

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

SIMULAČNÍ MODEL Y EDUMOD PRO PLC V NI LABVIEW

SIMULATION MODELS FOR PLC IN NI LABVIEW

DIPLOMOVÁ PRÁCE

DIPLOMA THESIS

AUTOR PRÁCE

AUTHOR

STANISLAV JANČÍK

VEDOUCÍ PRÁCE

SUPERVISOR

ING. TOMÁŠ MARADA, PH.D.

BRNO 2008

ZADÁNÍ DIPLOMOVÉ PRÁCE

ZADÁNÍ DIPLOMOVÉ PRÁCE

ABSTRAKT:

Diplomová práce se zabývá propojením programovatelného automatu (PLC) s osobním počítačem (PC) za účelem řízení softwarových simulací v NI LabVIEW.

Diplomová práce je rozdělena na dvě hlavní části. Navrhnout a zhotovit komunikační jednotku, která bude mít na starost komunikaci mezi PC a PLC a naprogramovat simulační modely v softwaru NI LabVIEW.

Diplomová práce vznikla při řešení výzkumného záměru Inteligentní systémy v automatizaci podporovaného MŠMT ČR pod registračním číslem MSM 0021630529.

ABSTRACT:

This diploma thesis discusses the connection through the programmable automation (PLC) with a personal computer (PC) for the purpose of operating software simulation in NI LabVIEW.

Diploma thesis is divided in two main parts. Design and manufacture of the communication unit, which is in charge of communication between PC and PLC and further programming simulation models in software NI LabVIEW.

This thesis has been supported by the Czech Ministry of Education in the frame of MSM 0021630529 Research Intention Intelligent Systems in Automation.

KLÍČOVÁ SLOVA:

Programovatelný automat, Komunikační jednotka, NI LabVIEW, EDU-mod

KEYWORDS:

Programmable automation, Communication unit, NI LabVIEW, EDU-mod

Poděkování:

Tímto bych rád poděkoval svému vedoucímu práce Ing. Tomáši Maradovi, Ph.D. za výborné rady a pomoc při vývoji a výrobě elektroniky pro komunikaci mezi PLC a PC. Dále za to, že se mi snažil vždy pomoci a vyjít vstříc při pochopení funkce modelů EDU-mod. Dále bych rád poděkoval Ing. Zdeňku Němcovi, CSc za pomoc při navrhování elektroniky v komunikační jednotce. V neposlední řadě bych rád poděkoval rodičům za podporu jakou mi dali v době studia.

Obsah:

1.	ÚVOD	10
2.	KOMUNIKAČNÍ JEDNOTKA.....	13
2.1.	MultiSIM.....	13
2.1.1.	Komunikace mezi PC a PLC	13
2.1.2.	Komunikace mezi PLC a PC	15
2.1.3.	Naměřené hodnoty na vstupech do PLC.....	17
2.1.4.	Charakteristika převodu mezi PLC a PC	19
2.1.5.	Charakteristika převodu mezi PC a PLC	20
2.1.6.	Logické úrovně jednotlivých automatů.....	20
2.1.7.	TTL logika	21
2.2.	Eagle 4.16r2	21
2.2.1.	Editor schémat.....	21
2.2.2.	Editor plošného spoje.....	22
2.2.3.	Autorouter	22
2.2.4.	Popis schémat Hlavní desky	22
2.2.5.	Popis schématu Přední desky.....	23
2.2.6.	Vytvoření Hlavní desky jako DPS	24
2.3.	Výroba desky plošných spojů	24
2.3.1.	Tisk předlohy	24
2.3.2.	Expozice.....	25
2.3.3.	Vyvolání.....	25
2.3.4.	Leptání	25
2.3.5.	Smývání	26
2.3.6.	Zásady při zpracování	26
2.4.	Programování jednočipového mikropočítače ATmega128.....	27
2.4.1.	Úvod do problematiky	27
2.4.2.	Překladač C/C++ - WinAVR	27
2.4.3.	Integrované prostředí – AVR Studio	28
2.4.4.	Popis programu Komunikační jednotky	28
2.4.5.	Seznam součástek	29
2.4.6.	Komunikační jednotka - finále.....	29
3.	PROGRAMOVÁNÍ V LABVIEW.....	31
3.1.	Úvod do programování	31
3.2.	Úvodní program v LabVIEW	32
4.	SIMULACE EDU-MOD VYTVOŘENÉ V LabVIEW	35
4.1.	Sériová komunikace v LabVIEW	36
4.2.	Suport (Hydraulická posuvná jednotka)	40
4.3.	Křižovatka	46
4.4.	Mísící jednotka.....	47
4.5.	Pračka.....	55
5.	ZÁVĚR	63
	SEZNAM POUŽITÉ LITERATURY	65
	PŘÍLOHY	67

1. ÚVOD

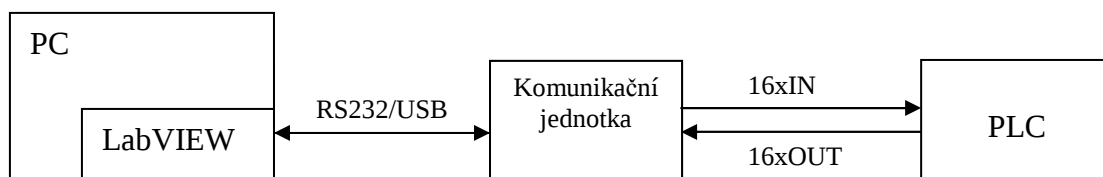
Automatizace je obor, který se neustále vyvíjí a to tak, že velmi progresivně. Dá se říci, že v poslední době se jejím hlavním prvkem stal programovatelný automat (PLC – „Programmable logic controller“). Jeho cena je sice stále relativně dosti vysoká, ale i přesto je mnohokrát levnější než systémy, které se vyrábí přímo na danou aplikaci. Výhoda programovatelného automatu je ta, že se dá použít k vyřešení široké škály problémů. Je tedy masově nasazován v průmyslu.

Programovatelný automat je relativně malý počítač, používaný pro automatizaci procesů v reálném čase. Pro PLC je charakteristické, že program se vykonává v tzv. cyklech. V moderním pojetí je výraz PLC nahrazován tzv. PAC (Programmable Automation Controller) i když označení PLC je celosvětově hojně rozšířené a udrží se i nadále. PLC automaty jsou odlišné od běžných počítačů nejen tím, že zpracovávají program cyklicky, ale i tím, že jejich periferie jsou přímo uzpůsobeny pro napojení na technologické procesy. Převážnou část těchto periférií tvoří digitální vstupy a výstupy, dále analogové vstupy a výstupy, komunikace, moduly pro řízení pohonů a jiné.

PLC se rozdělují do skupin. Hlavními skupinami jsou PLC kompaktní a PLC stavebnicové. Kompaktní PLC už obsahují všechny periferie (digitální vstupy/výstupy, analogové vstupy/výstupy) v jednom modulu. Jejich výhodou je ta, že jsou levnější. Nevýhodou, že se jen složitě a velmi omezeně dají rozšiřovat. Na rozdíl stavebnicové PLC se skládají z různých modulů. Hlavní modul je modul s CPU jednotkou a s konektory pro programování. K němu se dále přidává napájecí modul, komunikační modul, moduly se vstupy a výstupy...atd. Je omezen velikostí vany, ale vanu může být více (propojené pomocí komunikační jednotky). Nevýhodou je, že stavebnicové PLC jsou dražší, ale zase se lépe rozšiřují. Jsou výhodnější pro složitější řízení. [11]

Orientačně se cena malých kompaktních systémů pohybuje v cenách od 5.000 až 10.000 Kč, cena velkých a výkonově vyšších systémů v rozsáhlé konfiguraci může dosahovat částek 500.000 Kč i vyšších. Vzhledem k efektivnosti těchto systémů v průmyslu nejsou tyto položky nijak závratné. Z toho důvodu i ve školách je potřeba na tento trend navazovat. Proto vznikl nápad řídit pomocí PLC simulační modely v počítači.

V první řadě bylo nutné navrhnout elektroniku, která se bude starat o přenos dat mezi PLC a PC. PLC sice dokáže komunikovat s PC pomocí datového kabelu, ale jedná se o komunikaci k naprogramování PLC, nikoliv, že by PLC pomocí svých periférií dokázalo dávat příkazy pro PC. Jednotka, přes kterou se dá propojit PC s PLC existuje. Vyrábí ji například firma FESTO (typ EASYPOR D16), ale její cena se pohybuje v řádech desítek tisíců. Proto bylo potřeba vyrobit jednotku vlastní, která bude mít na starosti signály z automatu převádět na sériovou linku RS-232/USB, kterou data budou směřovány do PC. První verze této jednotky byla otestována a plně funguje s automaty od firem Siemens a Phoenix Contact. Má 16 digitálních vstupů a 16 digitálních výstupů a v průběhu navrhování obvodů se uvažovalo i o analogových vstupech a výstupech, od kterých se později upustilo vzhledem k tomu, že pro stávající úlohy nebudou potřebné.



Obr. 1.: Output PC-Input PLC při logické nule

Dalším úkolem bylo seznámit se s moduly EDU-mod. Tyto moduly se využívají pro studijní účely. Jsou to simulace běžných úloh, jako je automatická pračka, hydraulická posuvná jednotka obráběcího stroje, křížovátka nebo mísící jednotka. Pochopit tyto úlohy bylo nutné pro vytvoření jejich simulací v PC. Pro vytvoření simulací se jevilo vhodné vývojové prostředí LabView 8.2.

LabVIEW 8.2 je zkratkou pro *Laboratory Virtual Instrument Engineering Workbench* od firmy *National Instruments*. Využívá se hlavně pro programování komunikace mezi PC a různými periferiemi. LabVIEW používá k programování grafického jazyka G, který dle výrobce je téměř srovnatelný (co se týče debuggeru) s jazykem C. V současné době neexistuje česká lokalizace LabVIEW a dá se říci, že ani žádná učebnice či manuál v českém jazyce k tomuto programu. Určité usnadnění byly jiné diplomové práce od jiných studentů, kteří se LabVIEW zabývali, ale i přes tuto pomoc bylo nutné na spoustu věcí přijít metodou pokus, omyl či z anglického manuálu. Chválím nápovědu, která je součástí LabVIEW za její přehlednost a příklady, které v ní jsou zpracovány.

Jak nakonec komunikuje PC s PLC je znázorněno na obr.1. Data z výstupů PLC směřují ke komunikační jednotce, kde se přiřadí do rámce sériové linky a jsou odeslány do PC a data z PC jsou odeslány do komunikační jednotky, kde z rámce jsou nastaveny odpovídající vstupy do PLC.

2. KOMUNIKAČNÍ JEDNOTKA

Komunikační jednotka, jak bylo již v úvodu zmíněno, bude mít na starost komunikaci mezi PC a PLC. Srdcem jednotky je jednočipový mikropočítač (mikrořadič) ATmega128, který má na starost programově řešit komunikaci. Tedy ze strany automatu bude dostávat signály logických nul nebo jedniček a tyto signály bude převádět do kódu, který bude dále posílat sériovou linkou RS-232/USB do PC ve speciálním kódování. To stejné musí samozřejmě fungovat i reverzně.

Elektronika na straně programovatelného automatu, tím jsou myšleny vstupní i výstupní signály z PLC, je galvanicky oddělena. Je to pro ochranu jak komunikační jednotky proti zkratu, tak hlavně na ochranu PLC. PLC je sice samo o sobě už chráněno proti zkratům, ale u tak drahého zařízení je lepší mít ochran více.

Uvnitř komunikační jednotky je i její napájení pomocí dvou transformátorů. Vlastně jsou uvnitř dvě napěťové úrovně. Jedná má 5V a druhá 24V. Důvody proč tomu tak je budou uvedeny dále v textu.

Důvod proč se tato jednotka vyrábí je jednoznačně cena. Jak se dočtete dále, tak takhle vyrobená jednotka nestojí ani dva tisíce korun českých, na rozdíl od prodávaných jednotek od jiných výrobců, které jsou několikanásobně dražší.

2.1. MultiSIM

V první řadě je potřeba navrhnout obvody, které se budou starat o změnu napětí z 5V na 24V a naopak. Je to z důvodu, že PLC používají 24V logiku, kdežto mikrořadič v komunikační jednotce používá TTL logiku, tedy 5V logiku. Pro tento návrh se kromě výpočtů dá použít i program MultiSIM.

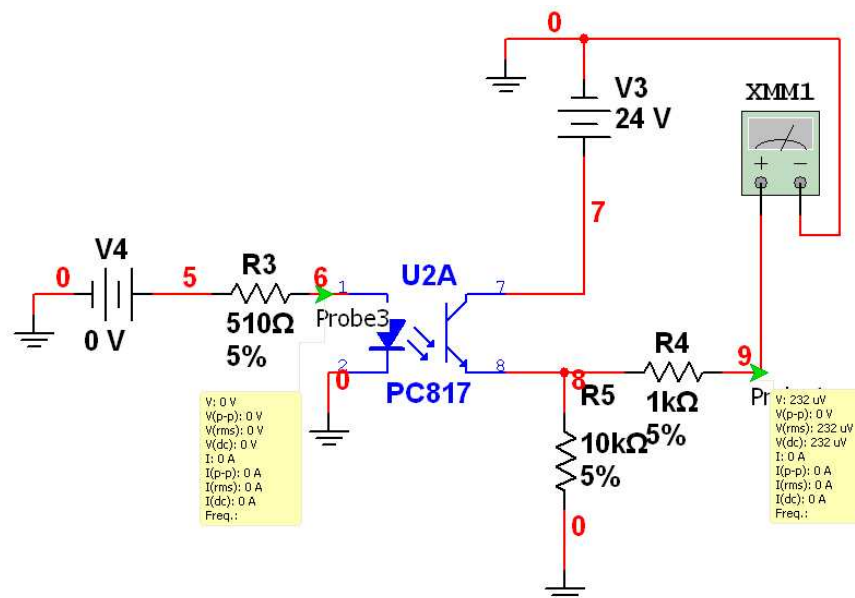
MultiSIM je součástí balíčku Circuit Design Suite 10.0 od společnosti National Instruments. Na internetu je ke stažení jako shareware s 30-ti denní platností používání. Po vypršení je nutné si koupit licenci. Dříve se program nazýval Electronics Workbench a vyráběla jej stejnojmenná firma. Slouží k návrhu elektronických schémat a následně k jejich simulacím. Je nutné si i stáhnout potřebné knihovny s elektronickými součástkami, protože v samotné instalaci programu jste jimi dosti omezeni [4].

Je více možností jak sestavit obvody, aby plnily žádaný účel. Například se dají řešit různým sestavením NPN tranzistorů nebo darlingtonovým tranzistorem. Tyto možnosti by plnily účel pro převod mezi různými úrovněmi napětí a další jejich výhodou je i cena. Ovšem už by nesplňovaly požadavek chránit zařízení proti zkratům a přepětí. Z tohoto důvodu je mnohem vhodnější použít optočleny. Výhoda optočlenů oproti tranzistorům je ta, že nám i jednotlivé zařízení od sebe galvanicky oddělí. Galvanické oddělení znamená, že je oddělené země. Tudíž při zkratu nebo naopak přepětí v jedné části obvodu se nedostane přepětí a zkrat do druhé části obvodu. Vzhledem k tomu, že tohle galvanické oddělení dělají právě optočleny, tak jsou také nejnáchylnější ke zničení. Proto je potřeba je moci co nejsnadněji vyměnit, a proto je lepší, když nebudou k desce plošných spojů připájeny, ale zasazeny do precizních patič.

2.1.1. Komunikace mezi PC a PLC

Na obr.2. je zapojení pro komunikaci mezi mikrořadičem ATmega128 a PLC, pokud mikrořadič má na výstupu logickou nulu. Při logické nule je na výstupu mikrořadiče napětí

0V. Tím pádem se nerozsvítí LED dioda v optočlenu U2A a neotevře se tranzistor. Z toho důvodu vstup na PLC je přizemněn nulovým napětím přes R4 a R5. Zdroj napětí V4 simuluje mikrořadič ATmega128.



Obr.2.: Output PC-Input PLC při logické nule

V případě, že se na výstupu mikrořadiče objeví logická jednička (na výstupu je napětí 5V), tak se pro změnu rozsvítí LED dioda v optočlenu a otevře se tranzistor. Tudíž se na výstup dostane 24V přes kolektor na emitor a přes R4. Odpor R3 byl vypočítán dle vzorce (1):

$$R_3 = \frac{U_{V4}}{I_{LED}} = \frac{5}{10 \cdot 10^{-3}} = 500\Omega \quad (1)$$

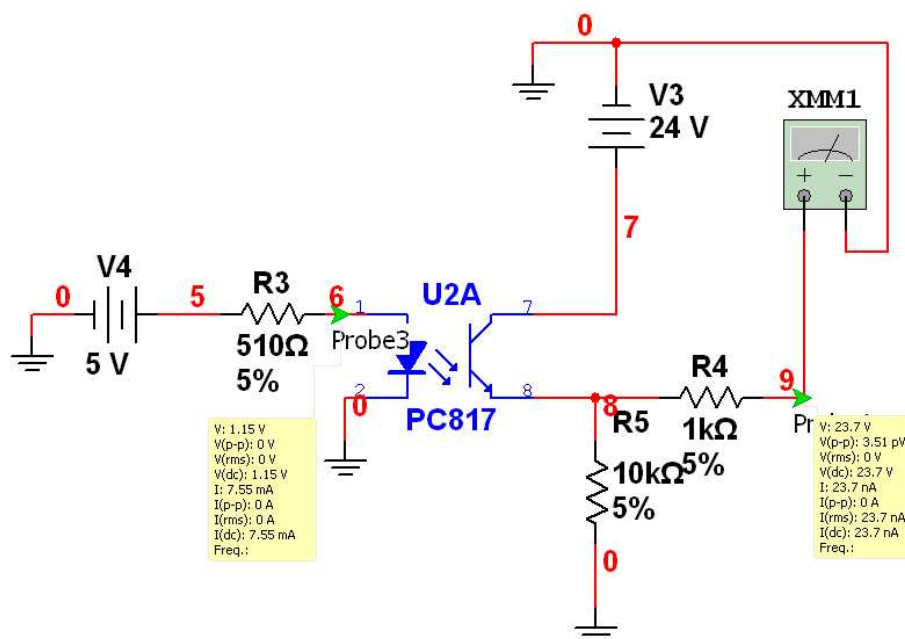
Použit byl tedy odpor SMD 510R 0805. Ovšem ze simulace je patrné, že do optočlenu neteče proud 10mA dle výpočtu, ale 7,55mA jak je vidět na obr.3. Je to ten samý obvod, jen s tím rozdílem, že je sepnutá logická jednička. Položme si tedy otázku proč? Na LED diodě u optočlenu je totiž úbytek napětí přibližně $U_u = 0,7V$, z toho důvodu je vzorec (1) pozměněn na:

$$R_3 = \frac{U_{V4} - U_u}{I_{LED}} = \frac{5 - 0.7}{10 \cdot 10^{-3}} = 430\Omega \quad (2)$$

Ovšem ve funkci obvodu to nějak nevádí, proto je ponechán odpor na hodnotě 510Ω. Tak se dostatečně otevře tranzistor v optočlenu a na výstupu je docíleno hodnoty viz. Tab.1. Hodnoty jsou nejen závislé na odporu R3, ale hlavně na odporech za optočlenem R4 a R5, jak bude dokázáno později. Z tabulky je patrné, že na výstupech z komunikační jednotky je při logické nule i na výstupu nula. Tedy měření bez i se zatížením vykáže vždy stejnou hodnotu. Při logické jedničce jsou rozdíly mezi napětími na různých automatech. Důvod bude dokázán tak též níže.

