky a návěstí. Hodnoty bodů budou přibližně odpovídat umístění bodů na Obr. 13. Úsek, na kterém je zastávka umístěna bude pro tento příklad zvolen například S2.

Pro body:

Bod1= -0.2x0.0

Bod2= 0.2x0.0

Bod3= 0.1x0.1

,pro které bude zápis vypadat takto:

#P0,#S2, -0.2/0.0, 0.2/0.0,

#N0, 0.1/0.1,

Využití návěstí není povinné, slouží pouze jako vizualizační složka, ale vizualizace je funkční i bez návěstí.

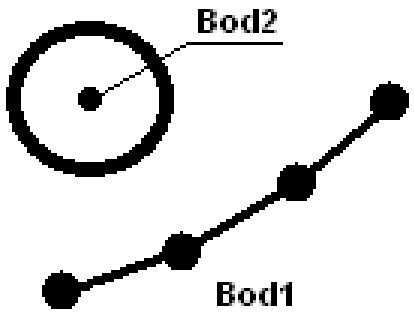
4.3.4 Semafory

Dalším prvkem jsou semafory. Jejich funkcí je zastavení vlaků v předem určeném bodě. Těmito předem určenými body se myslí body, které jsou již v databázi. Na Obr. 14 se jedna o Bod1. Bod2 už slouží pouze k tomu, aby program věděl, kde má semafor zobrazit. Semafory jsou v databázi uváděny ve formátu:

#[Semafor],[Bod1.x/Bod1.y],[Bod2.x/Bod2.y],

Semafory jsou označovány znakem ‘Z‘ a číselný zápis může vypadat, pro body uvedené v příkladu pro zastávku následovně:

#Z0,-0.2/0.0, 0.2/0.0,



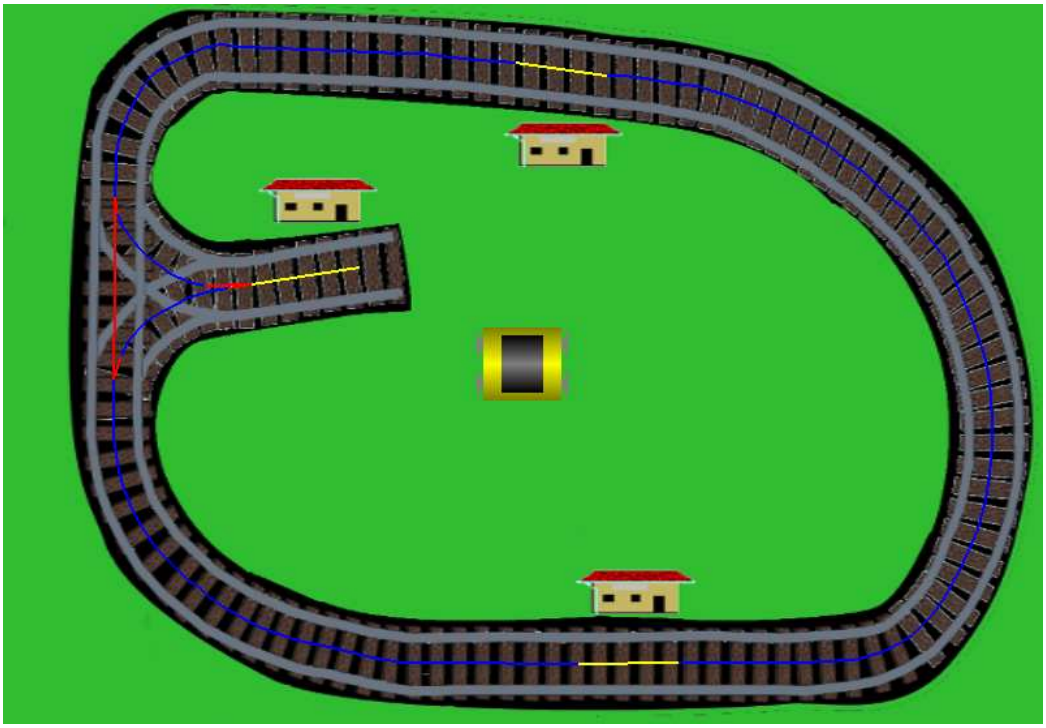
Obr. 14: Znázornění bodů pro semafory

4.4 Vytváření dráhy

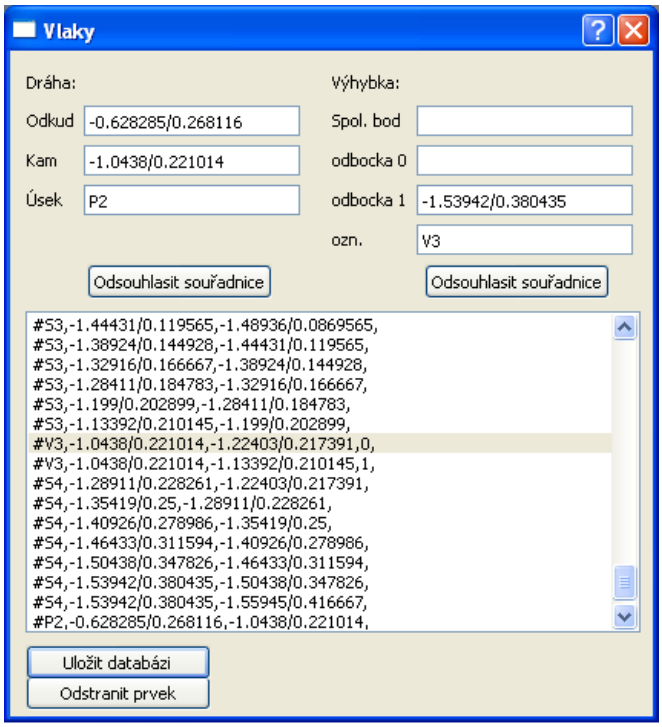
Protože návrh tratě by byl ručním vypisováním velmi obtížný, vytvořil jsem v programu okno sloužící pro vytváření databáze bodů, které je zobrazeno na Obr. 16. Okno by se dalo rozdělit do 3 částí. Před samotným vytvářením databáze pro simulaci je potřeba vložit obrázek s dráhou, který může sloužit i jako předloha pro tvorbu.

Obrázek je nutné umístit do stejné složky spolu s modelem pod názvem side11.png a obrázek by měl být 1:1. Program si sám obrázek roztáhne podle rozlišení monitoru a roztažení samotného okna simulačního programu. Po startu simulačního programu se vlak zobrazí na středu obrazovky. To nastane v případě, kdy program nenalezl databázi souřadnic a je nachystán k vytvoření nových souřadnic. Pokud by databázi našel, vlaky budou stát na svých zastávkách. V obou případech je možná vytvořit novou databázi

pomocí okna na Obr. 16, které najdeme v simulačním programu pod nabídkou Soubor/Nastavení, nebo pod klávesovou zkratkou ctrl+N.



Obr. 15: Vzhled modelu při vytváření databáze



Obr. 16: Nastavení dráhy pohybu vlaku

Oknu dominuje velká bílá plocha, která slouží jako přehled vložených souřadnic, které nereprezentují pouze souřadnice, po který se budou vlaky pohybovat, ale také souřadnice udávající polohu semaforů a návěští. V neposlední řadě i řídicí hodnoty pro výhybky.

Nad bílou plochou jsou dva sloupce, kde první je označen jako Dráha. Ten slouží k vytvoření přímého pohybu vlaku, zastávek, návěští a semaforů.

Pokud budeme zadávat úseky pro dráhu, pak první dvě kolonky určují dva body, mezi kterými se má vlak v daném úseku pohybovat. Protože by bylo složité body vypisovat a odhadovat požadovanou pozici bodů, je program vytvořen tak, že stačí do okna, ve kterém je vykreslen obrázek trati, pouze klikat a souřadnice se do kolonky sami zapíší. Druhá kolonka se vyplní automaticky až po druhém kliknutí. Tedy v první kolonce „Odkud“ máme poslední klik a v kolonce „Kam“ máme předchozí klik. Toto je uděláno z předpokladu, že uživatel trať vyznačí po delších úsecích a s velkou pravděpodobností by se do stejného místa už podruhé netrefil. Druhou možností by bylo body přepisovat, ale tato varianta by nebyla příliš výhodná. Současně stačí kliknout na další požadovaný bod a potvrdit pár bodů, který se vkopíruje do seznamu umístěného v dolní části okna nastavení. Následné pole slouží k označení úseku. Pro úsek nemusíme vpisovat znak #, který je uveden u všech prvků modelu. Tento znak se vkládá automaticky. Výjimkou je vložený zastávky, kdy se jeden znak # před písmenem značící zastávku, ale u úseku je třeba tento znak dopsat. Zápis by mohl vypadat P0,#S1. Na české klávesnici se znak # skrývá pod klávesovou zkratkou Alt Gr+X. Všechny 3 pole jsou povinné pro zapsání do seznamu.