	Napětí na vstupu bez zatížení (V)	LOGO (V)	Simatic (V)	Phoenix Contact (V)
log 0	0	0	0	0
log 1	23,85	19,96	18,5	19,42

Tab. 1.: Naměřené hodnoty na vstupech PLC (výstupech komunikační jednotky)



Obr.3.: Output PC-Input PLC při logické jedničce

2.1.2. Komunikace mezi PLC a PC

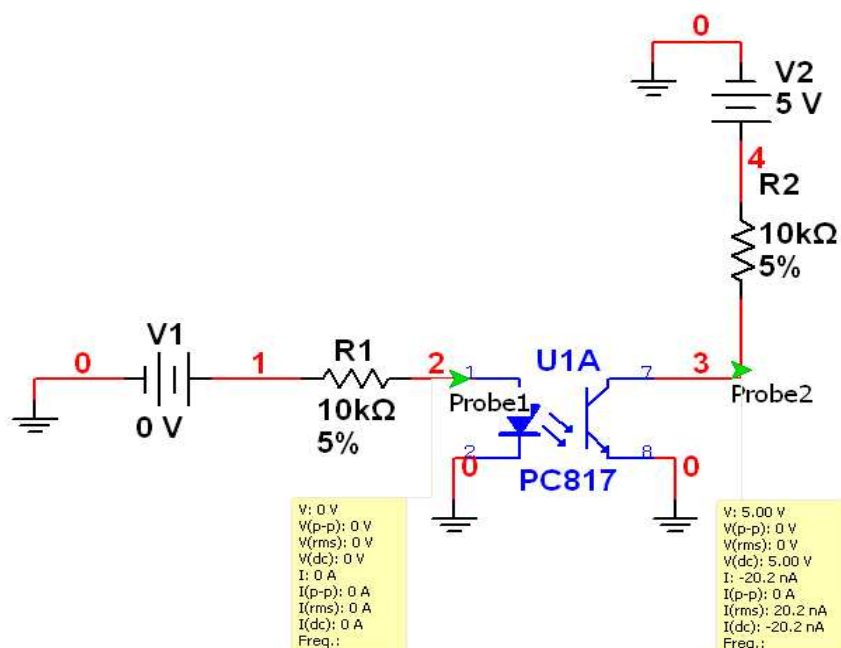
Na obr.4. je zobrazen obvod pro obrácenou komunikaci. Tedy z výstupu PLC do vstupu mikrořadiče. Na obrázku je patrné, že obvod obrací logiku. Tedy pokud je na výstupu PLC 0V, potom je na vstupu ATmega128 5V. Největší problém v tomto obvodu bylo vybrat rezistory před optočlenem. Tedy kromě výpočtu proudu pro rozsvícení LED diody (3):

$$R_1 = \frac{U_{V1}}{I_{LED}} = \frac{24}{2,5 \cdot 10^{-3}} = 9600\Omega \quad (3)$$

Při výpočtu se ztrátou na LED diodě bude odečtena od U_{V1} hodnota $U_u = 0,7V$. Výsledek se potom změní na $R_1 = 9320\Omega$. Zvolí se tedy odpory $10k\Omega$. Tady je výhodnější volit co největší odpor z důvodu velkého příkonu P , který zde vzniká. Je to patrné z výpočtu (4):

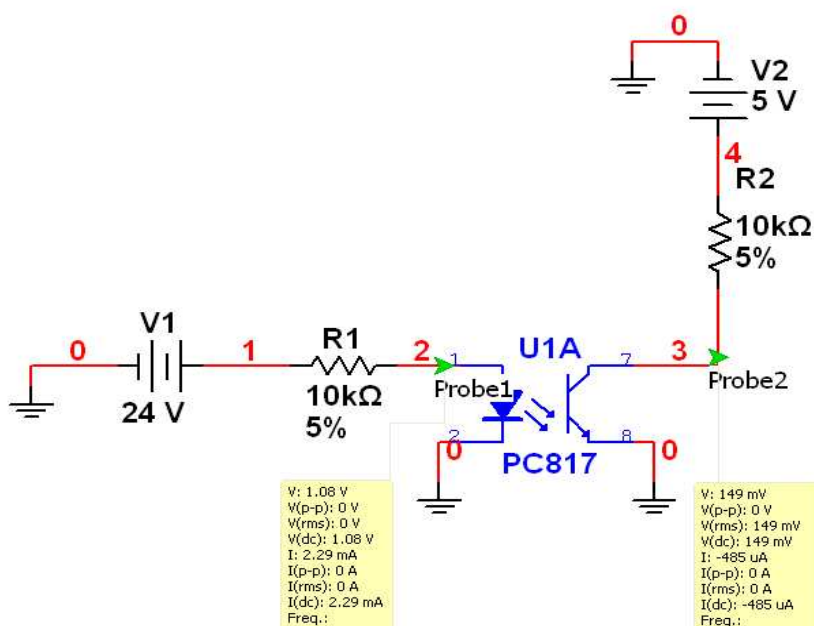
$$P = \frac{U_{V1}^2}{R} = \frac{24^2}{10 \cdot 10^3} = 0.0576W \quad (4)$$

Při tak velikém příkonu už se nedají použít běžně dostupné SMD rezistory 0805/1206, protože už by hrozilo spálení vlivem vysoké teploty.



Obr. 4.: Input PC-Output PLC při logické nule

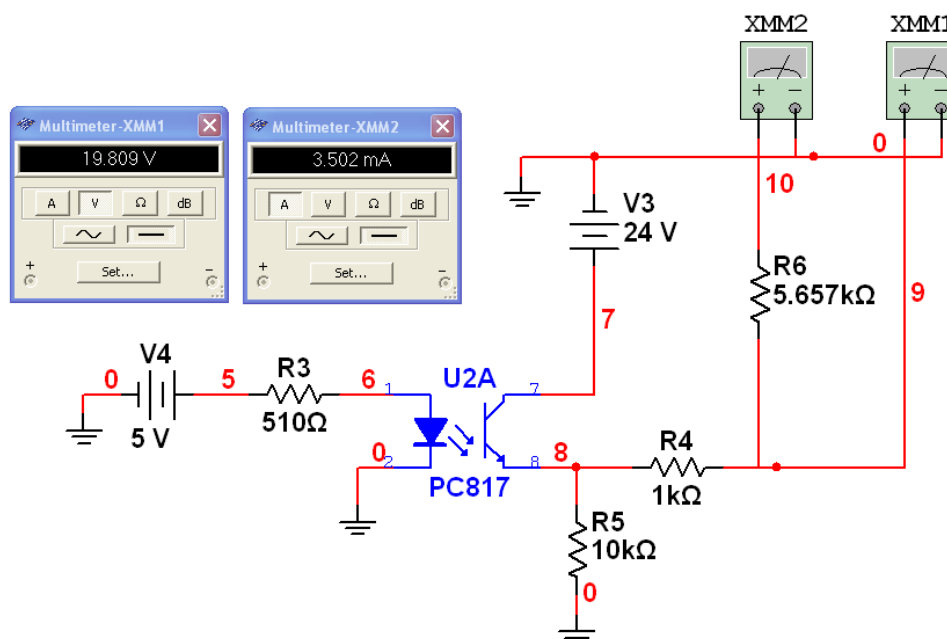
Na obr.5. je na výstupu PLC logická jednička. Tedy napětí 24V. Proud za odporem R_1 odpovídá dle simulace $I_{LED} = 2,29\text{mA}$, což postačuje na dostatečně velké otevření tranzistoru. Jakmile se tranzistor otevře, tak proud na kolektoru proteče emitorem do země. Z toho důvodu na výstupu (větev 3 na obr.5.) je napětí nulové.



Obr. 5.: Input PC-Output PLC při logické jedničce

2.1.3. Naměřené hodnoty na vstupech do PLC

Jak už bylo výše zmíněno, tak nyní se více zaměříme na Tab.1., kde bylo naměřeno napětí na prázdko a napětí při připojení k PLC Simatic S7, LOGO! a Phoenix Contact ILC150. Při měření naprázdno bylo na svorkách výstupní napětí $U_{\text{vyst}} = 23,85\text{V}$, ale pokud jsou svorky připojeny k jednotlivým automatům, tak napětí se sníží a vždy na jinou hodnotu, podle druhu automatu. To je způsobeno různým úbytkem napětí na svorkách.



Obr. 6.: Simulace úbytku napětí na vstupech do PLC (Dělič napětí)

Nejdříve se napětí z 24V vlivem ztrát mezi emitorem a kolektorem sníží na 23,7V podle obr.3. Jak je vidět na Obr.6., tak se dá říci, že automat svým chováním pomocí odporu R4, R6 připomíná dělič napětí [5]. Automat zde je simulován odporem R6. Pokud je měřen bez odporu (tedy na prázdko), tak výstupní napětí je 23,7V dle MultiSIMU. Při měření multimetrem byla zjištěna hodnota 23,85V (viz. Tab.1).

Výpočet odporu R6 vypočítán ze vzorce pro odporový dělič (5):

$$U_{\text{VYST}} = U_{\text{VST}} \frac{R_2}{R_1 + R_2} \quad (5)$$

Tedy ze vzorce je osamostatněno R_2 a je získána rovnice (6):

$$R_2 = U_{\text{VYST}} \frac{R_1}{U_{\text{VST}} - U_{\text{VYST}}} \quad (6)$$

V tomto případě je odpor R_1 roven odporu R4 viz obr.6. Za odpor R_2 se dosadí odpor R6, který jak je již výše uvedeno, simuluje úbytek napětí na vstupech do automatu. U_{VST} je napětí, které je naměřeno v bodě 8 dle obr.6. V našem případě se tedy rovná $U_{\text{VST}} = 23,3\text{V}$. Budou porovnávány hodnoty s MultiSIMU s naměřenými hodnotami a tím dokázáno jak velice přesný tento program je. U_{VYST} je hodnota, která byla naměřena na

svorkách automatu při logické jedničce (viz. Tab.1.). Z toho důvodu zde budou tři výpočty pro automaty LOGO!, Simatic S7 a Phoenix Contact.

Dosazeny, do výše zmíněného vzorce (6), hodnoty pro automat LOGO!:

$$R_2 = R_6$$

$$R_2 = 19,96 \cdot \frac{1 \cdot 10^3}{23,3 - 19,96} = 5,976 k\Omega \quad (7)$$

Pokud je hodnota R_2 dosazena v MutliSIMU za rezistor R_6 a spustí se simulace, tak na multimetru XMM1 (viz. obr.6.) bude hodnota napětí $U_{\text{multisim}} = 19,971V$. Když hodnota U_{multisim} je porovnána s hodnotou $U_{\text{VYST}} = 19,96V$ pro LOGO! (viz Tab.1.) tak je zjištěno, že program se od naměřené hodnoty liší v řádech setin voltu. V tomto případě je přesnost mezi hodnotami naměřenými reálným multimetrem a hodnotami naměřenými v MultiSIMU více než dostatečná. Na multimetru XMM2 (viz. obr.6.) bude odečten proud, který odebírání LOGO! a tj. $I_{\text{LOGO}} = 3,342 \text{ mA}$.

Dále jsou dosazeny do vzorce (6) hodnoty pro automat Simatic:

$$R_2 = R_6$$

$$R_2 = 18,5 \cdot \frac{1 \cdot 10^3}{23,3 - 18,5} = 3,854 k\Omega \quad (8)$$

Odběr u automatu Simatic je největší ze všech testovaných. Dle simulace v MutliSIMU je hodnota napětí $U_{\text{multisim}} = 18,384V$, což je rozdíl jedné desetiny voltu mezi naměřenou hodnotou a spočítanou hodnotou programem. Ani tady ale není rozdíl nikterak veliký. Hodnota proudu $I_{\text{SIMATIC}} = 4,77mA$.

Jako poslední jsou dosazeny hodnoty pro automat Phoenix Contact do rovnice (6):

$$R_2 = R_6$$

$$R_2 = 19,42 \cdot \frac{1 \cdot 10^3}{23,3 - 19,42} = 5,005 k\Omega \quad (9)$$

Phoenix Contact při dosazení hodnoty R_6 do simulace v MultiSIMU (viz. Obr.6.) vykazuje hodnotu napětí $U_{\text{multisim}} = 19,382V$ a proud $I_{\text{PHOENIX}} = 3,872mA$.

Veliká výhoda u automatu Phoenix Contact je ta, že LED diody, které indikují vstupy do automatu se rozsvítí jen a jen v tom případě, že automat detekuje logickou jedničku. Tohle pravidlo neplatí u Simaticu od firmy Siemens, který LED diody rozsvěcuje podle napětí na vstupu. Takže například pokud je 12V na vstupu LED dioda už slabě svítí, ale automat ještě nedetekoval logickou jedničku.

2.1.4. Charakteristika převodu mezi PLC a PC

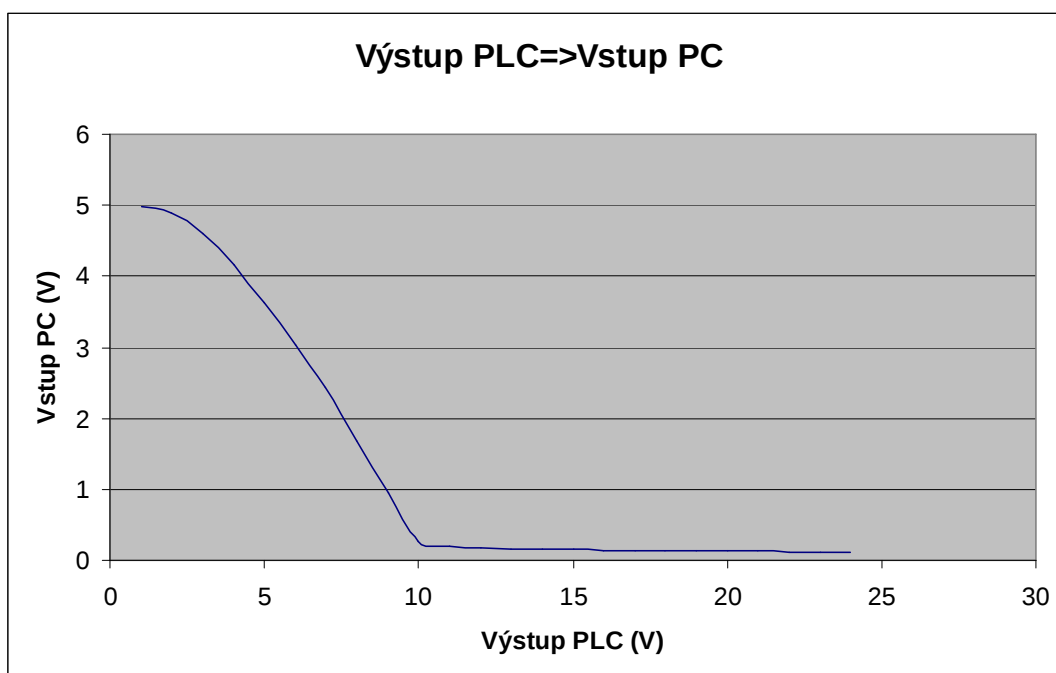
Na svorky, které se připojují k výstupům PLC byl připojen zdroj napětí a po 1V bylo zvyšováno napětí. Na vstupech do mikrořadiče byly pomocí multimetru (METEX M-4650CR) měřeny hodnoty napětí (viz Tab.2.).

Výstup z PLC (V)	1	2	3	4	5	6	7	8	9	10	11
Vstup do ATmega (V)	4,99	4,89	4,6	4,16	3,63	3,04	2,43	1,68	0,96	0,27	0,19
Výstup z PLC (V)	12	13	14	15	16	17	18	19	20	21	22
Vstup do ATmega (V)	0,17	0,16	0,15	0,14	0,14	0,13	0,13	0,12	0,12	0,12	0,12
Výstup z PLC (V)	23	24									
Vstup do ATmega (V)	0,118	0,115									

Tab. 2.: Tabulka pro vytvoření charakteristiky mezi PLC a PC

Z těchto hodnot vzniknul graf (Graf 1.) charakteristiky mezi výstupem PLC na vstupu PC. Vzhledem k tomu, že tento obvod otáčí logiku, tak se dá z grafu určit, že pokud je na výstupu PLC 0 až 7V, tak toto napětí ATmega (jednočipový mikropočítač) zaznamená jako logickou 0. Pokud bude na výstupu PLC 10 až 24V, potom se bude jednat o logickou jedničku. Proč tomu tak je bude vysvětleno v kapitole 2.1.7 TTL logika.

Získaná charakteristika je plně dostatečná pro úlohu typu komunikační jednotky. Pokud by dostatečná nebyla, tak by se musel použít Schmittův klopný obvod. Ten se používá pokud je potřebné získat z analogového signálu, signál logický. Má totiž tu vlastnost, že se dokáže razantně překlomit ze stavu 0 do stavu 1.



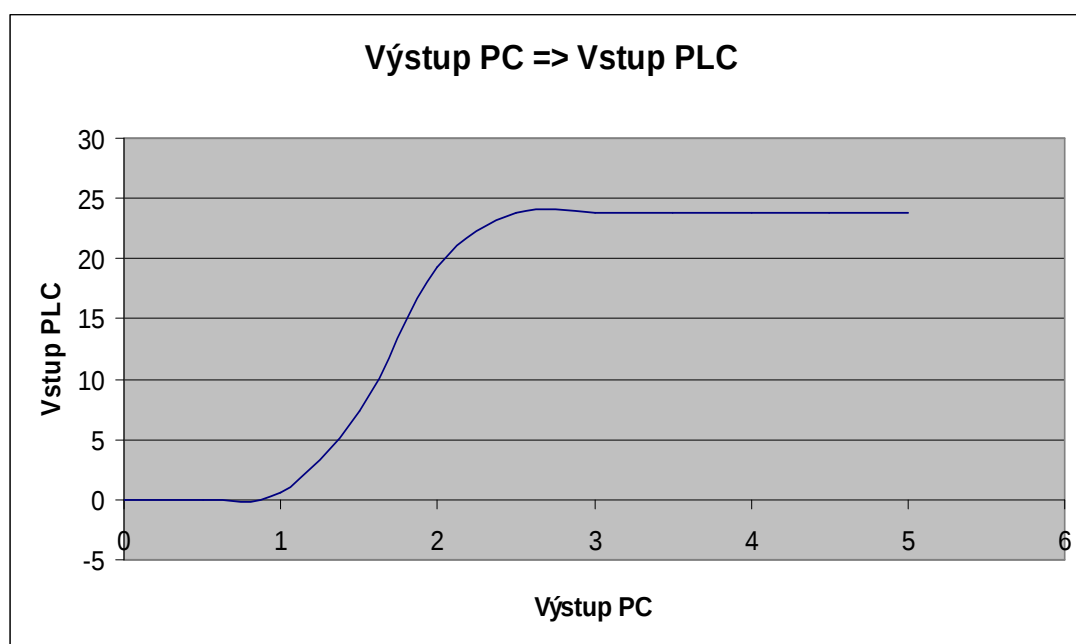
Graf 1.: Charakteristika mezi výstupem PLC a vstupem PC

2.1.5. Charakteristika převodu mezi PC a PLC

Výstup z ATmega (V)	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5
Vstup do PLC (V)	0	0	0,6	7,3	19,35	23,78	23,87	23,82	23,83	23,84	23,85

Tab. 3.: Tabulka pro vytvoření charakteristiky mezi PC a PLC

Tab.3. vznikne tak, že se na výstup z mikrořadiče připojí zdroj napětí. Samozřejmě se musí mikrořadič vyřadit z činnosti, aby na něj neměl zdroj napětí vliv. V tomto případě byl odpájen jeden prokov na desce plošných spojů. A je měřeno napětí na výstupu z komunikační jednotky, tedy na vstupech do PLC.



Graf 2.: Charakteristika mezi výstupem PC a vstupem PLC

Z grafu (viz. Graf 2.) je názorně vidět, že do napětí 1V se drží charakteristika na 0V, až překročíme 1V, tak začne prudce stoupat a při 2,5V už je obvod plně otevřený. Vzhledem k tomu, že při TTL logice se nachází logická nula mezi napětími 0 až 0,8V a logická jednička mezi napětími 2 až 5V. Z toho vyplývá, že je charakteristika vhodná a použitelná. Nehledě na to, že ATmega128 má na výstupu 0V při logické nule a 4,99V při logické jedničce. Ale opět pokud by charakteristika byla nedostatečná, tak se dá použít Schmittův klopný obvod jako v předešlém případě. Ovšem opět není potřeba pro tuhle aplikaci.

2.1.6. Logické úrovně jednotlivých automatů

Velice důležité pro elektronické obvody je znalost úrovně signálů v PLC. Což znamená, při jakém napětí programovatelný automat rozezná logickou jedničku a při kterém naopak logickou nulu. Všechny tyto úrovně bývají obvykle obsahem technické dokumentace k PLC.

	Logická 0	Logická 1
LOGO!	< 5V	> 8V
Siemens S7-300	-5V-5V	13V-30V
Phoenix Contact 150 (350)	0V-5V	15V-24V

Tab. 4.: Logické úrovně signálů

Z tabulky Tab.4. je patrné, jak jednotlivé automaty rozlišují logické signály. Výše v textu je uvedeno, že nejnižší napětí ze všech automatů má Simatic S7. Jeho úroveň odpovídá 18,5V. Z tabulky je patrné, že toto napětí je dostatečné k indikování logické jedničky. To platí i pro ostatní automaty.

2.1.7. TTL logika

TTL (tranzistor-tranzistor-logic) je standardem používaným u integrovaných obvodů. Vychází s technologie bipolárních tranzistorů a jejich napájecí napětí je přibližně 5V. Napětí 0V až 0,8V se označuje jako logická 0. Napětí 2V až 5V jako logická jednička. Pokud je dokázáno aby obvod fungoval v rozmezí 0V až 0,3V jako logická 0. 2,7V až 5V jako logická jednička, tak pak bude kompatibilní s logikou TTL. Trendem je nyní snižovat napájecí napětí, takže už existuje logika 3,3V, 2,5V, 1,8V a 1,2V.[6] V případě komunikační jednotky se ale stále jedná o logiku s napájecím napětím 5V.