V těchto kolonkách se nastavují i souřadnice pro semafory a návěští. Do kolonky, „Odkud“ se dávají souřadnice, kde má být semafor zobrazen. Do kolonky „Kam“ se zapisují souřadnice bodu, na kterém budou vlaky zastavovány. Posledním prvkem programu je návěští, které se nepatrně liší od výše popsaných prvků. Zde jsou povinné jen 2 vyplněné kolonky a to „Odkud“, kde jsou souřadnice vykreslovaného návěští, a druhou kolonkou je úsek, kde je označení návěští.

Druhým sloupcem v okně je „Výhybka“. Zde je potřeba propojit krajní body vyklikaných křivek. Do první kolonky se zapíše společný bod, ze kterého se vlak může rozjet do dvou směrů. Druhé a třetí označuje body, ze kterých se bude vlak sjíždět do společného bodu. Čísla 0 a 1 označují, jaké logická je potřeba nastavit, aby se vlak vydal požadovaným směrem.

Pod oběma sloupci je tlačítko pro odsouhlasení souřadnic. Tlačítko platí pro každý sloupec zvlášť.

Protože by bylo velmi nepřehledné, kde uživatel souřadnice již nastavil a kde nikoliv, jsou do obrazu vkreslovány čáry a kruhy, které zastupují již odsouhlasené úsek, zastávky, semafory a návěští, tak jako je to znázorněno na Obr. 15.

Pro rozlišení jednotlivých částí jsou jednotlivé prvky odlišeny barevně

Tab. 4: Barevné rozlišení čar při návrhu tratě

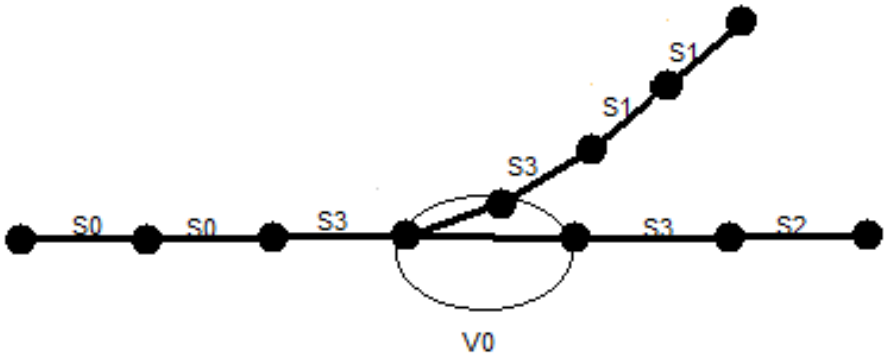
úsek	modrá
výhybka	červená
zastávka	žlutá

Tab. 5: Barevné rozlišení kruhů při návrhu tratě

semafor	červená
návěští	žlutá

Po dokončení nastavování souřadnic je nutné databázi uložit tlačítkem pod seznamem. Tím se změny uloží do souboru souradnice.txt. Veškeré změny se projeví až po restartu programu a zároveň po restartu zmizí i pomocné čáry a kružnice.

Model má tu vlastnost, že výhybky přebírají obsazení úseku, z kterého vlak přijíždí. U navržené trajektorie kolejiště může zabírat výhybka velmi malou plochu a model vlaku může potom zasahovat jak do oblasti výhybky, tak do oblasti úseku. Řešením je použití na všech koncích výhybky stejný úsek. Může se tím docílit i vizuální stránky, kdy by pro model bylo vše v pořádku, ale vizuálně by se vlaky překrývaly. Může se použít nový úsek pouze pro výhybku viz Obr. 17, nebo se může použít název úseku pro jeden z konců výhybky.



Obr. 17: Řešení návrhu úseku kolem výhybek

4.5 Nastavení modelu

K nastavení modelu slouží soubor confii.txt, který je umístěn ve stejné složce, jako program a databáze.

Prvním parametrem je Port1, který slouží k nastavení COM portu, pomocí kterého se bude program připojovat k převodníku, respektive k PLC. Pokud v souboru není tento prvek nastaven, pak je nastaven jako Port1=COM1.

Druhým parametrem je počet vlaků. Tato hodnota by měla být menší než je počet zastávek. Pokud by bylo číslo nastaveno větší, pak by se tato chyba projevila v simulaci vlakem, který bude stát ve středu obrazovky. Pokud by v souboru nebyl tento prvek obsažen, pak by byl počet vlaků nastaven jako VLAKU=2.

Následující parametry slouží k nastavení velikosti objektů. Nastavit se dá velikost vlaků, semaforů a návěští. Pro nastavení velikosti vlaků slouží parametr, který je defaultně nataven na VLAK_POM=1. Budeme-li chtít vlak o polovinu menší, nastavíme parametr na hodnotu 0.5. Stejně se dají nastavit semafory a návěští parametrem SEM_POM=1.

Posledním parametrem je nastavení rychlosti vlaků. Pro toto nastavení slouží parametr RYCHLOST_VLAKU=1.2. Defaultní nastavení je opět 1. V ukázce parametru je zvýšení rychlosti o 20%.

4.6 Kolize při simulaci

Program zastavuje simulaci ve 2 případech:

- Kolize na úseku, kdy se dva vlaky objeví na stejném úseku.
- Kolize na výhybce, kdy vlak najede na výhybku ze směru, na kterého není výhybka přehozena

V obou případech se simulace zastaví až do příjmu signálu restart.

4.7 Komunikace mezi PLC a modelem

4.7.1 Převodník

Jedinou funkcí převodníku je na vyžádání modelu nastavit výstupní napěťové úrovně na základě přijatých dat a zároveň posloupnost logických úrovní ze svých vstupů odeslat modelu.

4.7.2 Komunikace modelu

Model komunikuje s převodníkem automaticky a nezáleží na právě zvoleném modelu. Uživatel mimo výše uvedených nastavení nemusí do další komunikace zasahovat. Model má k dispozici 4 digitální karty, z nichž 2 vstupní a 2 výstupní. Každá karta má 8 vstupů/výstupů. Celkem tedy 16 vstupů a 16 výstupů.

4.7.2.1 Vstupy modelu

Pro každý port výstupu PLC je v komunikaci vyhrazen 1 bit. První bit0 je vyhrazen pro restart modelu, kdy se všechny vlaky přesunou do stanic. Následující bity jsou řazeny v pořadí:

- Povolení výjezdu z nádraží (Start0, Start1, …)
- Směr natočení výhybky (V0, V1, …)
- Semafory (Z0, Z1, …)

Prioritu v pořadí mají příkazy s nižším číslem. Na Obr. 15 nejsou vypsána označení jednotlivých zastávek a výhybek, ale je vidět, že jsou zde 3 zastávky a 3 výhybky.

V následujících tabulkách bude znázorněn přenos jednotlivých vstupních hodnot.

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
-	-	-	-	-	-	-	-

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
-	V2	V1	V0	Start2	Start1	Start0	RESTART

4.7.2.2 Výstupy

Podobně jako vstupy jsou odesílány informace pro PLC. První bit slouží k potvrzení restartu. Následné hodnoty informují o obsazení jednotlivých úseků. Pokud bude mít trať 8 úseků S0-S7, pak by odesílané 2 Byte vypadaly následovně.

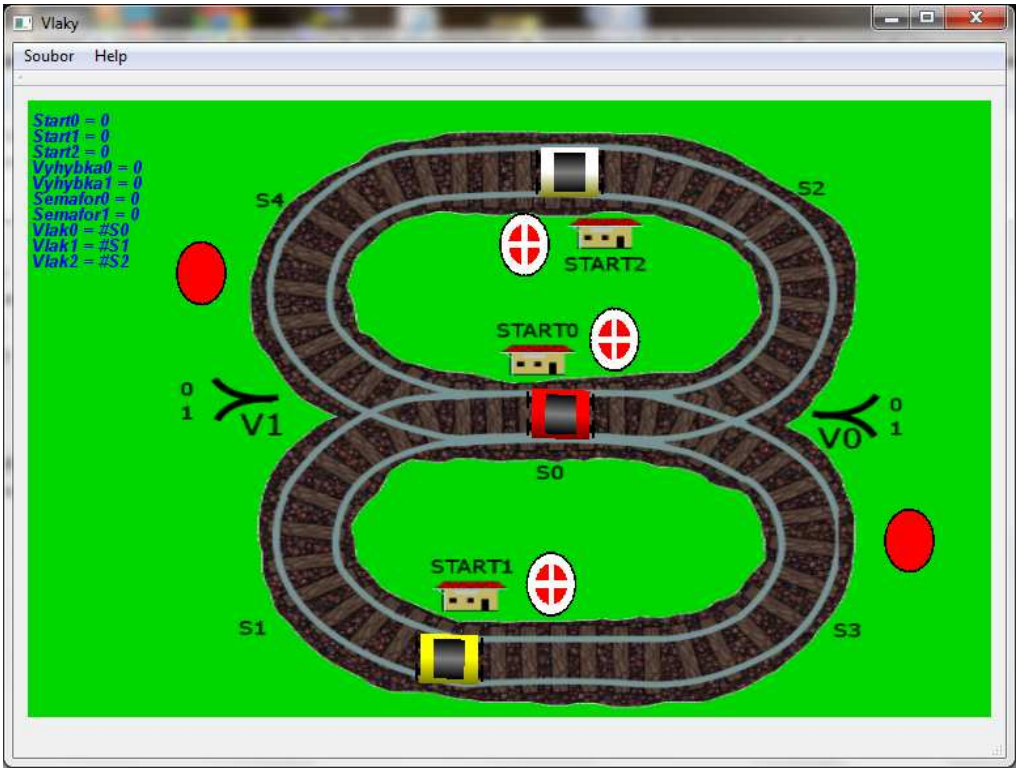
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
-	-	-	-	-	-	-	S7

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
S6	S5	S4	S3	S2	S1	S0	Potvr. R

4.7.3 Výsledný vzhled modelu

Na Obr. 18 je možné vidět jednu z možných variant modelu po zapnutí nebo po přijetí signálu restart. V levém sloupci je seznam vstupních proměnných, které model řídí. Pod nimi je informace o obsazení úseků vlaky.

Vlaky jsou umístěny na nádraží a jsou zde vykresleny i semafory a návěští u nádraží.

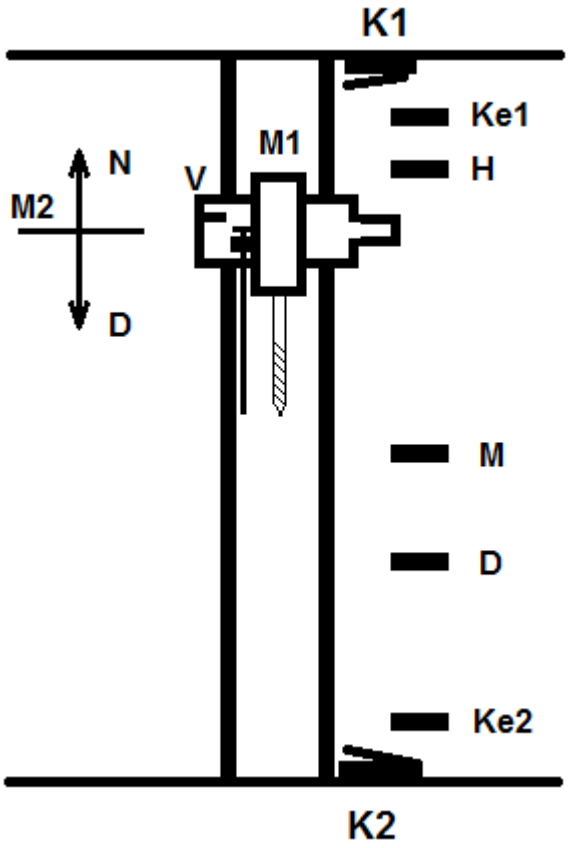


Obr. 18: Výsledný vzhled jedné varianty modelu

5 MODEL VRTAČKY

Druhým modelem je fyzický model vrtačky. Tímto modelem jsem se nechal inspirovat na hodině laboratoří. Oproti viděné verzi je tento model doplněn o další 4 snímače, které by měly umožnit ovládání modelu vzdáleně, bez nutnosti do něj fyzicky zasahovat.

Hlavním konstrukčním prvkem modelu je lineární vedení, na kterém je uchycen rotor M1, který simuluje vrtací hlavu. Pohyb vrtací hlavy je zprostředkován pomocí motoru M2 a to ve směrech N a D. Tento pohyb je zprostředkován pomocí řemenice, která je přichycena k vrtací hlavě. U motoru M1 je umístěn snímač V. Tento snímač hlídá tyčku, která je umístěna vedle vrtací hlavy. V případě, že tyčka narazí do materiálu, snímač informuje řídicí PLC změnou logické úrovně. Zároveň jsou k dispozici snímače polohy H, M a D. Kombinací těchto snímačů může student zjistit velikost vrtaného materiálu. Pokud snímač V detekuje materiál dříve, nežli dojede vrtací hlava ke snímači M, pak je třeba materiál vrtat nadvakrát. Tedy vrtací hlava sjede do úrovně M, poté vyjede z materiálu do bodu H a poté dovrtá zbytek materiálu dojetím do bodu D a opět se vrátí zpět do bodu H.



Obr. 19: Schéma modelu vrtačky

Druhou možností vrtání je, že vrtací hlava bude sjíždět ve směru D a projede úrovní M dříve, než dojde k indikaci materiálu snímačem V. Poté nebude probíhat vrtání nadvakrát, ale pouze jednou, kdy vrtací hlava dojede do úrovně D a opět vyjede nahoru do úrovně H.

Popsaný postup je pro ideální řízení modelu. Může ale dojít k situaci, kdy se, vlivem chybného programu, začne vrtací hlava pohybovat mimo úsek definovaný úrovněmi H a D. Z tohoto důvodu jsou za krajními body ještě snímače Ke1 a Ke2. Tyto snímače nemá student k dispozici. V případě jejich dosažení dochází k ignorování příkazů zasílaných od PLC a model sám dojede na úroveň H, kde čeká na další simulaci. Pro spuštění modelu je potřeba zaslat prvně příkaz RESTART, který je v případě dosažení úrovně H potvrzován signálem POTV_REST. Signál RESTART tedy neslouží pouze k odblokování řízení motoru M2, ale taky pro defaultní natavení polohy vrtací hlavy. Toto potvrzování je z důvodu stále běžícího chybného algoritmu v PLC, který by stále posílal vrtací hlavu mimo pracovní prostor.

Poslední zobrazené prvky jsou koncové snímače K1 a K2. Slouží pouze jako pojistka, která zajišťuje bezpečnost modelu, kdyby náhodou došlo k poruše snímačů Ke1 a Ke2. Jedná se o rozpínací kontakty, které okamžitě odpojí motor M2 od napětí.

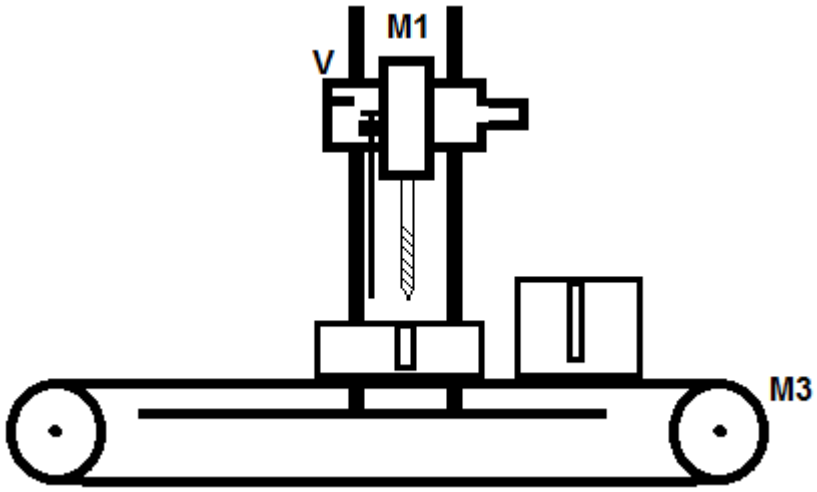
5.1 Řízení vrtačky

Řízení motoru M2 se provádí pomocí 2 logických signálů. Jeden spouští posun a druhý nastavuje směr posunu. Logická 0 posun směrem D a logická 1 směrem N. Signál M1 slouží pro zapnutí motoru simulujícího vrtací hlavu. Pro řízení modelu má uživatel k dispozici signály pro polohu vrtací hlavy H, M a D.

Dalším signálem je V, který má indikovat výšku materiálu. Protože by měl být model obsluhován vzdáleně, nemá účastník možnost zaměňovat vrtaný materiál. Toto by se dalo řešit například výměnou materiálu obden, ale tento postup by byl nepohodlný. Proto je modelem tento signál zároveň i generovaný. Uživatel si může vybrat, jestli bude zpracovávat:

- svoji konstantu, kterou si před simulací nastaví
- snímač S
- generovaný signál Sm od modelu, který po každém dojetí na úroveň D a zpět na úroveň H, změní pro následující běh programu velikost materiálu.