2.2. Eagle 4.16r2

Program byl použit pro návrh desky plošných spojů (DPS). Program Eagle vyrábí firma CADsoft. Je to program pro návrh plošných spojů. Návrhový systém EAGLE se skládá ze tří hlavních modulů:

- Editor schémat
- Editor plošného spoje
- Autorouter

Eagle je v Evropě velmi rozšířený, hlavně díky tomu, že existuje jeho volně šiřitelná verze, která je zdarma. Na domovských stránkách programu je k dispozici ke stažení jak program, tak jeho aktualizace, popřípadě knihovny.

2.2.1. Editor schémat

Na začátku práce s editorem je potřeba si dát pozor na pár záležitostí. V první řadě se jedná o zvolený rastr. Rastr by měl být vždy ponecháván v tzv. „defaultním“ nastavení. V tomto případě se rastr rovná 2,54mm. Pokud by rastr byl změněn na jiný, mohl by potom nastat problém při výrobě desky. U součástek by byla jiná rozteč nožiček, než rozteč u desky plošných spojů pro připájení nožiček a z toho důvodu by se prostě na desku nedaly součástky připájet. Deska by byla k ničemu.

Další problém může nastat při propojování jednotlivých součástí. V Eagle se nástroj pro propojování součástek nazývá *NET*. Tady se může stát vlivem nepozornosti, že se připojí součástka na jinou *NET* než na kterou by měla. Stává se to hlavně při výrobě složitějších schémat.

Funkce *Erc* slouží k překontrolování jestli nenastala nějaká chyba při vytváření schémat. Jedná se hlavně o nepřipojené cesty, nebo nesmyslně připojené cesty.

Pokud je vše hotovo, tak ze schématického editoru může být vytvořena deska plošných spojů stisknutím *Switch to board*.

2.2.2. Editor plošného spoje

Editor plošného spoje slouží již k návrhu desky plošného spoje (DPS). V úvodu práce by měl být nastaven rámeček. K tomu slouží funkce *WIRE*. Tenhle rámeček ohraničuje DPS. Pokud se používá Eagle volně dostupný na internetu jako freeware, tak je velikost tohoto rámečku omezena.

Další na řadě je propojování součástí. Eagle podporuje více vrstev (v případě komunikační jednotky jde o vrstvy spodní a horní). Při přechodu mezi vrstvami si Eagle sám vygeneruje prokov. Pro vytváření cest na DPS se používá funkce *ROUTE*.

Zde přibyla další možnost kontroly správnosti a tou je funkce *Drc*. Ve funkci může být nastaveno jak blízko mohou být cesty u sebe, jak blízko u patic, průměr prokovů, velikost průměru díry pro součástky a spousta dalších jiných specifikací.

2.2.3. Autorouter

Autorouter slouží k vygenerování všech cest na DPS automaticky. Cesty nevytváří projektant, ale jsou vytvořeny pomocí tohoto programu. Opět zde mohou být nastaveny všechny potřebné specifikace. Například šířka cest, vzdálenost cest a podobně. Je to docela hojně využívaná funkce, ale z vlastních zkušeností můžu říci, že pro složitější DPS není moc dobře použitelná.[12]

2.2.4. Popis schémat Hlavní desky

Schémat uvedená v příloze na konci práce patří k návrhu Hlavní desky. Hlavní deska se v komunikační jednotce stará o veškerou komunikaci mezi PC a PLC a je na ní umístěna i část napájení. Přesněji řečeno, obsahuje oba transformátory.

Co se týká prvních dvou schémat (Příloha 1. a Příloha 2.), tak jsou popsány v kapitolách 2.1.1 a 2.1.2. Na posledním schématu (Příloha 3.) jsou tři integrované obvody, již zmíněné transformátory a konektory (USB, COM-9, 2x CENTRONICS 24V pro komunikaci mezi PLC a Komunikační jednotkou, 2 konektory pro napájení 5V a 24V, konektor pro propojení Hlavní desky s Přední deskou).

Integrovaný obvod FT232RL slouží k převodu klasického USB na sériovou linku RS-232 pomocí TTL úrovní. Uživatel tedy nemusí znát detailně komunikaci po USB a programovat pro něj obslužné algoritmy. Zapojení FT232RL v obvodu lze vypožorovat z technické dokumentace, která se k obvodu dá volně stáhnout z internetu. Dále jsou k modulu potřeba ovladače, které jsou volně dostupné. Jsou dva možné typy:

- Direct driver
- VCP driver

Integrovaný obvod MAX232ECWE slouží jako převodník RS232 na TTL úrovni. Výhoda tohoto obvodu je ta, že potřebuje pouze jeden zdroj napětí +5V. Integrovaný obvod obsahuje 2 převodníky mezi TTL a RS-232 a 2 převodníky mezi RS-232 a TTL. Potřebné zapojení je uvedeno v technické dokumentaci k obvodu.

Posledním integrovaným členem na DPS je jednočipový mikropočítač ATmega128, který může být nahrazen mikrořadičem ATmega64. Rozdíl mezi těmito mikrořadiči je pouze velikost paměti. Co se týká zapojení napájení a programování mikrořadiče, tak schémata lze nalézt na webových stránkách firmy Atmel. U napájení jde o přivedení napětí +5V a země a u programování jde o připojení na konektor MOSI (master out slave in), MISO (master in slave out), SCK (SPI serial clock), RESET a GND (zem). Dále se musí vybrat 32 portů vstupně/výstupních jednotek, z nichž bude 16 portů použito jako vstupy a 16 jako výstupy. Pro vstupy jsou vybrány částečně porty B, D, E a pro výstupy celé porty A, C. Port D2 a D3 je použit pro sériovou komunikaci. ATmega128 má dohromady dva sériové kanály. Zbytek vývodů mikrořadiče zůstane neobsazeno. V úvodu práce byla poznámka, že se dá komunikační jednotka osadit ještě o analogové vstupy a výstupy. Pro ně je na mikrořadiči volný PORTF. Vývody XTAL1 a XTAL2 slouží k připojení externího krystalu. V tomto případě jde o krystal $Q_1 = 11.059$ MHz. Pomocí tohoto kmitočtu se lépe obsluhuje sériová komunikace RS-232. Další podrobnosti o mikrořadiči Atmel budou v kapitole 2.4.

2.2.5. Popis schématu Přední desky

Přední deska (Příloha 4.) slouží k signalizaci vstupů a výstupů, dále k signalizaci přenosu dat, k signalizaci napájení. Celkem tedy na předním panelu je 36 LED diod. Dále se přední deska stará o část napájení. Jsou na ní umístěny dva obvody. Jeden pro napájení 5V a druhý pro napájení 24V, které se starají pomocí usměrňovačů na převod ze střídavého proudu na stejnosměrný a získání stabilního napětí 5V a 24V.

Výpočet velikosti filtračních kapacitorů:

$$f = 50\text{Hz} \quad (10)$$

$$T = \frac{1}{f} = 0,02\text{s}$$

$$I_{\max 5} = 16 \cdot I_5 = 16,7,5 \cdot 10^{-3} = 0,12\text{A} \Rightarrow I_{\max 5} = 0,2\text{A}$$

$$I_{\max 24} = 16 \cdot I_{24} = 16,2,5 \cdot 10^{-3} = 0,04\text{A} \Rightarrow I_{\max 24} = 0,1\text{A}$$

$$U_5 = U_{\max 5} - U_{\min 5} = 9 - 5 = 4\text{V}$$

$$U_{24} = U_{\max 24} - U_{\min 24} = 36 - 24 = 12\text{V}$$

$$C_5 = \frac{I_{\max 5} \cdot 0,5 \cdot T}{U_5} = \frac{0,2 \cdot 0,5 \cdot 0,02}{4} = 500\mu\text{F}$$

$$C_{24} = \frac{I_{\max 24} \cdot 0,5 \cdot T}{U_{24}} = \frac{0,1 \cdot 0,5 \cdot 0,02}{12} = 83,3\mu\text{F}$$

Správně by v obvodu měly být filtrační kapacitory o těchto velikostech. V obvodu jsou zvoleny jiné, ale jde spíše o součástky, které nemají zásadní vliv na funkci Komunikační jednotky.

2.2.6. Vytvoření Hlavní desky jako DPS

Jakmile jsou schémata hotová, tak může být zhotovena DPS jak bylo popsáno v kapitole 2.2.1. U Hlavní desky jsou cesty natolik složité, že je třeba použít oboustrannou desku. Zrovna tato deska je natolik složitá, že zde právě nelze použít autorouter a všechny cesty je potřeba, aby byly navrženy ručně. Je třeba určit šířku cest. Nejširší cesty jsou napájeny napětím 230V přímo z elektrické sítě a jsou přivedeny na primární vynutí u transformátorů. Další širší cesty patří k hlavnímu napájení 24V a 5V a taky k oběma zeměním GND-5 a GND-24. A nejtenčí, v tomto případě 16 mil, jsou ostatní datové cesty.

Další věc, na kterou musí být dbáno, je dostatečně velká vzdálenost od míst, kde budou prokovy. Je to z důvodu, že v místě bude pájeno a musí tam proto být dostatečné množství mědi. Proto plošky u míst, kde musí být pájena nějaká součástka nebo prokov, je lepší udělat co možná největší a díry co možná nejmenší. Pokud by se deska nechala vyrábět profesionálně, tak to nutně rozhodně není, ale v případě výroby ruční je to lepší řešení pro snadnější usazování součástek.

2.3. Výroba desky plošných spojů

Existuje spousta metod na výrobu desek plošných spojů. Výroba foto cestou je poměrně jednoduchá a vyrobená deska vypadá velice slušně. Nejprve je potřeba mít patřičné vybavení. Většinu věcí může být koupena v obchodě s elektrosoučástkami:

- Leptací roztok FeCl_3 0,5 l
- Vývojka 1,5 % NaOH 0,5 l
- Aceton nebo jiné ředidlo na barvy
- Fix na plošné spoje
- Fotosenzitivní plošný spoj jednostranný 90 x 270
- Fotosenzitivní plošný spoj oboustranný 220 x 240
- Vrtačka na plošné spoje (např. Drill 3000)
- Plochá miska
- UV lampa (horské slunce)
- Smirkový papír
- Pauzovací papír
- Izolepa

Uvedené chemikálie vystačí na plochu až 15 dm^3 desek plošných spojů, což je více než dostatečné.

2.3.1. Tisk předlohy

Začíná se tím, že předloha bude vytištěna na pauzovací papír (nejlépe větší tloušťky než je klasický pauzovací papír). DPS, kterou byla vytvořena v Eaglu se vytiskne na laserové tiskárně. Při tisku inkoustovou tiskárnou se papír může mírně zdeformovat, což vadí a také tisk nedosahuje patřičného kontrastu a pak UV lampa může prosvítit i to co by prosvítit neměla. Před tiskem je potřeba obrázek zrcadlově otočit a před osvětlením je položen obrazcem na desku. Při tvorbě oboustranné desky strana TOP není zrcadlena. Vždy je třeba, aby výsledný plošný spoj nebyl zrcadlově otočen, protože pak by byl

většinou zničen. Předloha musí být vytištěna černou barvou. Pokud předloha je v pořádku vytištěna, tak se nad miskou s acetonem položí, tonerem směrem dolů, vytištěná předloha. Předloha nad acetonem musí být ponechána 15 minut, aby aceton mohl působit. Ředidlo naleptá toner a tím zvýší jeho kontrast. Při pohledu proti světlu nesmí předloha prosvítat. Místo předlohy je možné vytvořit film, který se také často využívá. Film je zobrazen v příloze (Příloha 5., Příloha 6., Příloha 7.)

2.3.2. Expozice

Pokud je film připraven, tak se může začít exponovat. Pustíme si UV lampu a necháme ji nahřát (min. 2min). Mezi tím je vytažena DPS z ochranného obalu a filmová předloha je přiložena tonerem směrem k desce. Filmová předloha musí být k desce připevněna. Například izolepou. Nejvhodnější je vložit filmovou předlohu s DPS mezi dvě průhledné plochy (nejlépe ze skla). Je to z důvodu, aby nehrozilo posunutí filmové předlohy od DPS a hlavně aby k sobě byly neustále přitlačovány a nehrozilo podsvícení spojů. Deska musí být položena kolmo pod UV lampu ve vzdálenosti 30 až 40cm a ponechána tam 5 až 7min. Správně vyexponovaná deska se pozná tak, že deska ztmavne a cesty pod filmovou předlohou zůstanou zesvětlené. Délka expozice závisí na kvalitě UV lampy, na kvalitě filmové předlohy a na kvalitě DPS.

Při expozici oboustranné desky se postupuje úplně stejně, jen s tím rozdílem, že filmová předloha je přilepena na obě strany desky tak, že musí přesně na sebe dosedat. Pokud by tomu tak nebylo, tak by byl velký problém obě strany spojit prokovy. Pokud spolu desky přesně lícují, tak celá sestava může být vložena pod UV lampu a po vyexponování jedné strany stačí otočit desku a vyexponovat stranu druhou.

2.3.3. Vyvolání

Na vyvolávání se používá roztok 1,5% NaOH při teplotě 20 - 25°C. Z DPS musí být sundána filmová předloha, DPS se omyje pod tekoucí vodou a položí do ploché misky. Na DPS se nalije vývojka tak, aby byla celá DPS ponořena. Vyvolávána je tak dlouho, dokud na osvětlených místech nejsou patrné stopy fotolaku. Vatovým tampónem lze vyvolávání napomoci.

Po vyvolání se musí celá deska patřičně omýt pod teplou tekoucí vodou (40°C až 50°C) a osušit. Dále musí být zkontrolováno, zdali se DPS správně vyvolala a patřičné nesrovnalosti se dají opravit skalpelem (nebo nožkem) a pokud by byla některá cesta přerušena, tak ji opravíme lihovým fixem. V některých textech je doporučeno místo fixy použít nitrolak nebo lak na nehty.

2.3.4. Leptání

Na leptání se dá použít:

- chlorid železitý: cca 650g v 1l vody při teplotě 20 - 25°C
- směs kyseliny chlorovodíkové a peroxidu vodíku. Do 750ml vody se přilije za stálého míchání 150ml HCL 35% a před leptáním se přidá 100ml H₂O₂ 30%. Kyselinu je nutné přilévat do vody za neustálého míchání, ne naopak!!

Pokud je leptací roztok připraven, tak na straně DPS je potřeba vytvořit nějaké úchyty. Například z kousků izolepy, aby se leptací roztok nedostal do styku s pokožkou při manipulaci s DPS v leptacím roztoku, popřípadě, aby se dostal do styku s pokožkou jen v malém množství. Potom je opatrně uchopena DPS a položena na hladinu stranou, kterou je třeba leptat. Strana musí být dokonale suchá a čistá. Potom deska po položení na hladinu plave a nepotopí se. Roztok pak také lépe reaguje a rychleji leptá. Pokud je teplota zvýšena nad 25°C, tak leptání probíhá rychleji. Je potřeba DPS občas kontrolovat, aby nedocházelo k podleptání cest.

Pokud je leptána oboustranná deska, tak postupuje velice podobně jako u jednostranné desky. Opět je deska položena na hladinu. Ovšem je lepší stranu, která se zrovna nebude leptat nebo už je vyleptaná, nějak zabezpečit proti styku s leptadlem. Například se může strana desky přelepit izolepou.

2.3.5. Smývání

Po vyleptání je potřeba desku důkladně omýt pod tekoucí vodou od leptacího roztoku. Pokud by na desce zůstal leptací roztok, mohlo by dojít k podleptání.

Jestliže je deska špatně vyleptaná je ještě možné znovu udělat expozici a vyvolání a dát ji znovu leptat. Pokud je deska v pořádku, tak pomocí smirkového papíru se deska přebrousí. DPS je broušena tak dlouho dokud nezmění cesty barvu na sytě zlatou. V ten moment totiž měď na desce dobře reaguje s cínem a cín se po ní krásně rozlévá. Jakmile je deska přebroušena ještě zkontrolujeme jestli nejsou cesty přerušeny nebo spojeny na místech kde by být neměly přerušeny nebo spojeny. Pokud jsou přerušeny, tak se dají opravit cínem, při větším problému můžeme použít drátek. Pokud jsou spojeny, tak stačí cesty mezi sebou proškrábnout.

Když je vše hotové, tak ještě je deska očištěna acetonem, který smyje zbytek laku a přetřena ochranným lakem nebo kalafunou. Pak už jsou jen vyvrtány otvory pro součástky a u oboustranné desky i pro prokovy. [13]

2.3.6. Zásady při zpracování

1. Exponovanou desku nelze vyvolat:

- studená vývojka
- krátká doba expozice
- nízká koncentrace vývojky
- příliš stará vývojka

2. Rozpuštění fotolaku ve vývojce na celé ploše:

- vysoká koncentrace vývojky
- vysoká teplota vývojky
- průsvitná filmová předloha
- moc dlouho ponechaná DPS ve vývojce

3. Desku nelze vyleptat:

- zbytky fotolaku z důvodu krátké expozice
- nedokonalé vyvolání
- nekvalitní filmová předloha

2.4. Programování jednočipového mikropočítače ATmega128

2.4.1. Úvod do problematiky

Aby jednočipový mikropočítač dělal to, co se od něj očekává je třeba udělat dvě věci:

- Napsat program pro procesor a přeložit jej do strojového kódu
- Výsledný strojový kód naprogramovat do procesoru

Velkou výhodou procesorů Atmel AVR je skutečnost, že obě zmíněné činnosti mohou být zajištěny pomocí volných nástrojů. Tím pádem nemusí být počítáno s dalšími náklady na softwarové vybavení. Část prostředků pochází přímo od firmy Atmel a část od GNU komunity. K dispozici jsou tyto nástroje:

- Integrované prostředí a simulátor procesorů (Atmel)
- Překladač jazyka C (GNU)
- Programátor procesorů (GNU)
- Programovací kabel (GNU)

Nástroje jsou výborně funkční a bezproblémové a kromě nich jsou k dispozici i ladící nástroje.

2.4.2. Překladač C/C++ - WinAVR

Toto je připravený balík GNU nástrojů pro AVR procesory připravený pro instalaci do Windows (používá knihovny CIGWIN). Při instalaci balíku na WindowsNT/2000/XP je třeba instalaci provádět pod účtem administrátora jinak se nenastaví cesta do BIN adresáře tohoto balíku.

Překladač může fungovat sám o sobě nebo se může integrovat s balíkem AVR Studio.

Charakteristika vývojového prostředí winavr:

- editor zdrojového textu s barevným zvýrazněním syntaktických symbolů a intellisense nápovědou
- kompilátor jazyka mikroprocesorů řady AVR, včetně knihoven symbolů jednotlivých typů procesorů
- debugger pro analýzu programu, umožňující krokování a trasování programu, zobrazení I/O registrů a obsahů všech pamětí
- možnosti definovat vstupní průběhy i záznamy výstupních stavů a jejich následné zobrazení v libovolném časovém měřítku, snadné odečítání doby mezi jednotlivými

událostmi. Toto je ideální pro ladění a demonstraci činnosti dynamického displeje, sériového kanálu nebo komunikace s PC

- emulátory umožňují připojení výukových a vývojových modulů nebo přímé připojení do Vašeho zařízení s možností ladění programu přímo ve vyvíjeném zařízení
- možnost přenesení dat z reálného procesoru, in-circuit debugging
- vlastní výukové a vývojové moduly jako displeje, klávesnice, převodníky, paměti atd.
- síťový přenos umožňující vícenásobné využití připojených výukových modulů
- programátory ProgAVR, umožňující programovat všechny typy mikroprocesorů řady ATMEL AVR i přímo ve Vaší aplikaci přes rozhraní ISP
- podpora operačních systémů Windows 98, 2000 a XP

2.4.3. Integrované prostředí – AVR Studio

Jedná se o integrované vývojové prostředí pro vývoj programů pro procesory ATMEL AVR (assembler, linker) s možností integrace překladačů jazyka C/C++. Prostředí obsahuje rovněž simulátor procesorů AVR a přímo podporuje základní druhy ladících nástrojů ATMEL.

Do AVR studia lze přímo integrovat GNU překladač jazyka C/C++ pro AVR procesory. **Je vhodné nejprve nainstalovat WinAVR a teprve poté instalovat AVR Studio.**

Balík AVR Studio potřebuje ke své činnosti IE5.0 (raději 6.0, nebo balík XML od MS). [10]

2.4.4. Popis programu Komunikační jednotky

Co má vlastně program na starost? Má na starost oboustrannou komunikaci mezi PC a PLC. Program bude popsán postupně. Nejdříve komunikace mezi PLC a PC.

Mikrořadič musí načíst svoje vstupy, které vedou na výstupy z PLC. Program se podívá, na kterých vstupech má logickou jedničku a podle toho nastaví logické jedničky do svých proměnných, které byly v programu vytvořeny. Ovšem přesně obráceně. Tedy na těch vstupech kde je logická jednička, tak do proměnné nastaví logickou nulu a tam kde je logická nula nastaví do proměnné logickou jedničku. Jde totiž o to, že navržený hardware obrací logiku. Všechny proměnné (i ty, na kterých je logická nula) pak vynásobí násobky dvojkové soustavy a vznikne mu číslo, ze kterého lze jednoznačně určit nastavení vstupů v programu LabVIEW. V šestnáctkové soustavě je minimální hodnota tohoto čísla 0 0, a maximální 255 255. Tohle číslo je potom návratová hodnota funkce.

Dále je nutno inicializovat časovač TIMER1, po kterém je požadováno aby každých 50ms vyvolal přerušení. Přerušení je vyvoláno s předděličkou 1024. Aby předdělička korektně fungovala, tak se musí nastavit registry. To se udělá tak, že je zavolán časovač, načte největší a nejmenší hodnotu počítadla z inicializace a jakmile časovač vykoná přerušení, tak odešle stavy výstupů PLC do PC po sériové lince. Nejdůležitější registry jsou TCNT1H a TCNT1L.

Odesílání probíhá tak, že se odešle nejdříve kontrolní hlavička. Jsou dohromady dvě hlavičky. Ty slouží k odfiltrování jiných signálů, které by mohly narušit komunikaci. Až po odeslání hlaviček (v tomto případě s hodnotami 120 a 200) se odesílají data ze

vstupů. Příklad jak vypadá datový rámeček, který odesílá a přijímá komunikační jednotka je na obr.7.

START BYTE	START BYTE	DATA
120	200	

Obr.7.:Datový rámeček

Pokud se situace otočí a data jsou přijímány z PC, tak mikrořadič začne kontrolovat data, která přijímá. Pokud se první znak rovná první hlavičce, tak pokračuje ve čtení dalšího znaku. Pokud je tomu jinak, tak stále čeká na první znak. Jako další znak je na řadě druhá část hlavičky a až teprve po ní přicházejí znaky s nastavením výstupů mikrořadiče ve stejném kódu v jakém mikrořadič odesílal svoje vstupy.

Jakmile jsou všechny čtyři znaky přijaté zavolá se funkce Set_Outputs (nastavení výstupů), která má dva parametry. V tomto případě dva znaky, které přijdou po hlavičkách. Tyhle dva parametry funkce Set_Outputs bude porovnávat (maskovat) s osmi bitovým číslem a pokud se bude rovnat, tak přiřadí hned na port logickou jedničku. Jinak zůstává port nastaven jako logická nula.

V hlavním programu „main“ už je pouze zařízeno, aby proběhly všechny inicializace. Tedy nejdříve musí zakázat všechna přerušení, nastavit veškeré inicializace a spustit zdroj časovače. Poté musí znovu povolit všechna přerušení a přejít do cyklu, aby všechny funkce proběhly.

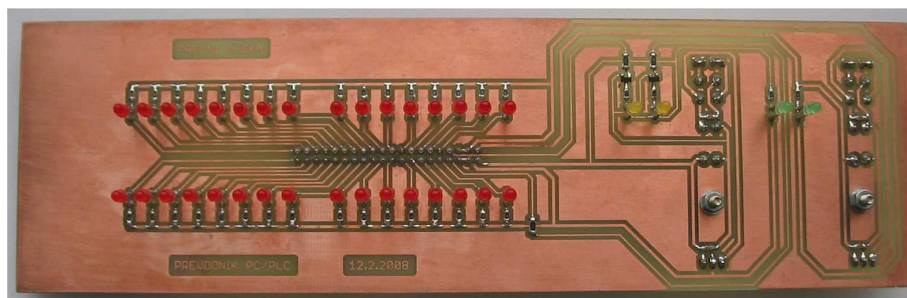
Program je uveden mezi přílohami, včetně popisu každého řádku programu (Příloha 8.).

2.4.5. Seznam součástek

Seznam a ceník součástek s aktuálními cenami viz. Příloha 9. odkrývá celkové náklady na výrobu Komunikační jednotky. Je to vlastně jeden z hlavních důvodů proč byla tahle komunikační jednotka vyráběna. Cena oproti obdobné jednotce vyskytující se na trhu je desetkrát až čtyřicetkrát nižší. Kód součástky je v tabulce dle GM electronics.

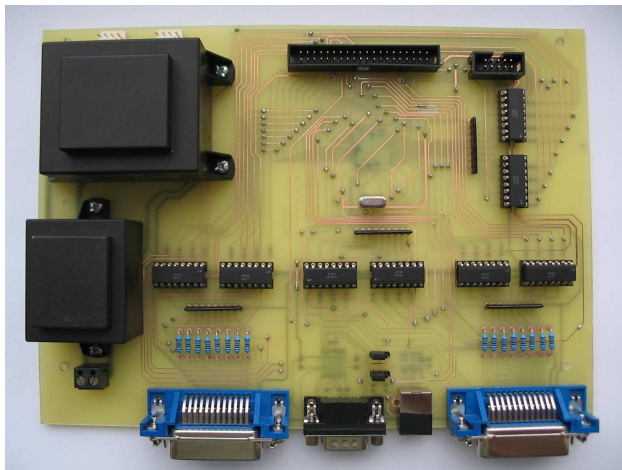
2.4.6. Komunikační jednotka - finále

Jakmile je komunikační jednotka vyrobena, tak přijde na řadu testování.



Obr.8.:Přední deska

Přední deska na obr.8. je testována relativně snadno. Vlastně všechny funkce detekují LED diody. Pokud jsou v pořádku tak se dá mluvit o jistotě, že je Přední deska v pořádku.



Obr.9.:Hlavní deska

S hlavní deskou je to složitější (obr.9.). Testuje se, zdali jsou přerušené cesty, zdali jsou všechny součástky v pořádku, zdali je v pořádku napájení a hlavně funkčnost mikrořadiče ATmega128. Kromě těchto chyb, které mohly vzniknout výrobou, je nutné se zaměřit i na chyby, které mohl zapříčinit špatný návrh desky.



Obr.10.:Hlavní deska

Na obr.10. je zachycena hotová komunikační jednotka. Navržená, vyrobená a testovaná.

3. PROGRAMOVÁNÍ V LABVIEW

LabVIEW je zkratkou pro Laboratory Virtual Instrument Engineering Workbench od firmy National Instruments. Jedná se o moderní výkonný systém speciálně vhodný pro programování komunikace osobního počítače s různými periferními zařízeními, zejména s měřicími přístroji.

LabVIEW lze chápat jako vývojové prostředí nad jazykem G. Narozdíl od Object Pascalu, kde zdrojovým kódem programu je text, zdrojovým kódem programu v jazyce G je obrázek. Místo aby byly programy psány, budou kresleny. Jazyk G tedy značí grafický programovací jazyk. Grafické programování je technika neobvyklá a nová. Byla patentována společností National Instruments teprve v roce 1990. Pro jazyk G existuje kompilátor, který produkuje samostatné spustitelné programy. Tvůrci LabVIEW prohlašují, že programy vytvořené v jazyce G běží po přeložení srovnatelně rychle, jako programy napsané v jazyce C, který je obecně považován za velmi efektivní. V jazyce G má programátor k dispozici jak rychlé funkce nízké úrovně, tak i hotové komplikované podprogramy pro matematickou analýzu, statistiku, komunikaci se standardizovanými periferiemi a podobně.

V současné době neexistuje česká lokalizace LabVIEW, pouze anglická, německá, španělská a japonská. Pokud je autorovi známo, neexistuje ani česká učebnice LabVIEW. Naprostá většina akcí se provádí myší. [1]

3.1. Úvod do programování

LabVIEW vyvíjí společnost National Instruments. Uživatelské rozhraní programu LabVIEW se dost podobá skutečným měřicím přístrojům. Proto se mu také říká virtuální přístroj. Anglický ekvivalent je „Virtual instruments“ a jeho zkratka je VI. Tuhle zkratku National Instruments používá dosti často ve svém textu v příručkách k LabVIEW.

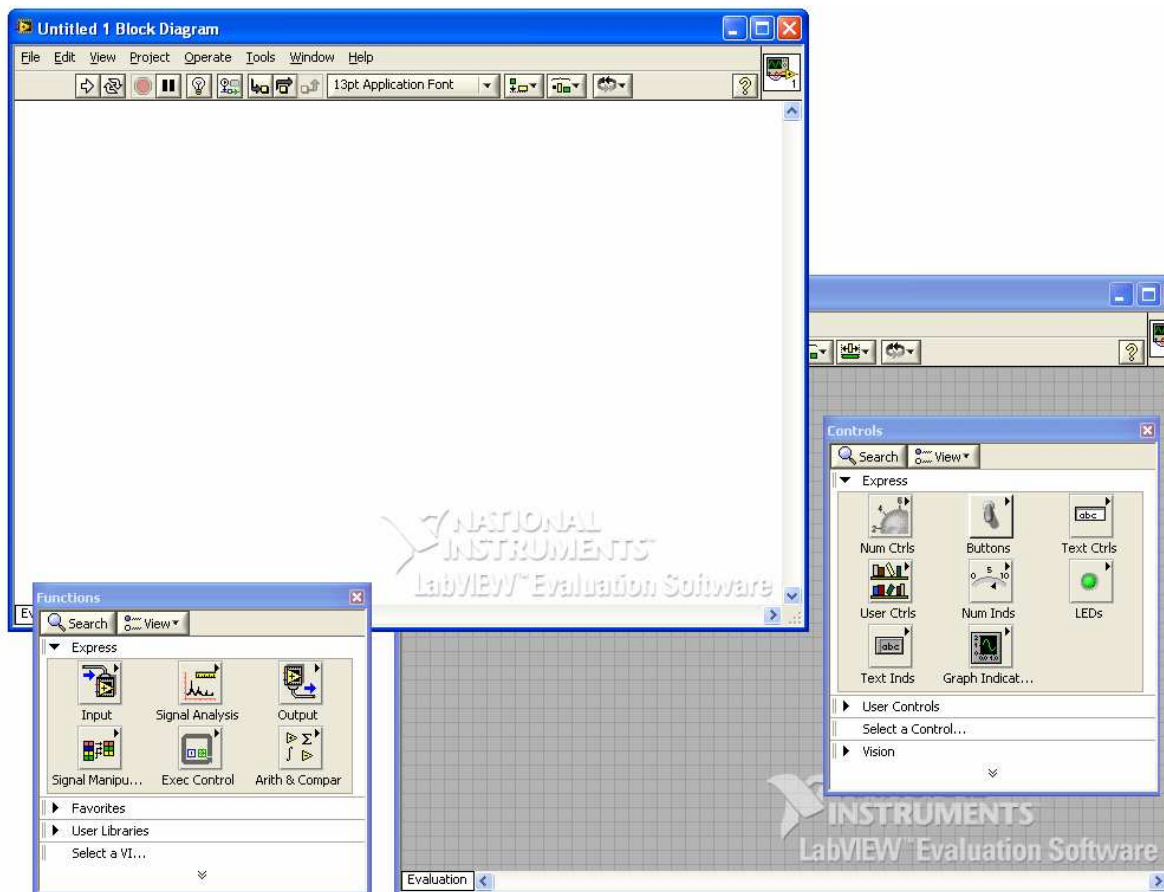
K vytváření VI slouží vývojové prostředí LabVIEW. Po otevření programu LabVIEW, se na monitoru objeví startovací nabídka (tzv. „Getting Started“). Startovací nabídka je rozdělená na soubory („Files“) a zdroje („Resources“). Soubory jsou dále rozděleny na vytvoření nového VI, projektu nebo VI ze šablony a na otevření VI ze souboru. Ve zdrojích se nachází různé nápovědy.

Pokud bude otevřen nový VI, tak se na monitoru objeví dvě okna. Front panel a Block diagram (viz. obr.11.). Pokud by tyto okna byly přirovnána k nějaké bedně (například skříň rádia nebo výše zmíněné komunikační jednotky), tak Front panel je strana kde je veškeré ovládání a indikace a Block diagram je veškerá elektronika a řízení, které je ukryto v bedně. Tedy Block diagram slouží k programování VI a jako výsledný produkt vznikne ve Front panelu.

Dále jsou na obr.11. znázorněny dvě palety. Paleta Function patří k blokovému diagramu a jsou zde funkce a proměnné potřebné k sestavení programu. Má stromovou strukturu a zobrazí se pokud se pokliká pravým tlačítkem myši do blokového diagramu. Po zobrazení se u názvu palety objeví nepřipíchnutý připínáček, pokud se pokliká na něj, tak paleta zůstane po celou dobu u blokového diagramu. Jinak po každém použití funkce se musí paleta znovu vyvolávat.

Paleta Controls patří k čelnímu panelu. Obsahuje veškeré indikátory a řídicí součásti. Má také stromovou strukturu a stejně tak i připínáček u názvu palety ze stejného důvodu jako paleta Function.

Další zajímavostí je, že jednotlivé palety jsou viditelné pouze, když je aktivní dané okno buď blokového diagramu nebo čelního panelu. To znamená, že nemohou být aktivní obě palety současně.



Obr.11.: Přední panel a Blokový diagram v LabVIEW

3.2. Úvodní program v LabVIEW

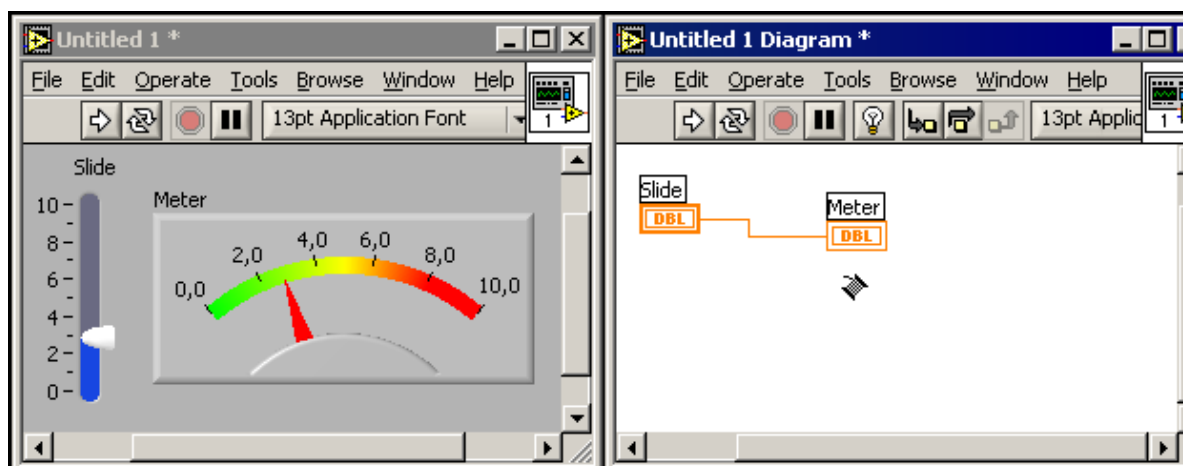
Proniknout do základu programování v LabVIEW je snadné. Program má příponu *.VI. Aniž bychom o programování v LabVIEW cokoli věděli, pokusíme se krok za krokem sestavit první funkční VI. Čelní panel našeho VI bude sestávat ze dvou objektů: z tahového potenciometru a z ručkového měřicího přístroje. Náš VI nebude provádět nic jiného, než že hodnotu nastavenou na tahovém potenciometru zobrazí na měřicím přístroji. Jinými slovy, pohneme-li s jezdcem tahového potenciometru, pohne se i ručka měřicího přístroje.

1. Klepneme pravým tlačítkem myši do prostoru okna čelního panelu Untitled 1. Objeví se paleta Controls. Paleta je obrázková, ale v horní části se zobrazuje i krátký textový popis vybrané položky. Ve stromové struktuře palety zvolíme Controls ! Numeric-Vertical Pointer Slide. Opětovným klepnutím do prostoru okna čelního panelu umístíme Vertical Pointer Slide do libovolné pozice v okně. Takto jsme umístili tahový potenciometr na čelní panel našeho VI.

2. Stejným způsobem umístíme na čelní panel objekt Meter. Má podobu ručkového měřicího přístroje a nachází se také v nabídce Numeric palety Controls, stejně jako tahový potenciometr. Odtud označení Controls ! Numeric-Meter. Tím je návrh čelního panelu našeho VI ukončen a můžeme se pustit do tvorby vlastního programu, tedy do tvorby blokového diagramu. Bude to velmi snadné. V zápětí totiž zjistíme, že celý blokový diagram je již v podstatě vytvořen.

3. V okně blokového diagramu Untitled 1 Diagram se objevily 2 nové objekty. Jak lze očekávat, objekt s popiskem Slide přísluší z tahovému potenciometru, druhý objekt s popiskem Meter ručkovému měřicímu přístroji. Objekt Slide odráží polohu jezdce potenciometru. Objekt Meter určuje polohu ručky měřicího přístroje. Všimneme si, že při stisku klávesy tab se mění vzhled kurzoru. Klávesou tab tedy zvolíme kurzor tvaru cívky s nití.

4. Intuitivně řečeno, nyní musíme spojit polohu jezdce potenciometru s polohou ručky přístroje. Když nyní klepneme myší postupně na objekty Slide a Meter, zjistíme, že se spojily červenou čarou.



Obr.12.:Příklad programu v LabVIEW

Tím je celý program hotov a nezbývá než jej vyzkoušet. VI spustíme tlačítkem Run Continuously. Pokud program běží, můžeme pomocí myši pohybovat posuvným potenciometrem Slide. Zároveň se hýbe i ručka měřicího přístroje Meter. Běžící program zastavíme tlačítkem Abort Execution. Nyní náš VI uložíme (Ctrl-S) do jediného souboru s příponou vi, jak je obvyklé, a zapamatujeme si jeho umístění, abychom s ním mohli později manipulovat. [1]

4. SIMULACE EDU-MOD VYTVOŘENÉ V LabVIEW

Simulační jednotky Edu-mod vyrábí firma TECO a.s. za účelem výuky automatizační techniky na školách technických směrů. V současné době si výuku totiž nedokážeme představit bez použití prostředků z oblasti číslicové techniky. Mezi nejpoužívanější číslicové systémy, které jsou dnes nasazeny ve výrobních i nevýrobních aplikacích jsou programovatelné automaty (PLC).

Jednotky Edu-mod vznikly jako součást programu EDUtec, který byl připraven pro zkvalitnění výuky automatizace na středních, vyšších a vysokých školách. Simulační moduly Edu-mod byly vyvinuty, aby doplnily tento program. Jsou souborem modelů „technologických procesů“ určených především k praktické výuce logických systémů realizovaných programovatelnými automaty (PLC), řídicími počítači, stavebnicemi logických obvodů (např. Dominoputer), PLD obvody, případně jednočipovými mikropočítači. Řízené objekty jsou prezentovány řadou periferních modulů osazených jednočipovými mikropočítači, které simulují funkci reálného objektu.

Určení pomůcky Edu-mod:

EDU-mod je soubor modelů “technologických procesů” určený především k praktické výuce logických systémů realizovaných programovatelnými automaty (PLC), řídicími počítači, stavebnicemi logických obvodů (např. Dominoputer), atd. Podle napěťové úrovně logických signálů se vyrábí ve dvou základních řadách.

Moduly EDU-mod řady 24V:

Logické signály s úrovní 24V ss umožňují univerzální použití pro libovolný typ PLC systému. Vstupní i výstupní signály jsou definovány proti společnému zápornému vodiči. Opačnou polaritu signálů je možno řešit přizpůsobovacími členy.

Moduly EDU-mod řady 5V:

Logické signály s úrovní TTL dovolují spojení s logickými automaty realizovanými na bázi stavebnic číslicových IO, programovatelných logických polí (PLD), procesorových obvodů atd.

Základní součásti EDU-mod:

- model křížovanky
- model mísicí jednotky
- model hydraulické posuvové jednotky
- model automatické pračky
- model soustavy pro regulaci spotřeby
- propojovací modul, kabely, základní deska
- dokumentace, učitelská disketa

Vstupní a výstupní signály - připojení k řídicímu systému

Vstupní a výstupní signály EDU-mod jsou vyvedeny na 20-ti pólový konektor zajišťující propojení plochým kabelem s rozbočovacím modulem s mechanickým provedením shodným s periferiemi EDUtec. Rozbočovač obsahuje 4 konektory Cannon 9 (2 vstupní, 2

výstupní), pro připojení max. 8 vstupních a 8 výstupních binárních signálů z/do libovolného systému. Chceme-li tedy použít zvolený modul, stačí jej kabelem připojit a do odpovídajících konektorů zapojit řídicí systém (např. EDUtec).[14]

4.1.Sériová komunikace v LabVIEW

V LabVIEW verze 8.2 a vyšší jsou na sériovou komunikaci vytvořeny bloky. Vkládají se v okně Block diagram a nalézt je lze v paletě *Function > Instrument I/O > Serial*.

Nejdříve je vložen blok *VISA Configure serial port*. Je to vlastně před vytvořením VI vložené jako blok. Přes konfiguraci portu lze nastavit 9 různých vlastností portu, které budou nyní popsány.