Další nerealizovanou možností, je rozšíření modelu o pás.



Obr. 20: Rozšíření modelu o pás

Pro studenta by mohlo dojít k rozšíření o jeden řídicí signál, kde by mohla logická 0 znamenat nízký materiál a logická 1 vysoký materiál. Na Obr. 20 je situace znázorněna. Objekty mohou být umístěny vedle sebe a posunutím objektů pásem do leva dojde k jejich záměně pod vrtací hlavou.

6 NÁVRH ELEKTRONIKY

6.1 Stejnosměrné motory [11]

Jako pohony jsou použity stejnosměrné motory. Princip motoru funguje na základě odpuzování dvou magnetických polí, která mají stejnou polaritu.

Motory se skládají z nepohyblivé části, které se říká stator a z pohyblivé části, které se říká rotor. U stejnosměrných motorů, s buzením permanentními magnety ve statoru, je rotor tvořen cívkami, které jsou uloženy v drážkách, a jsou vyvedeny na komutátor. Komutátor je zařízení, které slouží k přenosu elektrické energie do cívek rotoru. Tento přenos je proveden pomocí kartáčů, ke kterým je přivedeno napájecí napětí. Na statoru jsou po obvodu navzájem izolované vodivé lamely připojené k cívkám. Na tyto lamely doléhají kartáče, které jsou obvykle z grafitu (uhlíky).

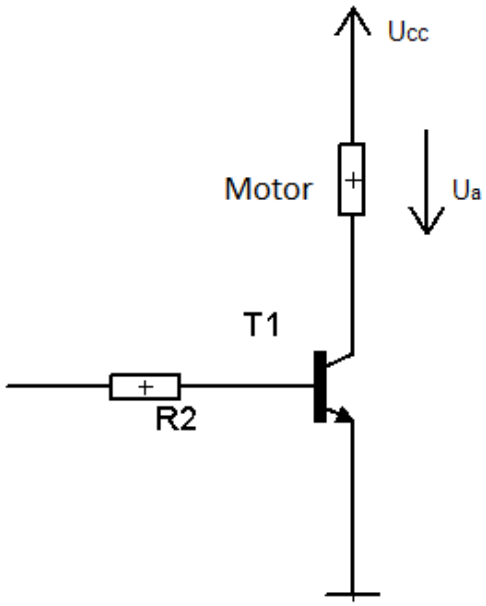
Stator vytváří magnetické pole, které se neotáčí. Oproti tomu průchodem proudu smyčkou rotoru vzniká kolem vodiče elektromagnetické pole, na něž působí pole statoru. Podmínkou, aby se motor točil, je odpuzování obou polí. Tuto funkci zajišťuje komutátor, který přivádí proud do smyčky vždy v takovém směru, aby elektromagnetické pole kolem vodiče bylo stále stejně orientováno, jako pole magnetické.

6.1.1 Řízení otáček stejnosměrného motoru

Otáčky motoru ω jsou přímo úměrné napětí U_a, které je připojeno ke komutátoru stejnosměrného motoru, tak jak to vyplývá ze vztahu závislosti otáček stejnosměrného motoru.

$$\omega = \frac{U_a}{(C\Phi)} - \frac{R_a}{(C\Phi)^2} M_z$$

Pro stejnosměrný motor s buzením permanentními magnety je CΦ napěťovou a momentovou konstantou motoru. Hodnotou R_a je odpor vinutí kotvy a M_z je zatěžovací moment motoru. Řízení otáček pomocí R_a by bylo možné, ale zbytečně by na odporu vznikaly ztráty. Proto se otáčky stejnosměrného motoru řídí napětím.



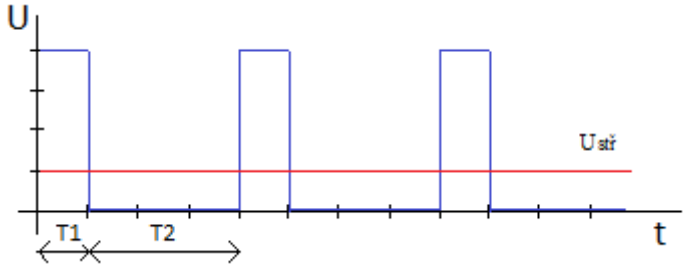
Obr. 21: Řízení rychlosti motoru

Na Obr. 21 je znázorněno zapojení pro jednoduché řízení stejnosměrného motoru. Pokud je tranzistor řízen pootevíráním, docházelo by k úbytkům napětí nejen na samotném motoru, ale i na tranzistoru, který by se začal zahřívat, a tím by na něm vznikaly ztráty. Vzniklými ztrátami by se začaly výkonové tranzistory zahřívat a vzniká tím i potřeba tranzistory více chladit.

6.1.2 Řízení otáček pomocí PWM

Řešením těchto problémů je PWM (pulzní šířková modulace). PWM vytváří signál, který má po čas T1 velikost U1 a po čas T2 velikost 0 V. Princip spočívá na rychlé změně řídicího signálu tranzistoru. Protože má motor setrvačnost a spínací frekvence je oproti ní vysoká, pak rotor nestíhá reagovat na změny napájení. Výsledkem je, že se motor chová tak, jako by byl napájen napětím o velikosti střední hodnoty napájecího napětí.

Ukázka takového řízení je na Obr. 22. Motor je připojen na střídavé kladné napětí, ale motor se chová jako by byl připojen pouze ke střední hodnotě. Hlavní výhodou je, že nedochází k velkým ztrátám na tranzistorech, protože ty jsou zcela otevřené a je na nich pouze saturační napětí a nebo jsou zcela uzavřeny.



Obr. 22: Řízení otáček motoru

Velikost napětí lze určit jako:

$$U_{stř} = \frac{T_1}{T_1 + T_2}.U_{\max} \qquad (1)$$

Pro řízení se může používat i pojem střída. Pojem střída je u periodických signálů, kde dochází během periody k přecházení z jedné úrovně do druhé. Tedy poměr časů T1 a T2. Střída může být udávána v poměru, tady obdélníkový signál, kde časy T1 = T2, má střídu 1:1. Zároveň může být střída udávána v procentech. Myslí se tím obvykle procentuelně čas, po který je časový signál v úrovni sepnuto proti celé periodě. Příklad výpočtu je ve vztahu (2):

$$D = \frac{T_1}{T_1 + T_2} = \frac{T_1}{T} \qquad (2)$$

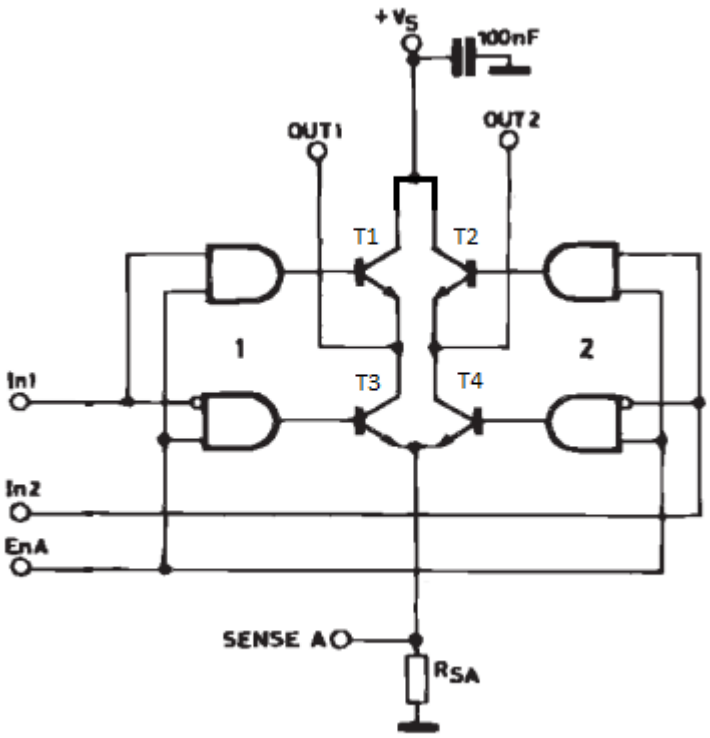
6.1.3 Řízení směru otáčení motoru [2]

Změna směru otáčení se provádí prohozením polarit napájení na svorkách motoru. Pro tyto účely se používá H-můstek. Na Obr. 23 je zobrazeno jedno z možných zapojení i s řídicí elektronikou. Pro otáčení motoru je potřeba docílit otevření tranzistorů T1 a T4 nebo T2 a T3 pro opačný směr otáčení hřídele motoru. Z bezpečnostních důvodů se používá řídicí elektronika můstku, která je řešena jednoduchou hradlovou logikou s hradly AND. Ta má za úkol zajistit, aby nedošlo k současnému sepnutí tranzistorů na pravé nebo levé straně, čímž by došlo ke zkratu. Druhou výhodou je snížení počtu signálů, které by bylo třeba přivést k můstku a zároveň takovýto můstek nezabere tolik místa, jako bychom stavěli celý řídicí obvod.