Ve *VISA resource name* je volen port na kterém bude sériová komunikace probíhat. To je dá se říci jediná volba, která musí být za každou cenu nastavena. Zbytek vlastností má default-ní nastavení. Další co lze nastavit je *Baud rate*, tedy rychlost komunikace, která je nastavena default-ně na 9600 baudů. *Data bits* jsou přednastaveny na 8-mi bitovou komunikaci. *Parity* komunikace není žádná. *Stop bits* je přednastaven na jeden stop bit. Jedná se o specifikaci, která indikuje konec rámce. *Flow control* slouží k nastavení typu kontroly v mechanismu odesílání. Ještě stojí za zmínku *Timeout*, kde se nastavuje maximální hodnotu pro odesílání a zápis.

Do dalšího bloku pokračují signály *VISA resource name* označené fialovou barvou a *Error* signál, který je označen hnědou barvou. Další blok se nazývá *VISA Set I/O Buffer Size*. Slouží k nastavení velikosti přijímaných nebo odesílaných dat. Pokud je nastaven kód „16“, tak jak je v tomto případě nastaveno, jedná se o přijímaná data. Pokud by místo 16-ti bylo nastaveno 32, jednalo by se o odeslaná data. Hned pod nastavením přijímání nebo odesílání je nastavena maximální velikost datového rámce. Ta by měla být nastavena tak, aby pokryla komunikaci, která je požadována. Tedy mírně větší číslo, než je samotná komunikace.

Blok *VISA Flush I/O Buffer* slouží k přetečení bufferu podle masky. Masky má čtyři možné nastavení viz Tab.5. V tomto případě se zdálo nejrozsudnější nastavení maskování na hodnotu „64“.

Proměnná masky	Hexadecimální kód	Popis
16	0x10	Přetečení a vyřazení obsahu po přijetí v bufferu
32	0x20	Přetečení a vyřazení obsahu po odeslání z bufferu
64	0x40	Přetečení a vyřazení obsahu po přijetí v bufferu (nefungují vstupy a výstupy do zařízení)
128	0x80	Přetečení a vyřazení obsahu po odeslání z bufferu (nefungují vstupy a výstupy do zařízení)

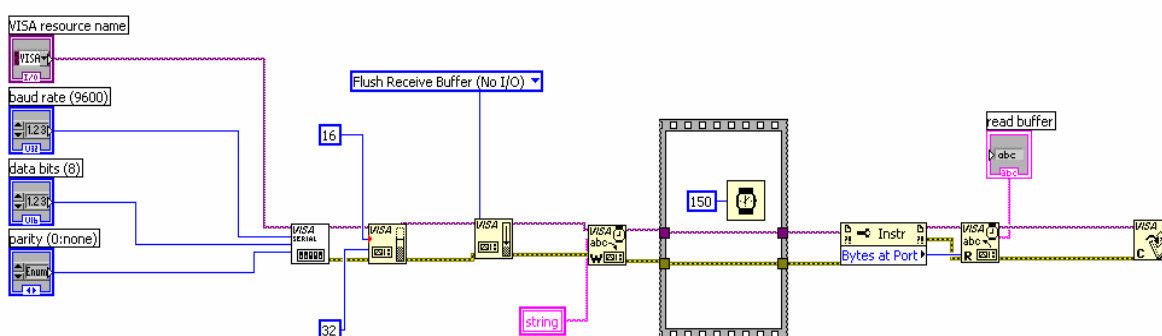
Tab.5.: Maskování u VISA Flush I/O Buffer

Po těchto dvou blocích, které mají na starost přijímání pouze požadovaných datových rámců, následuje blok zapisování do bufferu. Blok se nazývá *VISA Write*. Jako vstup do bloku *VISA resource name*, *Error* a *String*. *String* je hodnota datového typu řetězec. O jejím získání bude řeč dále v textu. Tato hodnota se zapíše do bloku a pošle dál na výstup bloku po *VISA resource name*. Druhý výstupní signál je *Error*.

Oba dva signály přejdou přes časovač nastavený na 150ms, který je umístěn ve struktuře *Flat Sequence Structure*, která má na starost fungovat tak dlouho, dokud je časovač spuštěn. Proto je zvolena tato struktura. Komunikace tedy díky tomuto časovači každých 150ms odešle a přečte data na sériové lince.

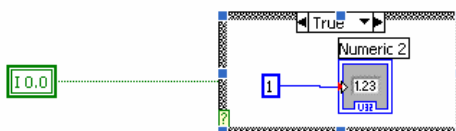
Signály směřují do *Property Node*, ze kterého spoustu informací lze sejmut a použít v blocích a funkcích sériové komunikace. Tenhle blok je zde použit, aby přibyl ještě jeden výstup a to počet bytů na portu. Tyhle tři signály pak pokračují do *VISA Read*, která má pak na starost přečíst buffer pomocí *Read buffer* v datovém typu taktéž řetězce. O Read buffer bude dále v textu řečeno něco víc. Další dva signály, které vystupují z bloku pak směřují do *VISA Close*, která uzavírá komunikaci.

Jak taková sériová komunikace v LabVIEW vypadá je ukázáno na obr.13.



Obr.13.:Sériová komunikace v LabVIEW

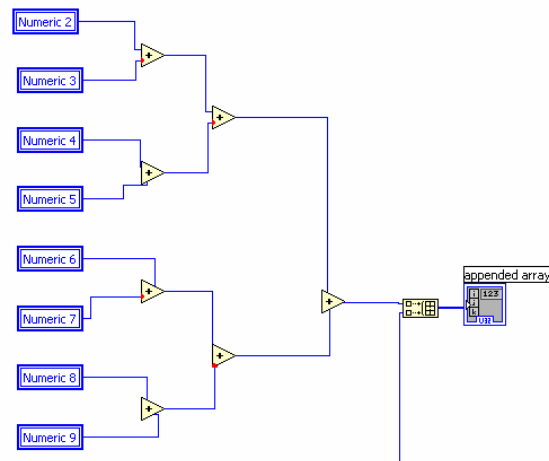
Výše jsem se zmínil o proměnné *string*, která je naplněna daty k odeslání. Nyní bude popsán vznik těchto dat. IO.0 je proměnná *boolean*, která pokud je true, tak v tom případě se indikátor *Numeric 2* naplní jedničkou viz. obr.14.



Obr.14.:Naplnění proměnné

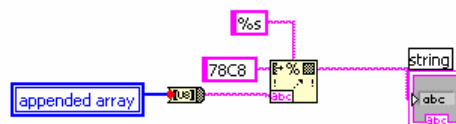
Jestliže I0.1 je true, potom indikátor *Numeric 3* se naplní dvojkou. Když I0.3 je true, tak se naplní indikátor *Numeric 4* čtverkou a tak dále až po I1.7 jeli true, tak *Numeric 16* se naplní hodnotou 128. Je to vlastně úplně stejný postup jakým se programoval jednočipový mikropočítač ATmega128. Jakmile je naplněno všech 16 proměnných, tak musí být z každého z indikátorů vytvořeny lokální proměnné (viz. obr.15.).

Lokální proměnná se vytváří tak, že je poklikáno pravým tlačítkem myši na Indikátor *Numeric 2* na obr.14., v nabídce je vybráno *Create -> Local Variable* a lokální proměnná je přesunuta v blokovém diagramu tam, kam je potřeba. Lokální proměnná je naplněná stejnou hodnotou jako Indikátor. Nejdříve bude vybráno prvních 8 proměnných a budou posčítány pomocí bloku *Add*, který se nachází ve *Functions -> Programming -> Numeric*. Maximální hodnota, která takto může vzniknout je 255 a minimální je 0 stejně jako u mikrořadiče ATmega128. To samé bude uděláno i s dalšími 8 proměnnými, které budou taky posčítány a vzniknou stejné číselné hodnoty jako v prvním případě. Vzniknou



Obr.15.:Data na výstupu

tím pádem dva signály, které by měly být taktéž spojeny. Proto se použije blok *Build Array* tedy „postav pole“. Zleva do bloku budou přivedeny oba signály od proměnných. Blok je bere jako elementy, které vkládá do sloupce pole. V tomto případě se dva elementy rovnají velikosti dvou řádků a jednoho sloupce matice na výstupu z bloku. Matice může tedy nabývat hodnot od $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ do $\begin{bmatrix} 255 \\ 255 \end{bmatrix}$. Z pole musí být vytvořena lokální proměnná stejným postupem jako z indikátorů. Tuhla lokální proměnná, která je naplněna polem appended



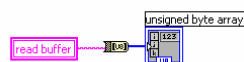
Obr.16.:Vytvoření string (char)

array, musí být převedena na proměnnou řetězec. To se dělá v LabVIEW pomocí bloku *Byte Array To String*. Tak se převede signál pole z *integeru* do *charu*. Tento řetězec musí být ještě naformátován. To se provede pomocí bloku *Format Into String*. Musí být přiveden do bloku signál převedeného pole. Jako hlavičku před data z pole bude dána konstanta 78C8 (viz obr.16.). Pokud je konstanta převedena do desítkové soustavy, tak se jedná o čísla 120, 200, stejně jako u hlavičky u jednočipového mikropočítače ATmega128. Na obrázku je ještě jeden signál, který je přiváděn a to %s, jedná se o již zmíněné formátování. V tomto případě %s znamená klasický formát *String*. Další možné formáty jsou například *string* jako *hexadecimální integer*, nebo časová známka a podobně. A jako výstup se naplní Indikátor string.

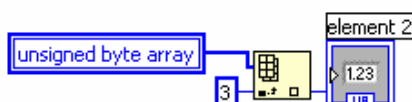
Z tohoto indikátoru bude vytvořena lokální proměnná a přivedena k *VISA Write* na obr.16. a vlastně už je hotový datový rámec, který bude sériová linka odesílat.

Při čtení z bufferu pro příjem je nejdříve potřeba vytvořit lokální proměnnou z Indikátoru *Read buffer* na obr.17. Lokální proměnná je v datovém typu *char*, nebo podle LabVIEW je označována jako *string*. Proto bude použit blok z LabVIEW *String To Byte Array*, tedy pravý opak jako v předchozím případě. Pole potom bude mít podobu jednoho sloupce a čtyř řádků. První dva řádky patří k hlavičce a druhé dva k nastavení portů.

Minimální podoba pole je $\begin{bmatrix} 120 \\ 200 \\ 0 \\ 0 \end{bmatrix}$ a maximální zase $\begin{bmatrix} 120 \\ 200 \\ 255 \\ 255 \end{bmatrix}$. Pole se nazývá

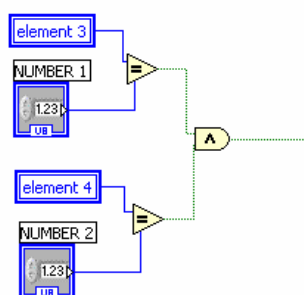


Obr.17.: Načtení buffferu



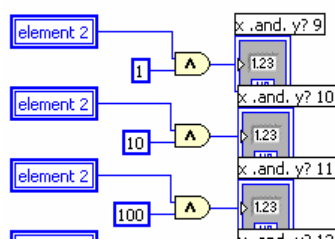
Obr.18.: Rozložení pole

unsigned byte array a v dalším kroku je nutné jednotlivé hodnoty řádků rozložit a zpracovat. Na obr.18. je zobrazeno jak se dá z pole vybrat řádek a zapsat jej do Indikátoru. Používán je k tomu blok *Index Array*, do kterého vstupuje dané pole a dále index, v tomto případě číslo řádku, který má být vybrán a zapsán do Indikátoru. Indexování je zde voleno od nuly, tedy číslo tři v tomto případě označuje čtvrtý řádek. Stejně tak jsou zvoleny i předchozí dva řádky matice a získány čtyři hodnoty v integeru. První dvě jsou hodnoty hlavičky a druhé dvě jsou zakódované informace o nastavení výstupů programovatelného automatu, tedy vstupů simulace v LabVIEW.



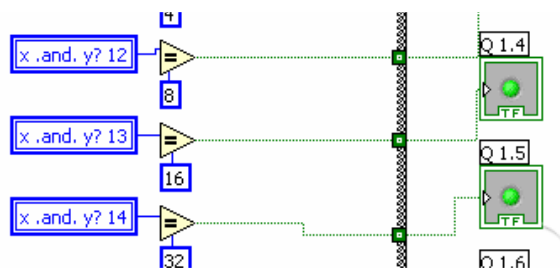
Obr.19.: Ověření podmínky

Na obr.19. je znázorněna podmínka, při které bude otevřena *CASE structure*. Tedy struktura funkce CASE. Ta funguje tehdy a jen tehdy platí-li, že hlavička obsahuje číselnou kombinaci 120 a 200. Udělá se to tak, že struktura CASE se otevře pokud dá podnět nějaká proměnná *boolean*. Vysvětleno na případě, pokud lokální proměnná *element 3* je rovna Indikátoru *Numer 1*, který má nastavenou hodnotu na 120 a zároveň (tedy bude používán logický součin AND) *element 4* je roven Indikátoru *Numer 2*, který je nastaven na hodnotu 200. Pokud by se *element 3* nebo *element 4* rovnal nějakým jiným číslům, podmínka by to vyhodnotila jako false a struktura CASE by neproběhla. Tak vypadá ošetření proti přijetí špatných informací.



Obr.20.:Maskování

Pokud obě hlavičky jsou správně, tak je otevřena CASE structure. Uvnitř se nejdříve musí vymaskovat z *elementu 1* a z *elementu 2* hodnoty. Je to uděláno tak, že se elementy porovnávají s konstantou. Konstanta ale není číslo v desítkové soustavě, ale v binární. Na obr.20. je vidět, jak je porovnáván *element 2* s konstantou, která má hodnotu 1. Jednička tu představuje hodnotu 2^0 . Pokud tomu tak je, tak se naplní hodnota do



Obr.21.:Závěr komunikace

indikátoru *x. and y?* 9. To platí i pro maskování mezi *elementem 2* a hodnotou konstanty 10. Tak je označeno číslo v binárním kódu, tedy $1.2^1 + 0.2^0$ a tak by se dalo pokračovat s popisem dále. Do Indikátoru se naplní čísla buď 0 nebo hodnota konstanty. Pokud tedy by hodnota konstanty byla binární číslo 1000, tak by hodnota Indikátoru byla 8 v desítkové soustavě.

Jakmile jsou všechny indikátory naplněny hodnotami, ať nulovými nebo s daným číslem, tak je potřeba z těchto indikátorů vytvořit lokální proměnné a tyto proměnné porovnat s hodnotami a pokud nejsou naplněny nulou, tak by měla naplnit booleanovský indikátor (v LabVIEW jej představuje dioda – svítí-li, tak je naplněna logickou jedničkou, nesvítí-li, pak je tam nula pokud není obvod negován). Tak se nám rozsvítí výstupy z automatu jako vstupy v LabVIEW (viz. obr.21.).

Komunikace po sériové lince v LabVIEW musí fungovat skoro stejně jako komunikace v komunikační jednotce. Jediný rozdíl je ten, že tady se odesílá a přijímá pouze pomocí časovače, i když v komunikační jednotce se využívá přerušení.

4.2.Suport (Hydraulická posuvná jednotka)

Jako první simulace modulů EDU-mod bude uveden suport.

Funkce modulu:

- Pohyb suportu je simulován pomocí deseti LED diod, z toho čtyři mají zároveň funkci snímačů polohy K1 až K4.

- Model je řízen třemi výstupními signálovými bity. Výstup EM1 řídí pohyb suportu vpřed, EM2 řídí pohyb vzad a EM3 ovládá dvoupolohově rychlost (EM3 = 1 pracovní posuv).

Inicializační stav:

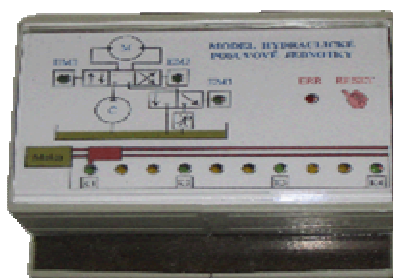
- Po zapnutí napájení nebo po restartu (tlačítko RESET) se suport automaticky nastaví do inicializačního stavu - pozice na snímači K1.

Chybová hlášení:

Model dokáže vyhodnotit dva druhy funkčních chyb:

- Přejezd krajního snímače (K1, K4)
Rozsvítí se červená LED ERR a zelené LED snímačů K1 až K4. Systém se vrátí do výchozího stavu po stisku tlačítka RESET.
- Současné sepnutí EM1 a EM2

Signalizační dioda ERR bliká, po odstranění chybového stavu systém bez restartu pokračuje v činnosti. [14]



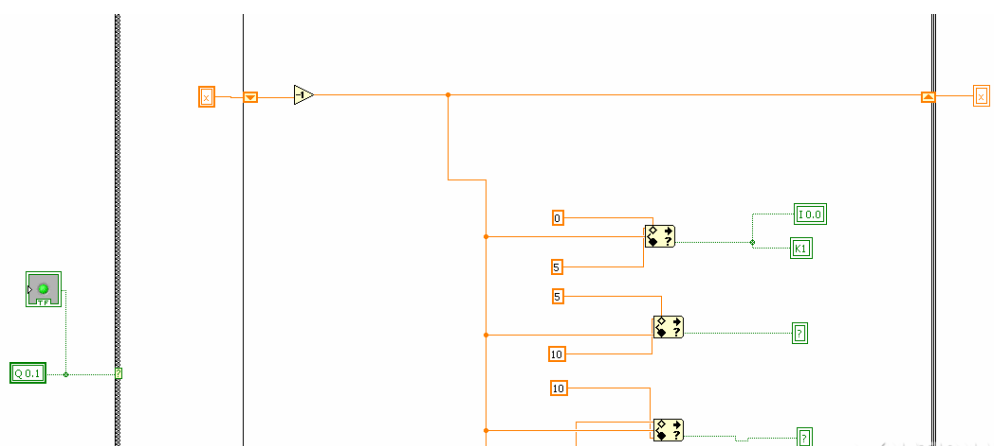
Obr.22.: Suport [14]

Na obr.22. je znázorněn model EDU-mod. Ve spodní části modelu je 10 LED diod, které signalizují pohyb suportu. Z toho 1., 4., 7. a 10. LED dioda se chová jako čidlo polohy suportu. Jsou označeny K1, K2, K3 a K4. Další LED diody na obrázku EM1 a EM2 jsou solenoidy, tedy elektromagnety, které mají na starost pohyb suportu doprava nebo doleva. Elektromagnet EM3 jestliže je zapnut, slouží k přibrzdování pohybu. Poslední LED dioda na modelu je ERR a signalizuje chybové stavy modelu. Chybové stavy jsou lépe popsány výše v kapitole Chybová hlášení. Pokud nastane chyba, tak je na modelu tlačítko RESET, který tuto chybu má na starost odblokovat a nastavit model do výchozí pozice.

Tak tady je přesně popsáno jak by měl model EDU-mod pracovat a teď tedy jak pracuje simulace?

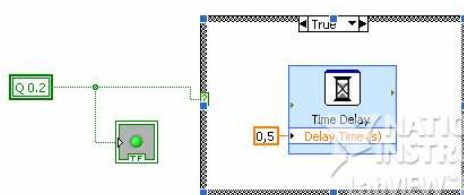
hodnotami 0 a 5, tak svítí LED dioda tohoto intervalu a hodnota 5 patří do tohoto intervalu. Hodnota 5 ale také patří do intervalu 5 až 10. Pokud ovšem svítí LED dioda, která patří do intervalu 0 až 5, tak se automaticky z intervalu 5 až 10 stane interval 6 až 10. Tím chce autor říci, že pokud nastane stav, že je hodnota rovná pěti, tak nesvítí obě dvě LED diody zároveň.

Ve zkratce tedy se k hodnotě přičítá jednička po 250ms a postupně se rozsvěčují a zhasínají LED diody. Ty mají na starost simulovat polohu suptu. Důvod proč je použitý blok s intervaly a není stanovena přímo jedna hodnota, které se rovná daná LED dioda je ten, že komunikace mezi PC a PLC trvá zhruba 3 kroky. To znamená, že by v simulaci svítila LED dioda, ale PLC by dostávalo informace, že svítí LED dioda ještě o 1 a půl kroku dříve. Zvolen interval o šířce 5. hodnot byl ten, že je potřeba mít rezervu pro případ například rušení v komunikaci.



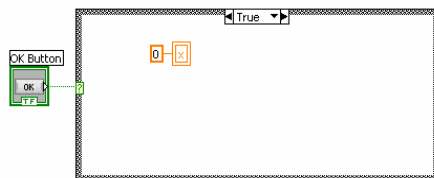
Obr.24.: Suport – pojezd doleva

Na obr.24. je zobrazen pojezd doleva programovaný v LabVIEW. Funguje téměř totožně jako pojezd doprava. Rozdíly mezi oběma částmi programu tedy jsou tyto. Jako první rozdíl je, že se zapíná struktura CASE pomocí signálu z Q0.1. Pokud je proměnná Q0.1 naplněná logickou jedničkou tak se rozsvítí LED dioda EM2. V prvním případě se rozsvítila LED dioda EM1. Opět se zapne časovač *Time Delay* 3 na stejnou hodnotu jako *Time Delay* 2, tedy 250ms. Co se týká nastavení větve, která má nastavovat rozsvícení LED diod, které označují polohu suptu, tak ta zůstane úplně stejná jako v prvním případě. Do cyklu *For* se načte lokální proměnná *x*, která je naplněna stejnou hodnotou jako proměnná *x* v případě pohybu doprava. Na vstupu z cyklu je také lokální proměnná *x*. Tím se zaručí, že si bude program pamatovat, kde má pokračovat od předchozí iterace. A další rozdíl je ten, že místo bloku +1 bude používán blok -1, takže se hodnota bude o jedničku snižovat a tím pádem se budou LED diody rozsvěcet pozpátku.