V tomto zapojení je zvláště ovládaná levá strana vstupem In1 a zvláště pravá strana vstupem In2. Naše zapojení zajišťuje otevření pouze jednoho tranzistoru na každé straně. Vstup EnA slouží k povolení otáčení motoru.

Stav, kdy jsou otevřeny tranzistory T1 a T2 nebo T3 a T4 se nazývá brzda.

Toto zapojení nabízí další rozšíření řídicích obvodů. Je možné spojit pomocí negace vstupy In1 a In2 a vstupem EnA řídit otáčky motoru. Jinou možností je trvale připojit EnA k napájení.



Obr. 23 Řízený H-můstek[2]

Podobné obvody, jako je znázorněno na Obr. 23 jsou prodávány jako integrované obvody i s řídicí elektronikou. U některých obvodů jsou vyvedeny podobné řídicí obvody, jako je tomu na obrázku, jiné jsou vybaveny komunikační sběrnicí, po které jsou posílání požadavky na směr a rychlost otáčení.

Pro řešení úlohy jsem zvolil možnost, kdy je řízen vstup In1 a In2=~In1. Tedy na In2 je invertovaný signál. Pro příklad. Pokud mám napájecí napětí 12V a na In1 přichází signál, který generuje na jedné svorce motoru střední hodnotu napětí 2V, pak na druhé svorce bude střední hodnota napětí 10V, pokud nebudeme uvažovat žádné ztráty na můstku. Samozřejmě je nutné, aby byl vstup EnA připojen trvale k napájení, nebo musí být přivedena na jeho vstup logická 1.

6.2 Řídicí obvod

6.2.1 Mikrokontrolér

Pro řízení obou úloh byl vybrán mikrokontrolér STM32F100R4T6B. Tento procesor patří do rodiny ARM. Kromě mnoha obvyklých funkcí, na jaké jsme zvyklí u hodně používaných procesoru Atmel je zde vypracována struktura připojení hodin, kdy lze ke každému bloku procesoru nastavit hodiny. Další patrnou odlišností je spouštění jednotlivých části mikrokontroléru. Pokud nedojde ke spuštění požadované části, pak ani přivedení hodin nepomůže a požadované zařízení nebude pracovat. To může být pro programátora nepříjemné, ale ARM pracuje úsporně a nepotřebné části, které nejsou používány, nepřipojí k napájení a ty pak neodebírají elektrickou energii.

Tento procesor má 32 bitů, obsahuje5 bran po 16-ti pinech. Na některých lze nalézt 7 PWM, které jsou 16-ti bitové. Pro komunikaci s okolím používá I2C, SPI a USART. Pro analogový výstup lze využít 7 kanálů DA převodníků. Dále v procesoru je 17 TIMERů. Některé mají omezené využití, ale některé jsou všestranné pro běžné aplikace, převodníky, čítače a další.

6.2.2 Využití mikrokontroléru

Řídicí systém je realizován pomocí dvou řídicích desek vybavených mikrokontrolérem.

6.2.2.1 Využití u převodníku pro vláčkoviště

V úloze s vláčky je jeho úlohou číst a nastavovat logické úrovně na vstupy a výstupy PLC. Při příjmu zprávy nastaví převodník na své výstupy, tedy na vstupy PLC, požadované hodnoty a poté přečte výstupní hodnoty PLC na svých vstupech a odešle je modelu. Komunikace mezi PC s převodníkem je obstarána pomocí USB. Rozhraní převodníku obstarává obvod FT232, který s mikrokontrolérem komunikuje pomocí USART.

6.2.2.2 Využití pro řízení vrtačky

V úloze s vrtačkou obstarává mikrokontrolér mezičlánek mezi řízením motoru pro pohyb vrtací hlavy a PLC. Mikrokontrolér pracuje ve 2 režimech. V prvním reaguje na řídicí signály od PLC, které promítá v modelu. Druhý režim může být označován jako porucha, nebo přechod do defaultního stavu. V případě příkazu restart, nebo přejetí krajních pozic lineárního vedení, dojde k automatickému přesunu vrtací hlavy do bodu H.

6.2.3 Návrhy schémat

Protože mikrokontrolér pracuje s napětím 3,3 V a karty PLC s napětí 24 V, je potřeba zajistit převod mezi těmito úrovněmi. Za tímto účelem jsou mezi vstupy a výstupy vloženy u obou schémat optrony. Rozdílem mezi obvody, při pohledu na digitální vstupy a výstupy, je pouze množství vstupů a výstupů. Navržená schémata zapojení, desky plošných spojů a jejich osazení je v příloze.

6.2.3.1 Schéma pro převodník

V případě převodníku obsahuje obvod komunikační rozhraní USB zajištěné obvodem FT232. Největší rozdíl mezi obvody zajišťuje především software, který je v příloze na CD. Navržené schéma je v Příloze 5, navržená deska plošných spojů v Příloze 6 a rozmístění součástek v Příloze 7. Celý návrh je ještě uložen na přiloženém CD v projektu Eagle.

6.2.3.2 Schéma pro vrtačku

Pro vrtačku obsahuje návrh 2 plošné spoje, protože je částečně využito obvodu ze semestrální práce, který byl postaven s odlišným cílem. První, již popisovaný, obvod slouží k řízení a komunikaci s PLC. Druhý obvod slouží k obsluze stejnosměrného motoru. Hlavním obvodem celého návrhu je integrovaný H-můstek L298. Obvod byl vybrán především proto, že s rezervou splňoval parametry pro potřeby motoru. Výstupní napětí můstku může být až 46V a může jím protékat proud 4A. Obvod obsahuje 2 plné můstky. Především je určen pro řízení krokových motorů nebo se dá využít k proudovému posílení můstku tím, že jsou oba můstky připojeny k jednomu motoru. Tento můstek byl vybrán nejen podle parametrů, ale i podle dostupnosti a ceně. Většina srovnatelných můstků není u prodejců skladem nebo jsou pouze na objednávku.

Schéma obsahuje ještě 2 obvody sloužící pro měření proudu procházejícím motorem. Tyto obvody ale nejsou pro model zapotřebí, protože sloužili pro polohovou regulaci. Pro tuto úlohu stačí nastavená střída výstupního napětí.

Návrhy zapojení řídicího obvodu je v Příloze 8, deska plošných spojů v Příloze 9 a osazovací plán v Příloze 10. Pro návrh zapojení H-můstku je schéma zapojení v Příloze 4, návrh plošného spoje v Příloze 2 a osazovací plán v Příloze 3.

Program pro mikrokontrolér je opět na přiloženém CD.

7 ZÁVĚR

Touto prací vznikly 2 modely, které je možné připojit k systému LabLink a provozovat je bez nutné fyzické přítomnosti přímo u modelů.

Prvním z modelů je model vláčkoviště, který částečně simuluje provoz vlaků na trati. Model obsahuje vlakové zastávky, semafory, výhybky a především reaguje na kolize vlaků na dráze. Tento model je softwarový a může pracovat na běžném stolním PC. Předností softwaru je možnost vytvořit libovolnou trať a provádět v ní změny bez nutnosti nové překladu zdrojových kódů. Pro tvorbu navrhované tratě bylo vytvořeno dialogové okno, které se snaží tvorbu zjednodušit. Návrh topologie trati je pouze pro pohyby vláčků. Pozadí, na kterém jsou znázorněny koleje a vlakové zastávky, je tvořen obrázkem, který je možné měnit. Model je řízen PLC Wago, který má připojeny 4 digitální karty, které ovládají pohyb vlaků. Další možností je řízení pohybu pomocí Control Webu. Ke komunikaci mezi PLC WAGO 750 a systému Control Web bylo zvoleno OPC, i když bylo možné využít MODBUS TCP/IP. Důvody k této volbě byly dva. Prvním je, že MODBUS TCP/IP již byl použit při přípravě jiného modelu a chtěl jsem připravit jiný způsob komunikace. Druhým důvodem je nižší pořizovací cena ovladače „OPC klient pro Control Web“.

Druhý model je fyzický a má simulovat automatickou vrtačku s rozlišováním výšky materiálu. Podle výšky dochází ke způsobu vrtání, kdy pro nízké materiály vrtačka provrtá materiál najednou a pro vysoké nadvakrát. Vrtačka je opět řízena PLC Wago a obsahuje elektroniku, která zajišťuje návrat vrtací hlavy do defaultní polohy v případě uživatelského příkazu, nebo najetí vrtací hlavy mimo definovaný prostor.