Obr.25.: Suport – EM3

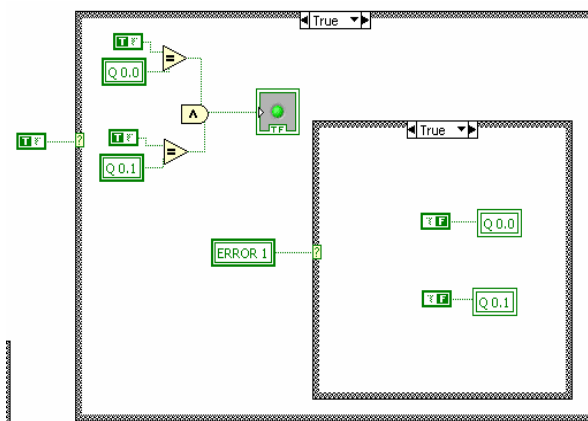
Pokud se na výstup automatu Q0.2 dostane logická jednička, tak vstup Q0.2 v LabVIEW přepne strukturu CASE z false na true viz. obr.25. Kromě přepnutí struktury ještě přivede logickou jedničku na LED diodu EM3. EM3 je vlastně simulace pracovního posuvu, proto uvnitř struktury CASE je pouze časovač a je nastaven na zpoždění 500ms. Tím pádem se LED diody pohybují jak doprava tak doleva o polovinu pomaleji.



Obr.26.: Suport – RESET

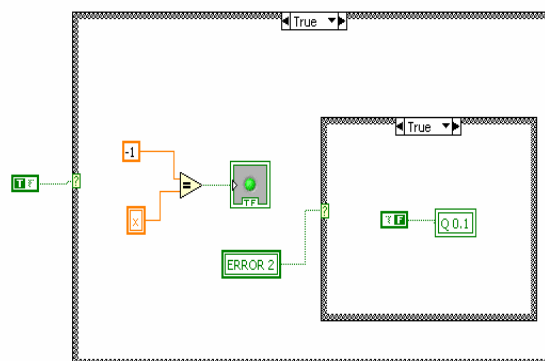
Po stisknutí tlačítka RESET se celá simulace dostane do výchozího bodu. To se dá udělat velice jednoduše. Opět se využije CASE struktura a vloží se do ní, že lokální proměnná x se bude rovnat konstantě 0 (viz. obr.26.). To znamená, že se suport dostane do výchozí pozice a případný ERROR, který by se týkal například nárazu suportu do vřetena by zhasnul LED diodu ERROR.

V simulaci jsou použity 3 LED diody na ERROR hlášení na rozdíl od modelu EDU-mod. První LED dioda slouží k zaznamenání ERROR-ového stavu pokud jsou zapnuty oba pohyby zároveň, jak doleva, tak doprava. Celá procedura musí být ukryta ve struktuře, která je neustále nastavena do stavu logické jedničky konstantou True (viz. obr.27.). Uvnitř procedury je podmínka, že se vstupy v LabVIEW Q0.0 a Q0.1 musí rovnat konstantě true a to tak, že zároveň, jedině pak se rozsvítí ERROR 1. A pokud svítí LED dioda ERROR 1, pak se otevře druhá struktura a to ta, že se nastaví vstupy Q0.0 a Q0.1 na logickou nulu. Díky nadřazenostem mezi jednotlivými strukturami se konstanty *boolean* mezi sebou nijak nepřeskakují. V simulaci svítí ERROR a suport stojí na pozici, kde ERROR vzniknul.



Obr.27.: Suport – ERROR 1

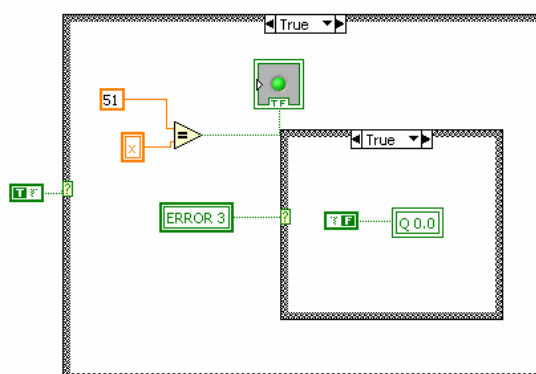
ERROR 2 se rozsvítí v tom případě, že suport narazí do obrobku zprava do leva. Znamená to, že přejede první LED diodu. Tedy v programu to vypadá tak, že jestliže se do proměnné x dostane menší číslo, než je jeho nejmenší hodnota (v tomto případě 0), tak se musí inicializovat ERROR a zastavit pohyb suportu, tedy Q0.1 musí být konstantně nastaven na logickou nulu. První číslo menší než 0 je v *integeru* -1 a pokud se tomuto číslu rovná proměnná x , tak pak se inicializuje již zmíněný ERROR (viz. obr.28.).



Obr.28.:Suport – ERROR 2

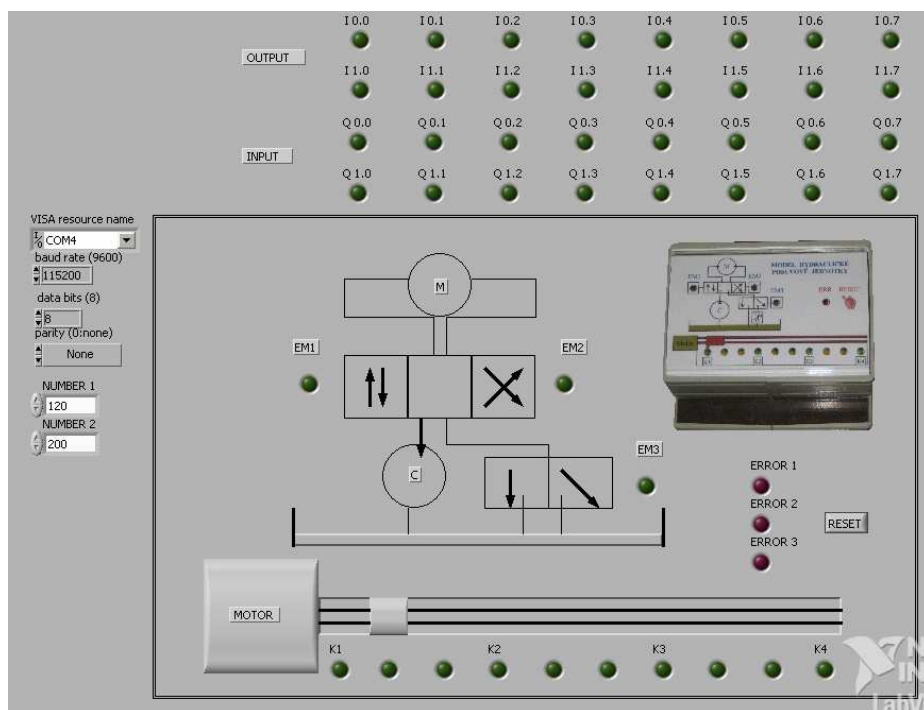
Co se týká nárazu suportu zleva doprava, tak to je ošetřeno dle obr.29. Rozsvítí se třetí ERROR LED dioda a to tehdy, pokud proměnná x je naplněná větší hodnotou, než jakou pojme poslední LED dioda, která inicializuje polohu suportu. V tomto případě je to číslo 51. Pak se výstup z programovatelného automatu a tedy vstup do simulace LabVIEW nastaví do konstantní proměnné a to logické nuly.

Tím by bylo programování v blokovém diagramu ukončeno. Na obr.30. je znázorněn panel přední desky. LED diody nahoře indikují zapínání vstupů a výstupů. Je vidět, že vstupy jsou netradičně označeny Q0.0 až Q1.7 a výstupy I0.0 až I1.7. Je to z důvodu, aby se snáze dal připojit programovatelný automat. To znamená, že LED dioda označená jako Q0.0 se připojí na výstup programovatelného automatu Q0.0. Nemusí se pak díky značení přemýšlet nad tím, jak připojit jednotlivé vývody. Dále může být na levé straně nastavena v komunikaci jaký port je připojen, rychlost komunikace, parita, datové bity a obě čísla hlavičky komunikace.



Obr.29.:Suport – ERROR 3

Co se týká samotné simulace, tak dole je 10 LED diod indikujících polohu suportu. LED diody označené K1, K2, K3 a K4 mají ještě na starost simulovat čidla polohy. LED dioda EM1 indikuje posun suportu rychloposuvem doprava, EM2 zase posun suportu rychloposuvem doleva. LED dioda EM3 indikuje pomalý posuv suportu. Vpravo jsou vidět zmiňované tři červené LED diody, které indikují chybové stavy a hned vedle tlačítko RESET, který má na starost nastavit suport do výchozí polohy. Zbytek grafiky na panelu slouží jako dekorace. Není funkční.



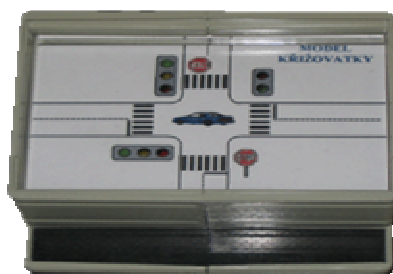
Obr.30.:Suport-LabVIEW

4.3.Křižovatka

Simulace modelu Křižovatka je úplně nejjednodušší. O celé řízení se stará automat, model nesimuluje žádné pohyby, pouze rozsvěcuje LED diody. Není zde žádná zpětná vazba. Jsou připojeny pouze výstupy automatu na vstup modelu.

Funkce modelu:

- Křižovatka je pasivní modul (neobsahuje procesorovou jednotku), který zobrazuje pomocí LED diod stavy výstupních signálů řídicího automatu.
- Pomocí přídavných vstupních jednotek (např. modul spínačů a tlačítek systému EDUtec) je možné modifikovat běh programu, realizovat tlačítko “chodci” atd.[14]



Obr.31.:Křižovatka[14]

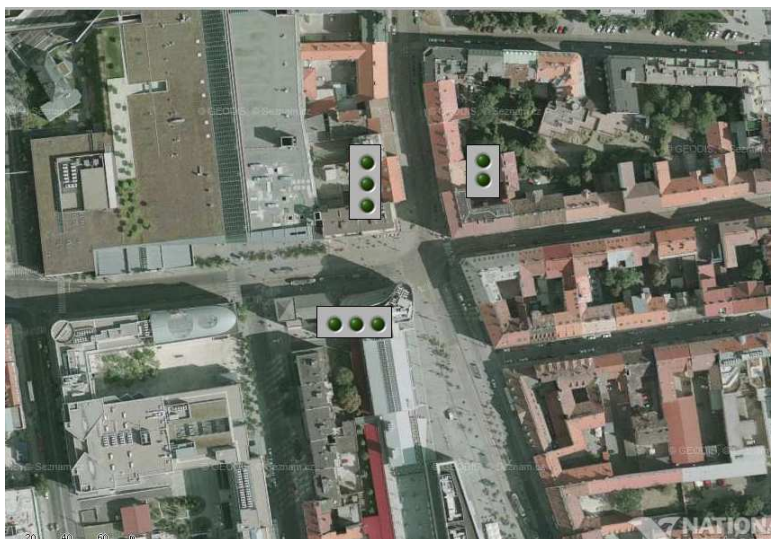
Křižovatka se skládá ze tří semaforů. Dva pro auta a jeden pro chodce (viz. obr.31.). Vzhledem k tomu, že jsou v modulu použity pouze vstupy je programování v LabVIEW velice jednoduché.



Obr.32.:Blokový diagram

Jediné co v LabVIEW musí být nastaveno je propojení patřičných vstupů LabVIEW s výstupy z PLC. Tedy program obsahuje sériovou komunikaci, která je popsána výše a k jejím vstupům jsou jen připojeny LED diody viz. obr.32.

Pak na přední desce vypadá zobrazení programu jako na obr.33. Jako pozadí je použita jedna z Brněnských křižovatek a v ní jsou zobrazeny semaforey přesně tak jako u modelu EDU-mod.



Obr.33.:Křižovatka

4.4.Mísící jednotka

Dalším modelem je Mísící jednotka. Patří ke složitějším modelům. Docela složité bylo navrhování podmínek ke správnému fungování simulace.

Funkce modelu:

- Mísící jednotka je aktivní modul s vlastní inteligencí simulující funkci technologie složené ze tří plnicích tanků a mísící nádoby
- Jednotka je řízena šesti výstupy (5 solenoid. ventilů, mixér), vnitřní procesorová jednotka ovládá LED simulující snímače výšky hladiny a generuje chybová hlášení.
- Po sepnutí ventilů SV1 až SV3 se začnou plnit příslušné tanky s objemem 84 litrů rychlostí 6 l/s. Hladinoměry H1 až H8 snímající výšku hladiny v jednotlivých nádobách mají následující význam:
dolní snímače (H3, H5, H8) - minimální množství kapaliny (cca 10l)
střední snímače (H2, H7) - polovina nádrže
horní snímače (H1, H4, H6) - plná nádrž.

- Mísicí nádoba má objem 253 litrů, průtok napouštěcím a vypouštěcím potrubím (přes SV4, SV5) je 18 l/s.

Inicializační stav:

- Po zapnutí napájení nebo po restartu (tlačítko RESET) se jednotka automaticky nastaví do inicializačního stavu - všechny nádoby prázdné. Zároveň se rozblíká červená LED dioda ERR, která zhasne po prvním vybuzení některého ze solenoidových ventilů.

Chybová hlášení:

- Při přetečení kterékoli nádoby (včetně mísicí) se vyhodnotí chyba, která je signalizována rozsvícením LED diody ERR. Systém se vrátí do výchozího stavu po stisku tlačítka RESET. [14]



Obr.34.: Mísicí jednotka [14]

Mísicí jednotka má na starost simulovat míšení třech různých kapalin v jednu. Program v automatu napustí jednotlivé tanky dle stanovených úrovní. LED diody v tancích slouží jako čidla hladiny. Jakmile je dostatečné množství kapalin v tancích, tak se uzavře přívod (ventily) a začne se z tanků napouštět do nádoby na míchání kapalina, při zapnutém smíchávání kapalin. Tomu napomáhá mixér, vrtule. Mísí se určitou dobu, podle toho jak je nastaven v automatu časovač a poté se začne vypouštět. Vzhledem k tomu, že v mísicí nádobě není žádné čidlo, tak se musí také vypouštět pomocí nějakého časovače. Tedy co se týká dob napouštění a vypouštění, tak musí být model a simulace téměř totožná.

Proto je důležité, aby bylo spočítáno jak dlouho trvá napouštění a vypouštění nádob:

$$V_1 = 84l$$

$$V_2 = 253l$$

$$Q = 18l/s$$

$$t_1 = \frac{V_1}{Q} = \frac{84}{18} = 4,7s$$

$$t_2 = \frac{V_2}{Q} = \frac{253}{18} = 14,05s$$

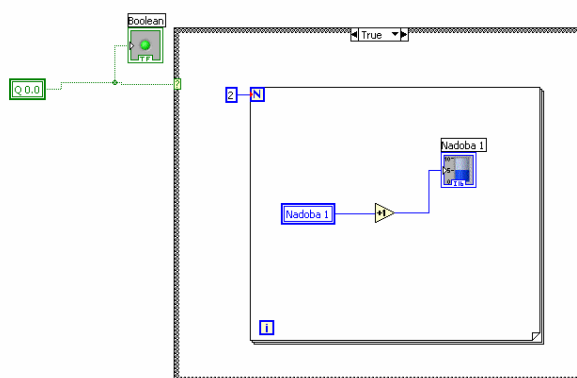
$$t_d = \frac{t_1}{84} = \frac{4700}{84} = \frac{[ms]}{[-]} = 55,95ms \quad (11)$$

$$N = 2$$

$$t_t = N \cdot t_d = 2 \cdot 55,95 = 111,9ms$$

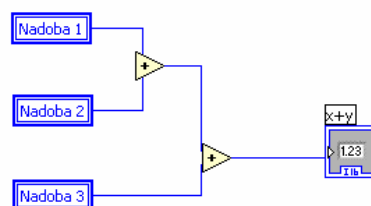
Ze vzorce 11, V_1 je objem malých tanků, V_2 je objem nádoby na smíchání kapalin a Q je průtok potrubím. Po výpočtech dle vzorce 11 vychází, že čas výtoku malé nádoby je

$t_1=4,7\text{s}$ a čas výtoku nádoby na smíchání kapalin je $t_2=14,05\text{s}$. Pomocí těchto časů je nutné vypočítat jak má být nastaven časovač. Tedy t_d je teoretická doba čekání časovače. Převedeme rychlost výtoku t_1 ze sekund na milisekundy a podělíme tento čas počtem dílků tanku v programu. Jelikož se jeden dílek rovná jednomu litru, tak se počet dílků rovná počtu litrů V_1 . Tedy podělíme čas t_1 počtem dílků. Výsledek časové prodlevy časovače je dost malá hodnota a vzhledem k tomu, že časovač se bude starat i o komunikaci, tak by se s takhle malou hodnotou mohlo stát, že by komunikace neprobíhala korektně. Z toho důvodu je posléze v programu u všech tanků dáno $N = 2$ a tím pádem můžeme časovou prodlevu 2x zvýšit. Co se týká hlavní nádoby na míchání, tak v tomto případě bylo jen změřeno, jak rychle probíhá vypouštění a podle toho se upravilo N v cyklu For u tohoto vypouštění. Program tedy nakonec vykazuje časy $t_1 = 4,9\text{s}$ a $t_2 = 14,58\text{s}$. Neodpovídají přesně časům modelu EDU-mod, ale přesnost je dostatečná k řízení.



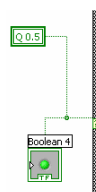
Obr.35:Naplnění Nádoby 1

Při naplnění nádob kapalinou (obr.35.) se využívají stejné struktury jako u suportu. Tím je myšleno, že automat sepne logickou jedničku, která se promítne v LabVIEW na Q0.0. Pokud je na tomhle vstupu Q0.0 v LabVIEW logická jednička, tak se rozsvítí LED dioda napouštění k dané nádobě a nastaví se struktura CASE do True. Tím pádem je povoleno, aby se spustil cyklus For. V tomto případě se cyklus opakuje vždy 2x a až poté promítne výsledek. Je to z důvodu urychlení napouštění, aby časovač nemusel být nastaven na moc velkou rychlost. Mohlo by se pak stát, že by byl problém s odesíláním a přijímáním pokynů z komunikační jednotky. Vlastně je tu využívána nová součást LabVIEW, která lze nalézt v nabídce *Control =>Modern =>Numeric =>Tank*. Součást se tedy jmenuje *Tank*. Připomíná napouštění nějaké nádoby. Napouštění se dá nastavit, aby graficky zobrazovalo přítok kapaliny ze spodu nahoru, ze shora dolů, mezi úrovněmi nebo prostě jen pohybuující se ryska. Stejně tak Tank pojme různé množství proměnných. V tomto případě se jedná o 16-ti bitový integer. Další důležitá položka v nastavení Tanku, která musí být nastavena, je rozmezí odkud pokud má kapalina stoupat, v případě mísící jednotky se jedná o velikost nádoby. Spodní limit je tedy nastaven na 0 litrů a horní limit na 84 litrů. V cyklu For je tedy programová část vyřešena tak, že z Nádoby 1 byla vytvořena lokální proměnná Nádoba 1. Lokální proměnná je vždy přednastavena jako hodnota, do které se zapisují data. V tomto případě se musí přenastavit na proměnnou, ze které se budou načítat data. To se provede tak, že je poklikáno na proměnnou pravým tlačítkem myši a zvolí se možnost *Change to read*. K této lokální proměnné je přidán blok +1, který bude přičítat jedničku k hodnotě. Hodnota je v proměnné uložena a zapsána do součásti Nádoba 1. Tyto menší tanky jsou v programu tři a tedy i tři tyto grafické struktury.