Literatura

[1] Dokumentace Control Web 6.1 SP5

[2] L298 [online], [cit. 2010-12-18],Dostupné z http://www.gme.cz/_dokumentace/dokumenty/332/332-005/dsh.332-005.1.pdf

[3] CoDeSys V2.3 Programmin Systém [online], [cit. 2011-05-05], Dostupné z http://www.3s-software.com/index.shtml?en_oem1

[4] Manuál pro programování PLC v prostředí CoDeSys 2.3, verze 1.0, 2005

[5] Vacek, Roman, *Grafický panel 844-8070 Instalace a nastavení ethernet komunikace*, 2010/06

[6] WAGO-I/O-SYSTÉM 750 Modbus/TCP OPC server, version 1.1.0

[7] Dokumentace komponent systému Control Web 6, OPC ovladač pro Control Web

[8] Qt Licensing, [online], [cit 2011-04-23], Dostupné z http://qt.nokia.com/products/licensing/

[9] Programming Language Support [online], [cit 2011-04-23] http://qt.nokia.com/products/programming-language-support

[10] Supported Platforms [online], [cit 2011-04-23], Dostupné z http://doc.qt.nokia.com/4.7/supported-platforms.html

[11] Skalický, Jiří, *Elektrické regulované pohony*, 2007, 123s

[12] Fiala, O., Burget, P., Fencl, T. , *Popis řešení projektu za období 07/2006 06/2010*, Vzdálené laboratoře, virtuální a fyzické modely

[13] Komponenty, [online], [cit 2011-04-25], Dostupné z http://www.wago.com/cps/rde/xchg/SID-2CB96107-BDCEDA29/wago/style.xsl/csy-1855.htm

[14] Instruction List (IL), [online], [cit. 2011-05-05], Dostupné z http://www.3s-software.com/index.shtml?en_CoDeSys_IL

[15] Modulární systém průmyslových vstupů/výstupů DataLab IO, [online], [cit. 2011-05-19], Aktualizováno: 6.1.2009, Dostupné z http://www.mii.cz/art?id=134&cat=77&lang=405

[16] Qt, [online], [cit 2010-04-18], Dostupné z http://qt.nokia.com/products/

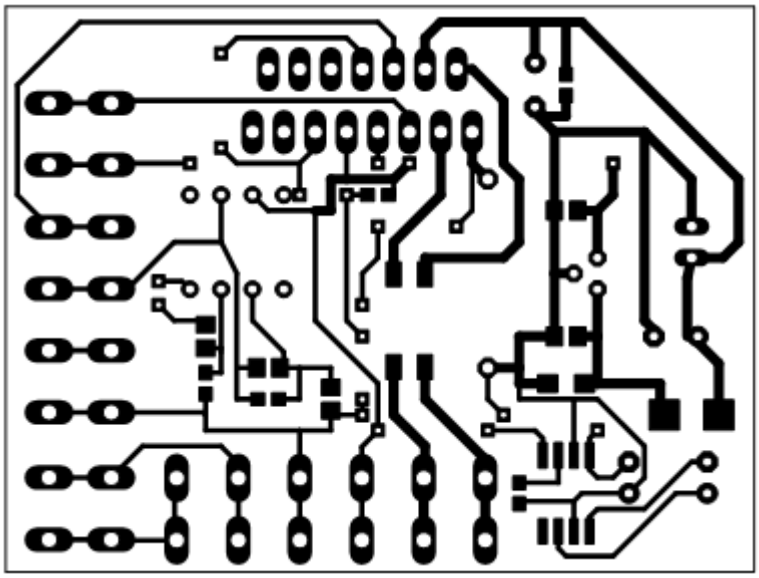
[17] Tucek, Michal, *NeHe OpenGL Tutoriály*, [online], 29.02.2004, Dostupné z http://www.root.cz/knihy/nehe-opengl-tutorialy/

Seznam zkratek

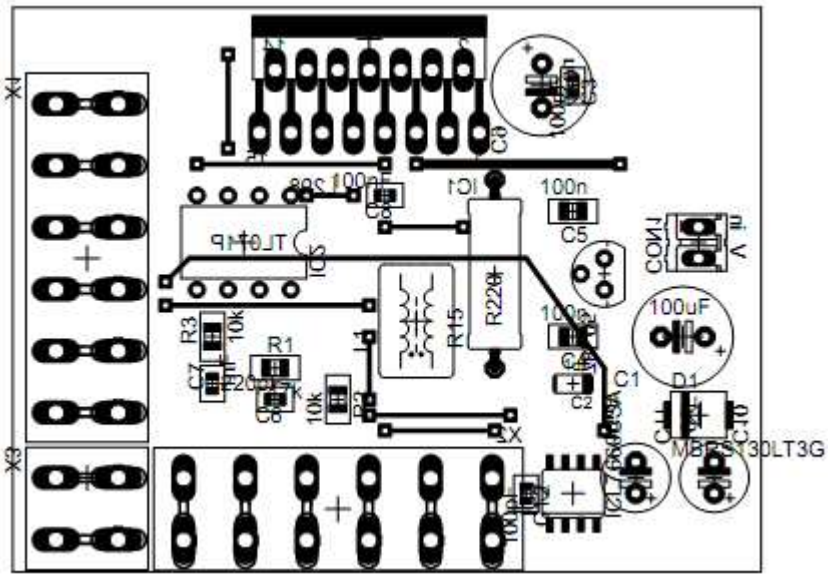
GUI	- Graphical user interface
IDE	- Integrated Development Environment (integrované vývojové prostředí)
OpenGL	- Open Graphics Library
PLC	- Programmable Logic Controller (programovatelný logický automat)
CPU	- Central Procesor Unit
SCADA	- Supervisory control and data acquisition
OLE	- Object Linking and Embedding
OPC	- OLE for Process Control
VLAN	- Virtual Local Area Network
PCI	- Peripheral Component Interconnect
USB	- Universal Serial Bus
TCP/IP	- Transmission Control Protocol / Internet Protocol
HTTP	- Hypertext Transfer Protocol
FTP	- File Transfer Protocol
HTML	- HyperText Markup Language
XML	- Extensible Markup Language
LD	- Ladder Diagram
FBD	- Function Block Diagram
IL	- Instruction List
ST	- Structured Text
SFC	- Sequential Function Chart
DC	- Direct Current
DMF	- Driver Map File

Přílohy

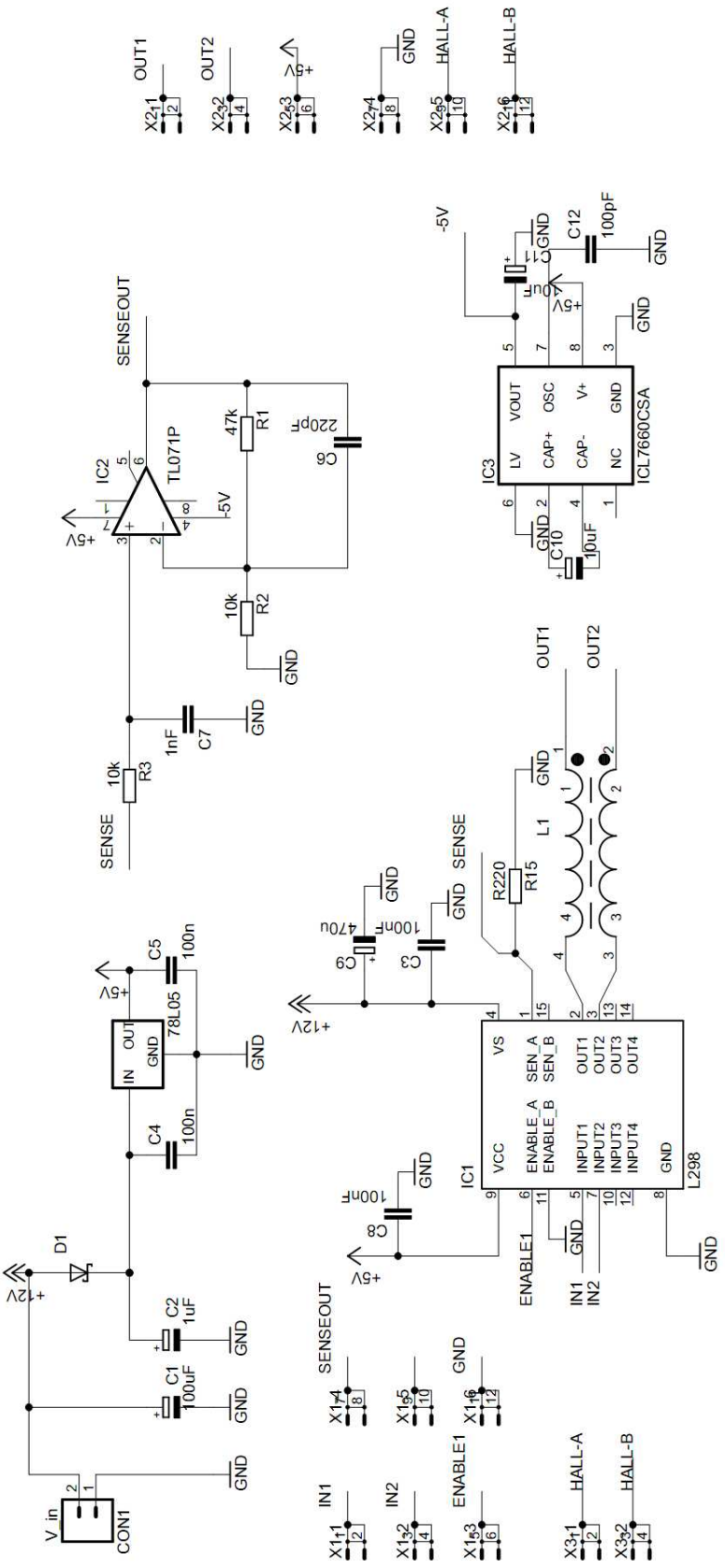
Příloha 1: CD



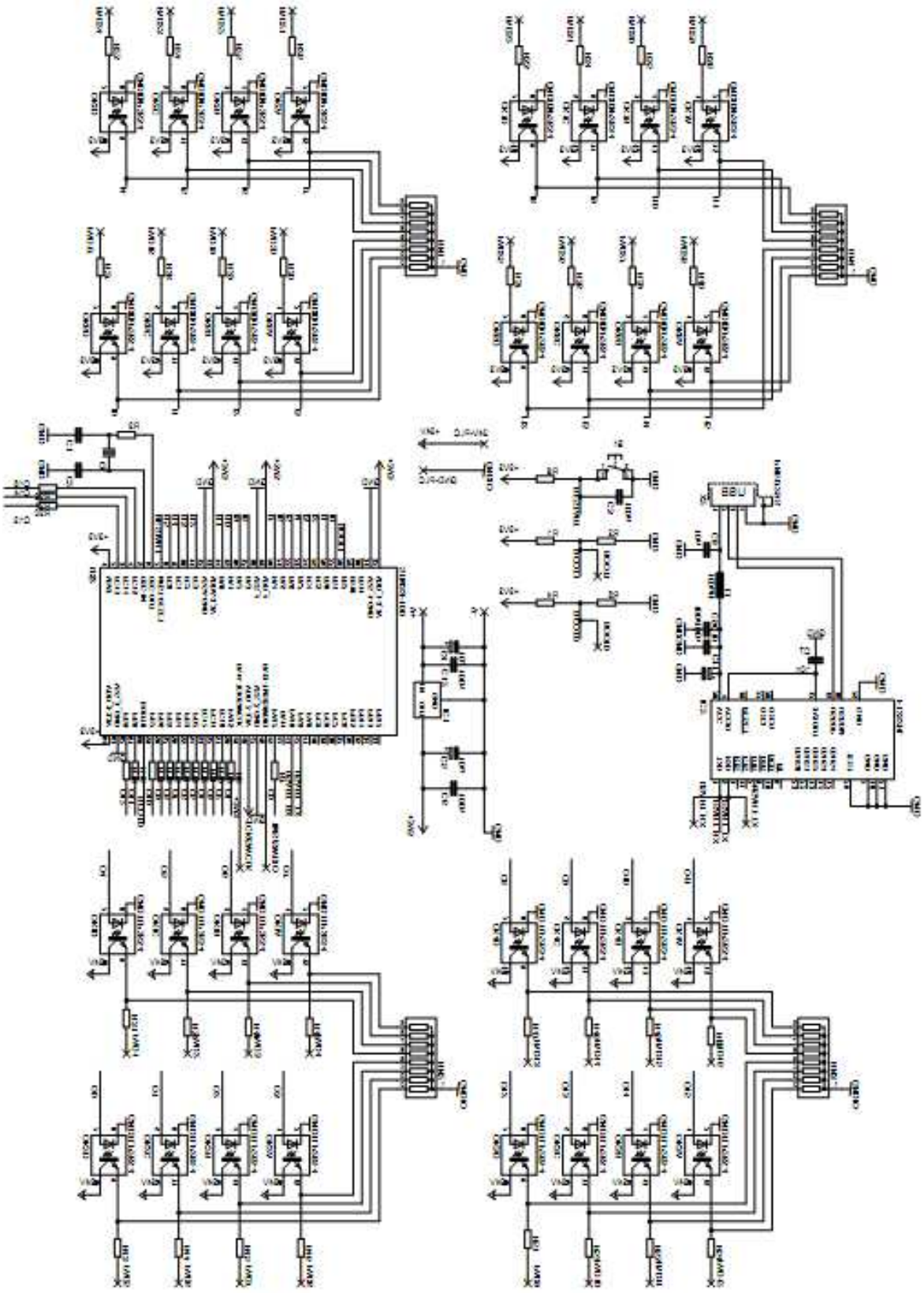
Příloha 2: Návrh plošného spoje pro řízení motoru



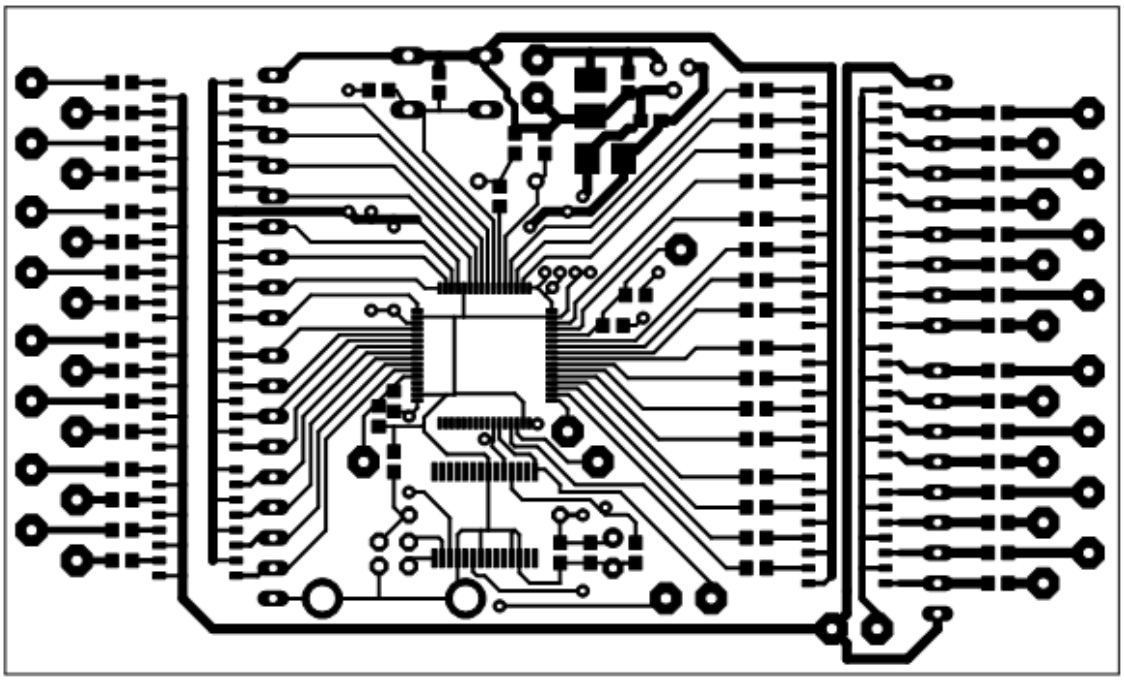
Příloha 3: osazení plošného spoje pro řízení motoru