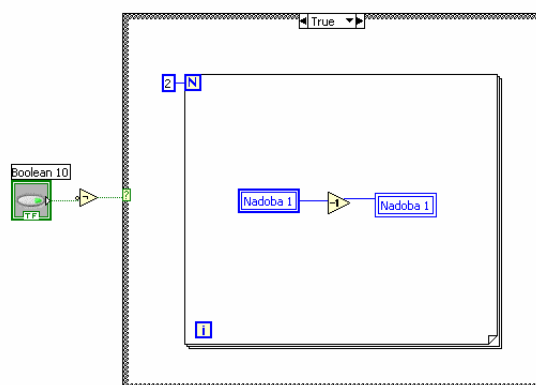
Obr.36.: Množství kapaliny

Tak jsou získány tři proměnné (Nádoba 1, Nádoba 2 a Nádoba 3 viz. Obr.36.). Tyto proměnné jsou naplněny různými hodnotami, podle toho kolik je v nádobách kapaliny. Aby mohlo být simulováno napouštění z tanků, tak je nejdříve nutno tyto hodnoty všechny posčítat a uložit do další proměnné (viz. Obr.36.) Hodnota v nově vzniklém indikátoru $x+y$ je množství kapaliny v nádobě na míchání. Lokální proměnné Nádoby 1 až 3 musí být opět nastaveny tak, aby se z nich dala číst data. To znamená stejně jako v příkladu výše.



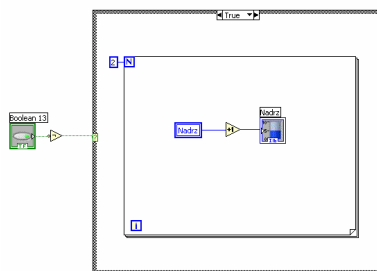
Obr.37.: Napouštění

Pokud jsou napuštěny všechny tanky do patřičné výšky, tak je zapnuto vypouštění kapaliny z tanků a tato kapalina musí být napuštěna do nádoby na mixování. Tedy je přivedena logická jednička do struktury CASE a tím pádem se rozsvítí LED dioda, která signalizuje otevření ventilu mezi tanky a nádobou na mixování (viz. obr. 37.).



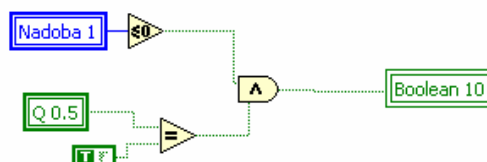
Obr.38.: Vypouštění nádob

V předchozím kroku se spustila struktura, ve které jsou čtyři podstruktury. První tři se týkají vypouštění z tanků. Na obr.33. je znázorněno jak vypadá program pro vypouštění. Pokud není tlačítko Boolean 10 zapnuto, tak platí stav True, protože mezi tlačítkem a strukturou je blok negace. Uvnitř struktury je cyklus For, který se opakuje 2x protože $N=2$. Je to z důvodu, aby vypouštění trvalo maximálně 5s. Uvnitř cyklu je opět proměnná Nádoba 1, jen s tím rozdílem, že teď je z ní odečítána jednička, na místo toho jak v předchozím příkladě byla jednička přičítána. Podobně vypadají i další dvě podstruktury.



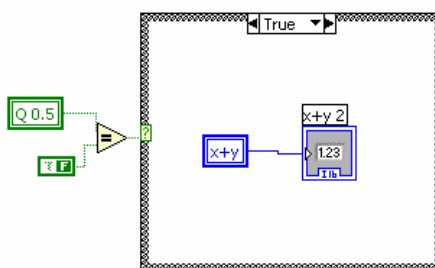
Obr.39.: Napouštění nádoby na mixování

Při napouštění nádrže je používán stejný postup, jako při napouštění malých tanků. Jen struktura byla inicializována přes negaci Boolean 13 (viz. obr.39.). Inicializování přes negované tlačítko patří k podmínkám, které jsou definovány dále v textu.



Obr.40.: Podmínka vypuštění nádoby

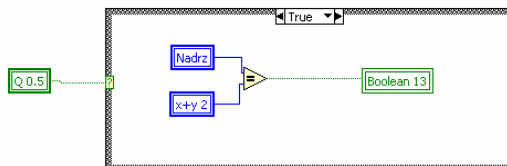
První podmínkou je, že pokud se nádoba vyprázdní, program má tu funkci, že by dále odečítal hodnotu a buď by nádrž nabývala záporné proměnné, nebo by přetekla a měla proměnnou nejvyšší možnou, jakou povolí datový typ. Podmínka je tedy vyřešena tak, že pokud je Q0.5 výstup z PLC roven logické jedničce a zároveň hodnota v Nádobě je menší nebo rovna nule, tak tehdy a jen tehdy se načte logická jednička do Boolean 10 (viz. Obr.40.). Pokud je Boolean 10 nastaven na logickou jedničku, tak podle obr.38. se logická jednička neguje na logickou nulu a z tohoto důvodu se přepne struktura CASE do stavu False a dále se neodečítá jednička od proměnné Nádob 1. Podobná podmínka platí i pro zbylé nádoby.



Obr.41.: Podmínka pro napuštění mixéru

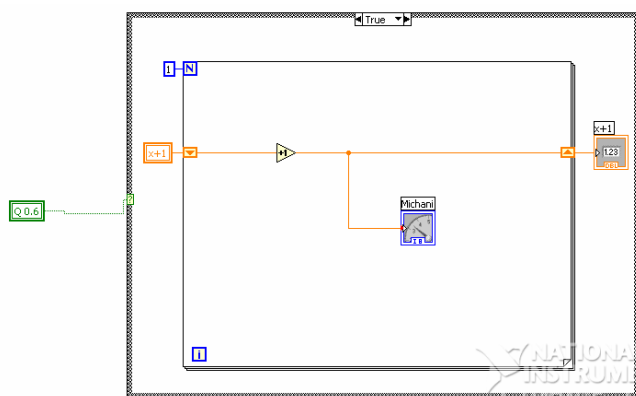
Další podmínkou, která musí být vytvořena je předání hodnoty z lokální proměnné $x+y$ do $x+y 2$ a to z důvodu, že jestliže je naplněna pouze hodnota $x + y$ viz. obr.41., tak pokud by bylo spuštěno vypouštění, hodnota v proměnné $x + y$ by se měnila podle toho, jak by se vypouštěly jednotlivé tanky. Což je nežádoucí. Je potřebné, aby hodnota v proměnné zůstala stále na maximální hranici. Programově je tento problém vyřešen tak, že se nechá ve struktuře předat hodnotu z proměnné $x+y$ do proměnné $x+y 2$ a tato

struktura je spuštěna tehdy a jen tehdy, pokud je Q0.5 roven logické nule. Jakmile se zapne vypouštění tanků, tak struktura je vypnutá a nepředává hodnotu a tím pádem zůstává v Indikátoru $x + y$ 2 hodnota maximální.



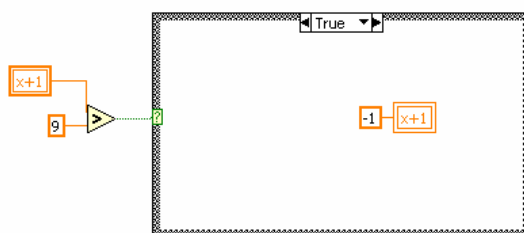
Obr.42.: Podmínka pro napuštění mixéru 2

Tahle podmínka navazuje na podmínku na obr.42. Pokud je Q0.5 tedy naplněna logickou jedničkou, tak se vykoná struktura CASE. Q0.5 je tedy lokální proměnná a přepnutá pomocí *change to read* na čtení hodnoty z této proměnné. Potom tedy pokud se hodnota v lokální proměnné Nádrž rovná lokální proměnné $x+y$ 2 viz. výše, tak se přiřadí logická jednička do lokální proměnné Boolean 13. Na tomto příkladě jsou znázorněny rozdíly mezi lokálními proměnnými, které jsou nastaveny pro čtení (Boolean 13) a lokálními proměnnými nastavenými pro zápis (Nádrž, Q0.5). Z grafického hlediska mají lokální proměnné nastavené pro zápis tenčí rámeček než lokální proměnné nastavené pro čtení.



Obr.43.: Míchání

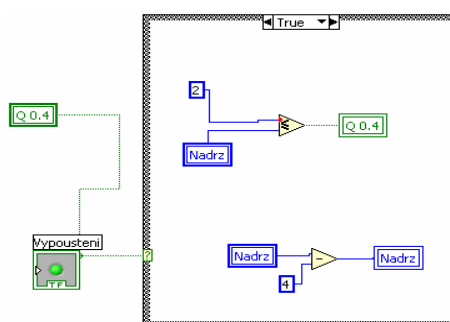
Pro míchání (obr.43.) musí být nastavena hodnota „logická jednička“ v lokální proměnné Q0.6. Pak je spuštěna struktura Case. V ní proběhne cyklus For. N je nastaveno na jedničku, tedy načítání do proměnných probíhá po každém proběhnutí cyklu. K proměnné $x+1$ se přičítá jednička a hodnota se promítá do Míchání. Musí zde být poskytnuta podmínka, aby Míchání běhalo pořád dokola a ne, aby hodnota vyšla z rozsahu ručičky.



Obr.44.: Míchání-podmínka

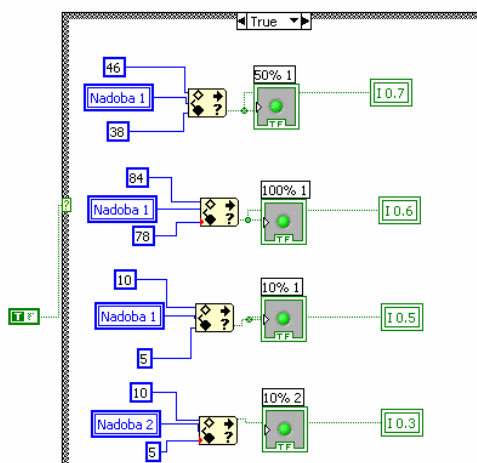
Podmínka je tedy taková, že pokud je v proměnné hodnota větší než 9, tak se naplní struktura CASE True a zapíše do proměnné hodnotu -1. Rozsah ciferníku je 0 až 10 a jakmile je $x+1$ rovno 10-ti, tak se nastaví do proměnné ona -1 a v druhém kroku už je v proměnné 0, takže ručička plynule přejde z 10-ti do nuly.

Při vypouštění musí být Q0.4 naplněna proměnnou logická jedna. Pak se rozsvítí LED dioda Vypouštění a struktura CASE se přepne do režimu True. V nádrži se odečítá od lokální proměnné, určené pro čtení, konstanta čtyři přes blok LabVIEW minus a zároveň je tu i podmínka, že pokud jsou v nádrži méně než 2 litry vody, pak se nastaví Q0.4 do logické nuly. Jednoduše zde bude negována logická jednička. Veškerý popis v tomto odstavci byl směřován k obr.45.



Obr.45.: Vypouštění

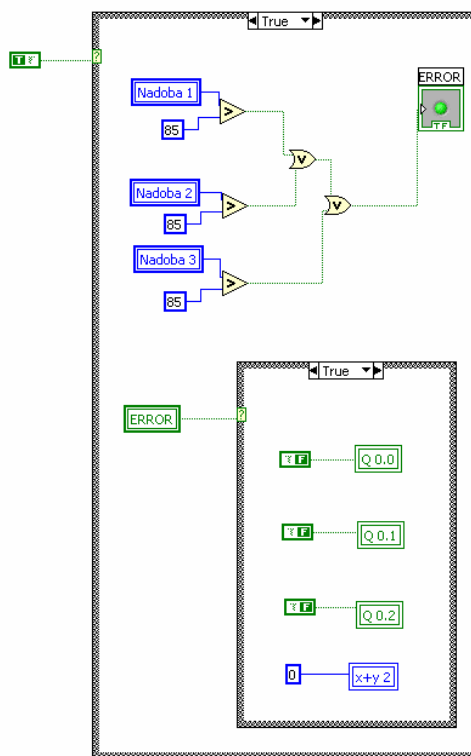
Nastavení čidel u mísící jednotky je podobné jako nastavení čidel u suportu. Struktura je neustále nastavena konstantou True na logickou jedničku, tedy na True. A uvnitř platí, že pokud se kapalina v nádobě nachází mezi hodnotami 46 a 38, tak se rozsvítí čidlo na 50% naplnění tanku a signál je odeslán na vstup PLC IO.7. Dále pokud se kapalina v nádobě nachází mezi hodnotami 84 a 78, tak je nádoba plná (tedy 100% kapaliny v nádobě) a rozsvítí se LED dioda čidla. Signál je odeslán na vstup PLC IO.6. Čidel je dohromady osm a všechny pracují velice podobně jako jsou popsány první dva příklady. Tím dostává automat informace o stavu množství kapaliny v jednotlivých tancích.



Obr.46.: Čidla

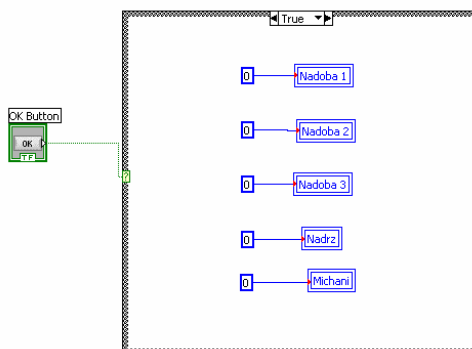
Čidlo ERROR se rozsvítí tehdy, pokud některá z nádrží přeteče viz. obr.47. Struktura CASE je naplněna konstantou True a tedy neustále přepnutá ve stavu True.

Pokud v Nádobě 1 je větší hodnota proměnné než 85 nebo v Nádobě 2 je větší hodnota proměnné než 85 nebo v Nádobě 3 je větší hodnota proměnné než 85, pak se rozsvítí ERROR. Jestliže se rozsvítí ERROR, tak se přepne druhá struktura CASE ze stavu False do True. A v tom případě se naplní lokální proměnné Q0.0, Q0.1 a Q0.2 logickou nulou a lokální proměnná $x+y$ 2 se naplní nulou datového typu integer. To znamená, že přestane stoupat hladina v jednotlivých tancích a ani nebude dovoleno tyto tanky napustit do nádoby na míchání.



Obr.47.: Čidlo ERROR

Po stisknutí tlačítka RESET jsou naplněny lokální proměnné Nádoba 1, Nádoba 2, Nádoba 3, Nádrž a Míchání hodnotou nula. To znamená, že po stisknutí se vyprázdní všechny nádoby (tedy všechny hodnoty v proměnných se nastaví na 0) a vrtule míchání se nastaví do výchozí polohy. (viz. obr.48.)



Obr.48.: RESET

Přední panel je zobrazen na obr.49. V horní části jsou tři menší nádrže, které pojmu 83 litrů (mají tedy 83 políček). Kapaliny v každé z nádrží je zobrazena jinou barvou. LED diody nad nádržemi indikují, zdali jsou otevřeny ventily, nebo jsou naopak uzavřeny. LED diody vedle nádrží indikují jaké množství kapaliny se nachází v nádržích. Vzhledem k tomu, že připojení simulačních modelů počítá s PLC s osmi vstupy a osmi výstupy, tak jedna z nádrží má pouze měření minimální hladiny a maximální hladiny. Při otevření ventilu nad nádrží k míchání se rozsvítí LED dioda nad touto nádrží a začnou se vypouštět všechny tři menší nádoby a začne se napouštět nádoba na míchání. Při zapnutí míchání nám jej bude signalizovat otáčení rafiky míchání. Rafika se otáčí po směru hodinových ručiček stejnou rychlostí kroku, jakým probíhá napuštění nebo vypuštění jednoho litru kapaliny. Při vypuštění se vypustí zbytek kapaliny. O otevření a zavření ventilu se stará čistě PLC a to tím způsobem, že otevře ventil a pomocí časovače nechá ventil otevřen určitou dobu. Na panelu už zbývá jen LED dioda indikující nějaký problém, ERROR a tlačítko RESET.



Obr.49.: Mísící jednotka v LabVIEW

4.5.Pračka

Posledním modelem Edu-mod je Pračka. Pračka je nejsložitější model. Simuluje základní funkce praní automatické pračky špinavého prádla.

Funkce modelu:

- Model je řízen 6 binárními výstupy PLC (akčních členů) na úrovni 24V logiky (společná GND), jejichž stav je na modelu zobrazován pomocí LED.
- Dva výstupy slouží pro otáčení bubnem. Jeho pohyb je znázorněn na osmi kruhově uspořádaných LED formou "běžícího světla". Rychlost otáčení bubnu (praní nebo ždímání) se řídí výstupem OTÁČKY (0=praní, 1=ždímání).

- Další funkcí modelu je simulace napouštění a vypouštění vody do prací vany a její ohřev (včetně chladnutí). Pro napouštění slouží bit VODA, pro vypouštění bit ČERPADLO, pro ohřev bit TOPENÍ.
- Hladina vody v prací vaně je snímána ve dvou úrovních (50% a 100%) a zobrazována na LED. Informace je samozřejmě také posílána na výstupy modelu (vstupy PLC).
- Při ohřívání vody se model chová jako soustava 2. řádu, avšak časové konstanty jsou zkráceny tak, aby se při ladění aplikací nemuselo příliš dlouho čekat (ohřev plné vany na 90°C trvá cca 60s). Teplota vody je snímána ve 4 bodech (30, 40, 60 a 90°C).

Inicializační stav:

- Po zapnutí napájení nebo po restartu (tlačítko RESET) se jednotka automaticky nastaví do inicializačního stavu - prázdný buben, počáteční teplota.

Chybová hlášení:

- Model generuje dva druhy chybových hlášení (opravitelná a neopravitelná chyba).
- Opravitelná chyba
Nastává pouze tehdy, přijde-li současně povel TOČ BUBNEM VLEVO a TOČ BUBNEM VPRAVO. V tom případě začne blikat červená LED ERR a buben se přestane otáčet. Po odstranění kolizního stavu buben pracuje normálně.
- Neopravitelná chyba vzniká ve dvou případech:
 - Prací vana přeteče
 - Teplota vody stoupne nad 90°C

Neopravitelná chyba je indikována rozsvícením červené LED ERR. Z tohoto stavu se lze dostat pouze stiskem tlačítka RESET.[14]

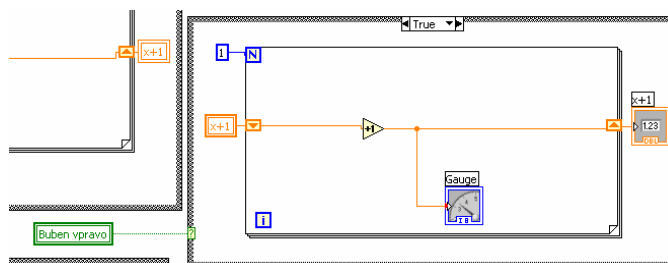


Obr.50.: Pračka[14]

Pračka je model který má na starost simulovat praní prádla (viz. obr.50.), jak je napsáno výše. Ve funkci modelu je napsáno, že ohřívání na 90°C trvá cca 60s. U simulace je tento čas kratší a trvá pouhých cca 12s. Programově se tento čas dá samozřejmě přenastavit na odpovídajících 60s. Je to dáno časovačem, který je použit. Nicméně v tomto případě čas simulace nehraje roli, protože PLC neřídí model pomocí časovače, ale pomocí indikace čidel. Tak tohle je jeden z mála rozdílů mezi simulací a modelem. Dalším menším rozdílem mezi modelem a simulací je ten, že u modelu teplota začne klesat pokud:

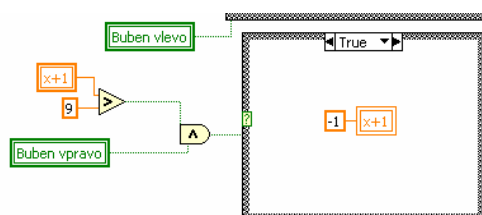
- napouštíme další vodu, tedy hladina stoupá
- vypouštíme vodu

U simulace začne teplota klesat pouze pokud vypouštíme vodu. Ovšem na úlohy, které jsou pro pračku připraveny nemá tahle změna vliv.



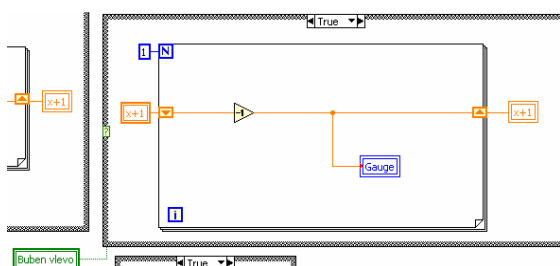
Obr.51.: Otáčení bubnu pračky doprava

Na obr.51. je naprogramování otáčení bubnu doprava. Jestliže lokální proměnná „Buben vpravo“ má hodnotu logické jedničky, pak se provedou příkazy uvnitř struktury CASE. Uvnitř je cyklus FOR, kde $N = 1$, tedy zápis do indikátoru $x+1$ se provádí po každém proběhnutí cyklu. Cyklus je vytvořen tak, že na začátku je lokální proměnná $x+1$, do které se přičítá jednička a tahle hodnota je zapsána do „Gauze“ což je indikátor stejný jako v případě míchání u míchací jednotky. Po proběhnutí se zapíše do indikátoru $x+1$ hodnota o jedničku větší a cyklus se rozběhne znovu.