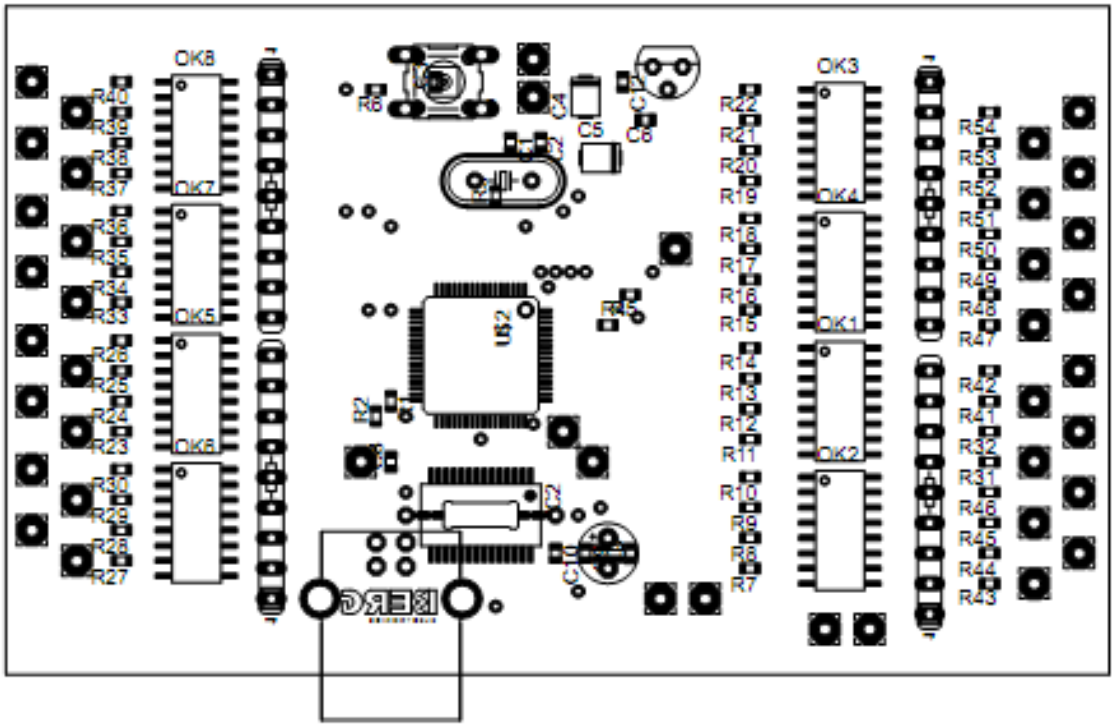
Příloha 4: Schéma zapojení obvodu pro řízení motoru



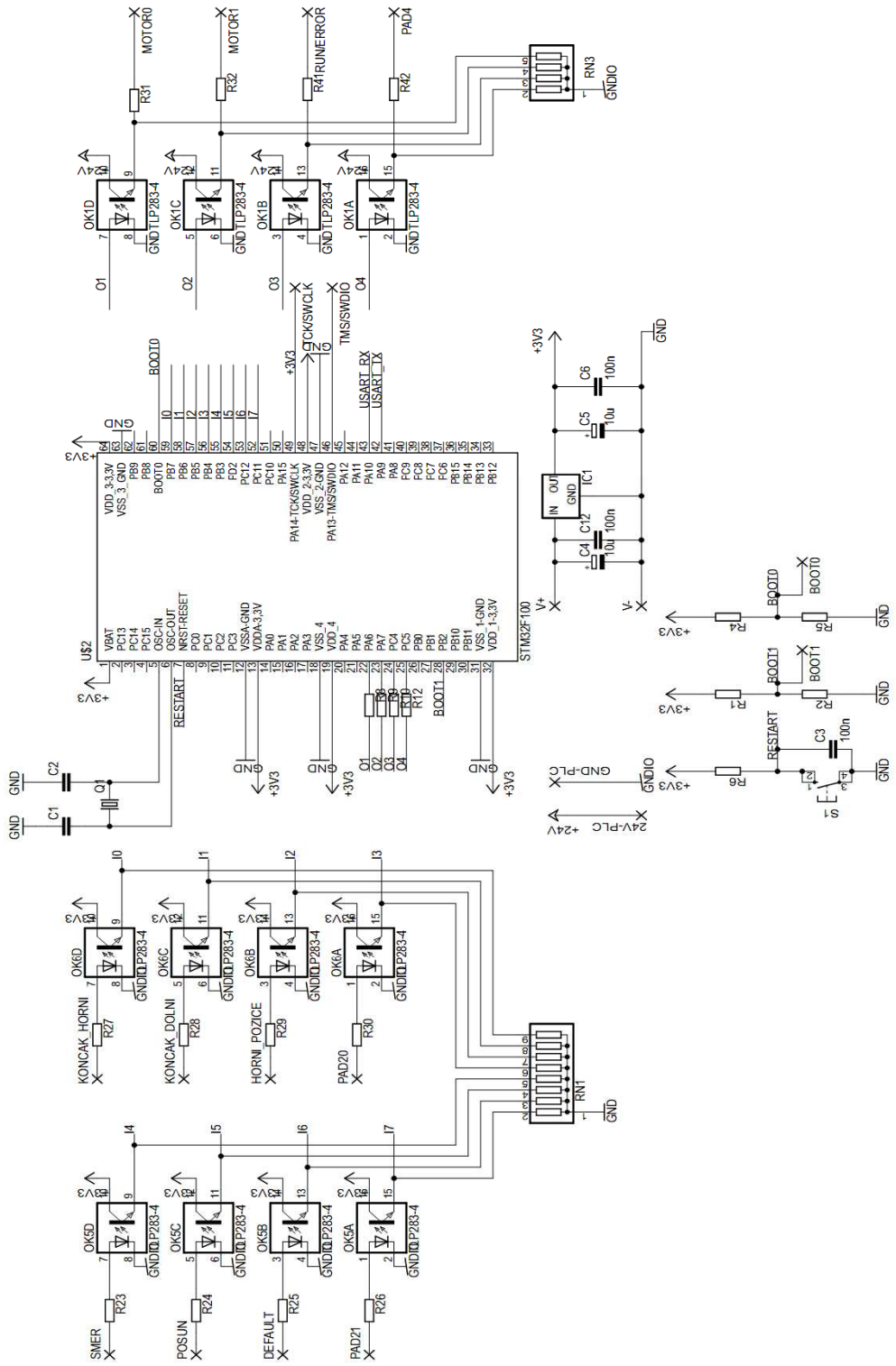
Příloha 5: Schéma převodníku



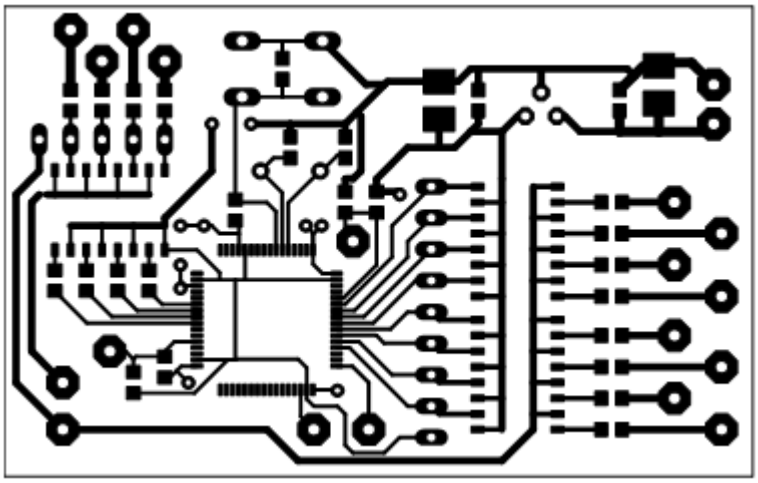
Příloha 6:Návrh plošného spoje pro převodník – spodní strana



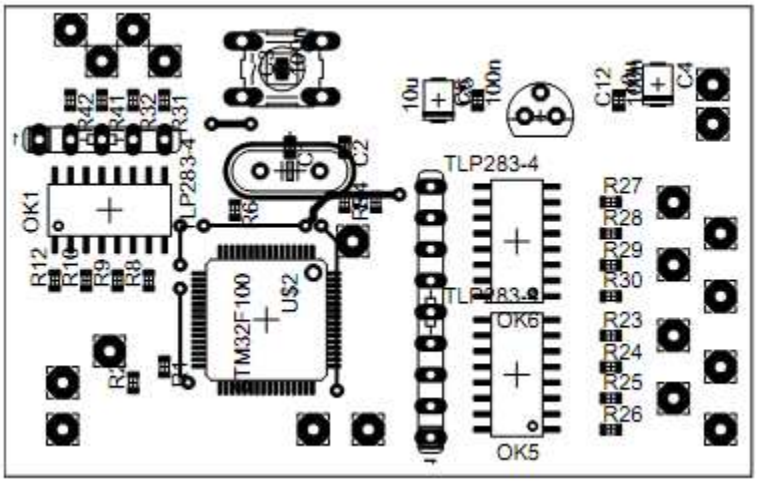
Příloha 7: Osazení plošného spoje pro převodník – pohled zdola



Příloha 8: Schéma zapojení obvodu pro komunikaci s PLC a řízení motoru



Příloha 9: Deska plošných spojů obvodu pro komunikaci s PLC a řízení motoru



Příloha 10: Osazovací plán obvodu pro komunikaci s PLC a řízení motoru