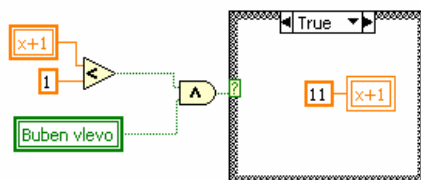
Obr.52.: Podmínka pro buben vpravo

Z programu je vidět, že hodnota v tomto případě by nabývala hodnot o jedničku větších až do nekonečna. Proto je potřeba vytvořit podmínku, která by vrátila hodnotu zpět na začátek. Tahle podmínka je na obr.52. Podmínka spuštění struktury CASE je taková, že musí být lokální proměnná „Buben vpravo“ nastavená na logickou jedničku a zároveň (pomocí AND) musí být hodnota v lokální proměnné „ $x+1$ “ větší než devět. Potom je okno aktivní a vykonají se funkce, které jsou v tomto okně. Momentálně je do lokální proměnné „ $x+1$ “ naplněna hodnota -1 a tím pádem tedy se vrátí ručička „Gauze“ na počáteční hodnotu 0. Ve finále to vypadá tak, že se ručička otáčí plynule po směru hodinových ručiček.



Obr.53.: Otáčení bubnu pračky doleva

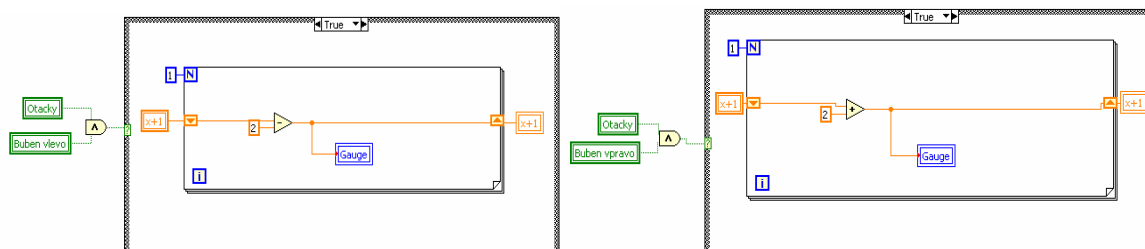
V případě otáčení bubnu vlevo je postup úplně stejný jen s tím rozdílem, že je jednička odečítána od proměnné a v podmínce tedy pokud je proměnná „ $x+1$ “ menší než jedna a zároveň v lokální proměnné „Buben vlevo“ je logická jednička, tak se hodnota v lokální proměnné „ $x+1$ “ naplní číslem jedenáct.



Obr.54.: Podmínka pro buben vlevo

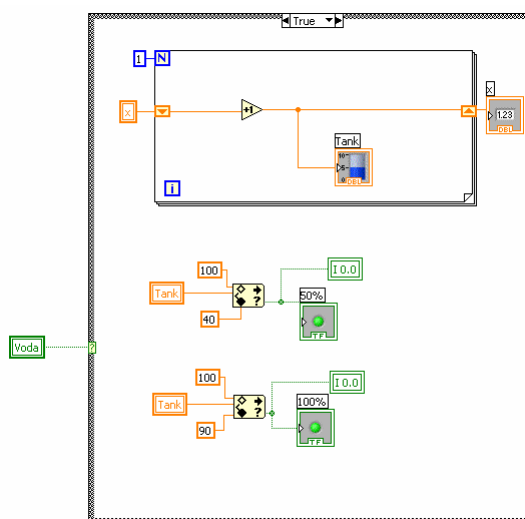
Při ždímání musí být zvýšeny otáčky bubnu. Pro tento případ jsou v programu naprogramované bloky viz. obr.55. Musí být splněno, že jsou v proměnné naplněny hodnotou logická jedna. Jedině v tom případě se splní funkce naprogramované uvnitř struktur. Pokud je lokální jednička v proměnných „Otáčky“ a „Buben vlevo“, pak se vykoná cyklus FOR, kdy se od proměnné „ $x+1$ “ bude odečítat dvojka a zobrazovat v „Gauče“. Pokud bude logická jednička v proměnných „Otáčky“ a „Buben vpravo“, bude se tedy vykonávat cyklus FOR, ve kterém se bude dvojka k proměnné „ $x+1$ “ přičítat.

Proč zrovna dvojka? Je to z důvodu, že se tím pádem bude opticky rafika pohybovat 2x rychleji. Vlastně bude přeskakovat vždy o jednu hodnotu, na kterou se nikdy nedostane. Dalo by se to vyřešit i jiným způsobem. Například tak, že by byla přičítána i odečítána jednička jako při praní, ale v cyklu by N bylo naplněno místo jedničkou, tak dvojkou. Tím pádem by se výsledná rafika pohybovala úplně stejně.



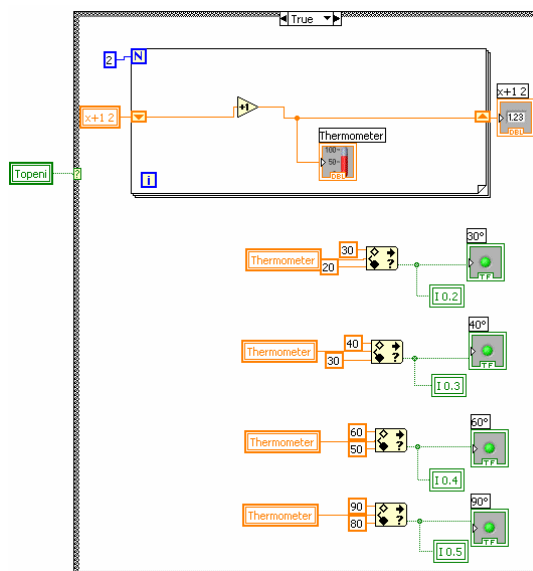
Obr.55.: Otáčky Bubnu, vlevo i vpravo

Další nezbytnou součástí pračky je napouštění vody. Naprogramování v jazyce G je na obr. 56. Jedná se o to, že jeli naplněna proměnná „voda“ logickou jedničkou, tak se vykonají příkazy ve struktuře CASE. V první řadě je to cyklus FOR, kdy se opět do tanku musí připočítávat hodnota jedna, aby hladina stoupala. Tentokrát se naplňuje proměnná „ x “. Jako další je na řadě indikace množství vody v pračce. Tedy pokud je hodnota v „tanku“ mezi čtyřicátkou a stovkou, pak svítí led dioda, která znázorňuje, že v pračce je minimálně polovina vody. Jestliže se už nachází hodnota mezi devadesátkou a stovkou, pak už je pračka plná. Pokud je tomu tak, svítí jak indikace 50% vody, tak 100% vody. Obě dvě LED diody. Stejně tomu je i u modelu EDU-mod. Rychlost napouštění vody se řídí podle časovače, který řídí sériovou linku. Nastavení časovače ve většině aplikací je mezi 100 až 150ms.



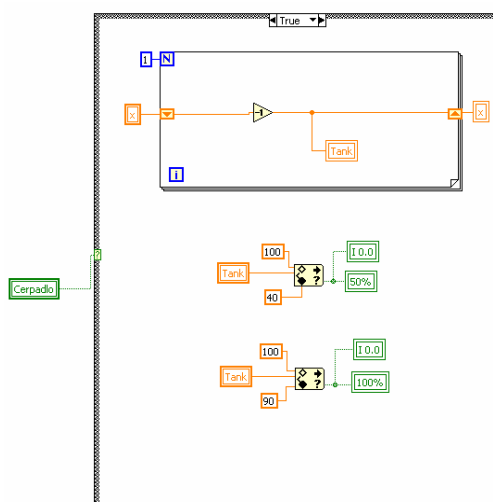
Obr.56.: Napouštění vody

Poté co je voda napuštěna do pračky, je potřeba ji vytopit na požadovanou teplotu. Jakmile je lokální proměnná „Topení“ naplněná hodnotou logická jedna, tak se vykonají příkazy ve struktuře. Opět se bude přičítat o jeden dílek v teploměru. Ale v tomto případě s $N = 2$, tak vlastně cyklus projede dvakrát, než zapíše hodnotu do proměnné „ $x+1 \ 2$ “ a „Thermometer“. Kromě signalizace pomocí integrovaného teploměru v LabVIEW se používá ještě signalizace pomocí LED diod. Takže jako příklad za všechny je hned první možnost na obr.57. Jestliže je hodnota v termometru mezi dvaceti a třiceti, tak se naplní logickou hodnotou jak proměnná 30°, tak proměnná IO.2. Tímto způsobem se naplní i zbylé tři LED diody.



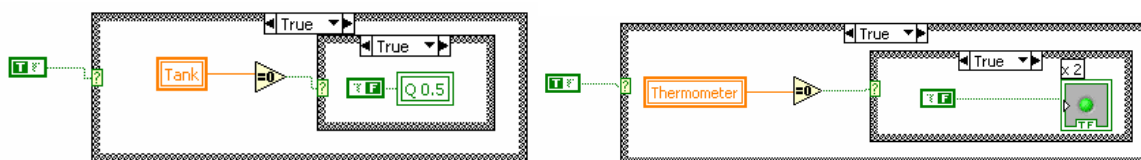
Obr.57.: Topení

Jakmile je vypráno, tak je potřeba vodu vypustit. Jedná se opět o analogii předchozích příkladů. Pokud je lokální proměnná „Čerpadlo“ naplněna logickou jedničkou, tak se vykonají příkazy. Začne se o jedničku snižovat hodnota v proměnné „ x “ a tedy hladina v pračce. Stejně tak budou zhasínat LED diody.(obr.58.)



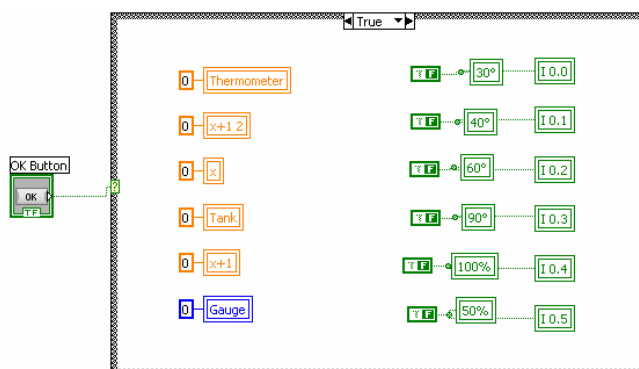
Obr.58.: Čerpadlo

Je potřeba opět stanovit podmínky, aby se do nádrže pračky, nebo teploměru nedostaly záporné hodnoty. Tedy jakmile bude nádrž prázdná, nebo teploměr bude na nulové hranici, tak už nesmí nabývat menší hodnotu. Na obr.59 je znázorněno, jak je tomu v LabVIEW docíleno. Pokud se proměnná „Tank“ rovná nule, pak je Q0.5 (ovládá napouštění) rovna logické nule a pokud je proměnná „Thermometer“ rovna nule, pak je proměnná „x 2“ naplněna logickou nulou. Díky těmto dvěma strukturám se nesníží hladiny v obou proměnných pod stanovené hranice.



Obr.59.: Podmínky pro vodu a teploměr

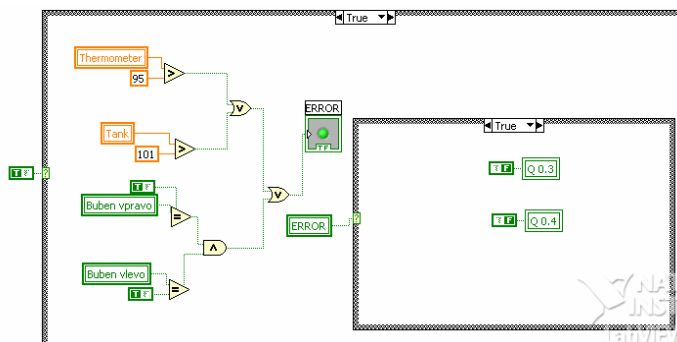
Po stisknutí tlačítka RESET je potřeba, aby pračka byla nastavena do výchozí polohy. Tedy nastavit všechny hodnoty u LED diod jako logické nuly a do veškerých proměnných nastavit nulovou hodnotu (viz. obr.60.).



Obr.60.: Tlačítko RESET

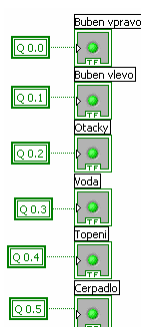
LED dioda ERROR se rozsvítí ve třech případech. Jestliže hodnota lokální proměnné „Tank“ bude větší než 101, jestliže hodnota lokální proměnné „Thermometer“

bude větší než 95 a nebo jestliže nastane stav, kdy jsou naplněny lokální proměnné „buben vpravo“ a „buben vlevo“ logickou jedničkou. Pak se rozsvítí LED dioda ERROR a nastaví se do lokálních proměnných „Q0.3“ a „Q0.4“ logická nula (viz. obr.61.).



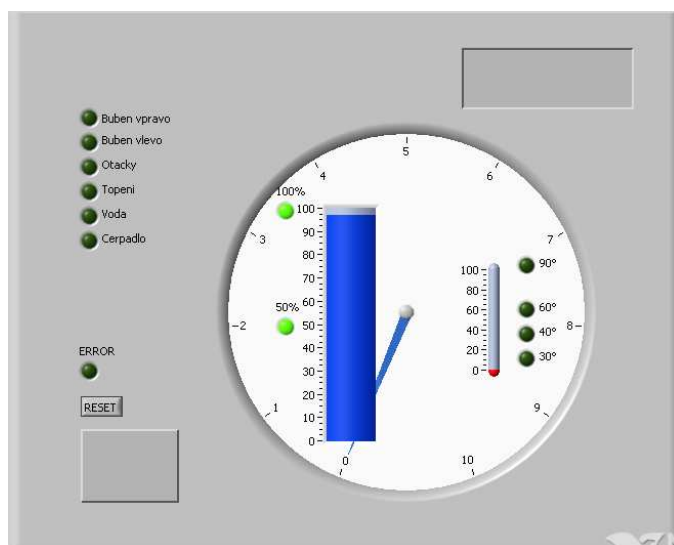
Obr.61.: ERROR

Ještě před programováním jednotlivých funkcí bylo v tomto případě lepší přiřadit k jednotlivým lokálním proměnným, které jsou určeny pro komunikaci (Q0.0, Q0.1, atd.), lokální proměnné, které budou signalizovat, co zrovna simulace provádí (Buben vpravo, Buben vlevo,atd.) (viz. obr.62.).



Obr.62.: Přiřazení proměnných

Na předním panelu LabVIEW potom vznikne pračka viz. obr. 63. Buben pračky simuluje rafika, která se otáčí doprava a doleva a dvěmi různými rychlostmi určenými buď pro praní, nebo ždímání. Je vidět, že na obr. 63., je napuštěna voda na maximální hladinu a v tento okamžik by mělo PLC začít topit. Co se zrovna děje v simulaci je vidět na LED diodách umístěných vlevo nahoře. Naopak vlevo dole je LED dioda ERROR, která se rozsvítí v případech, že nastala jedna ze tří chyb (viz. výše) a tlačítko RESET, aby se daly chyby anulovat a nastavit simulační model do výchozí polohy.



Obr.63.: Pračka - LabVIEW

5. ZÁVĚR

K splnění diplomové práce bylo nutné získat základní dovednosti se sestavováním elektronických obvodů, s programováním v jazyce C a programováním v NI LabVIEW. Díky těmto dovednostem je potom možné splnit všechny body zadání diplomové práce.

Počátečním úkolem bylo navrhnout modely Edu-MOD v simulačním prostředí NI LabVIEW. Z modelů Edu-MOD byly získány patřičné informace o jejich funkci a z těchto informací byl sestaven algoritmus v LabVIEW. Simulace v NI LabVIEW jsou sestaveny tak, aby co nejpřesněji napodobovaly modely Edu-MOD.

Aby mohl programovatelný automat řídit modely v osobním počítači byla navržena a vyrobena Komunikační jednotka. Jednalo se o návrh obvodů, naprogramování mikrořadiče a výroby celé jednotky na základě vlastní dokumentace. Komunikační jednotka pak převádí binární signály z PLC na sériovou linku do PC a naopak. Samozřejmě další možností by bylo použít PLC, které již umí komunikovat řízení po sériové lince. Výhoda komunikační jednotky je ovšem ta, že může být používán i PLC bez sériové linky a i když PLC má sériovou linku, ale program je vytvořen pro binární vstupy a výstupy, nemusí být program složitě přepracováván, ale jednoduše se připojí Komunikační jednotka.

Dosaženými cíly tedy byly, získaná komunikační jednotka, která se dá připojit k poměrně široké řadě automatů a simulační modely v NI LabVIEW. Program v PLC byl použit stejný jako pro moduly Edu-MOD. Tedy nakonec tedy PLC (LOGO!, Simatic S7 a Phoenix Contact) a řídí se přes komunikační jednotku simulační modely v PC. Po doladění programy jsou plně funkční a jednotka také.

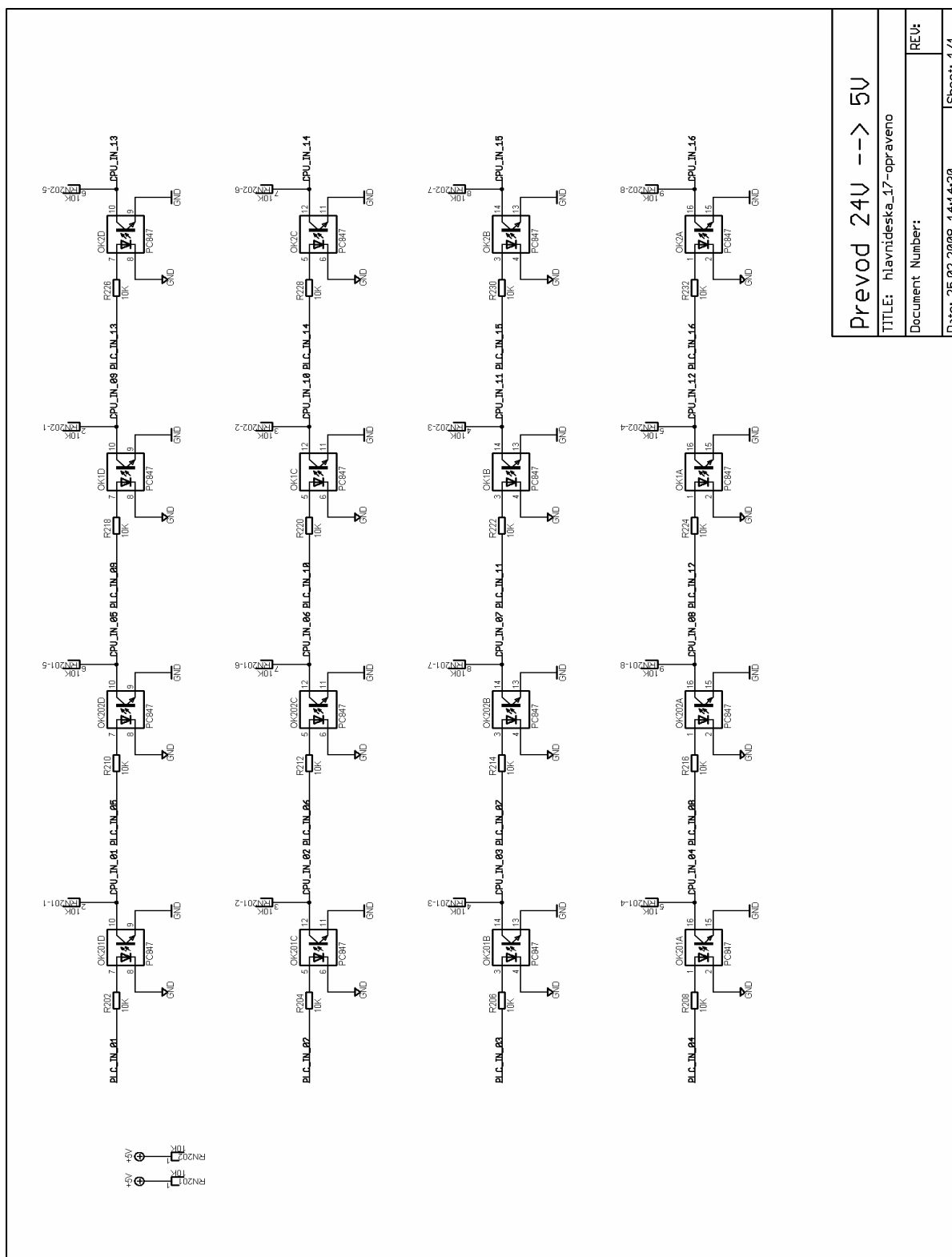
Výhodou tohoto řešení je značné ušetření finančních prostředků. Podobná Komunikační jednotka od jiných firem se cenově pohybuje v desítkách tisíc. Nicméně jednotka, která je vyrobena pro tento případ cenově vyjde na cca 2.000Kč. Samozřejmě v ceně jsou zahrnuty pouze součástky, tedy čas a práce strávená nad navrhováním a výrobou jednotky už v ceně zahrnuty nejsou. Další výhodou jsou úlohy, které mohou studenti využívat k rozvoji svých dovedností.

Bohužel k programům v LabVIEW je potřeba mít zakoupen software od společnosti National Instruments a dále mít alespoň základní znalost s prací v LabVIEW.

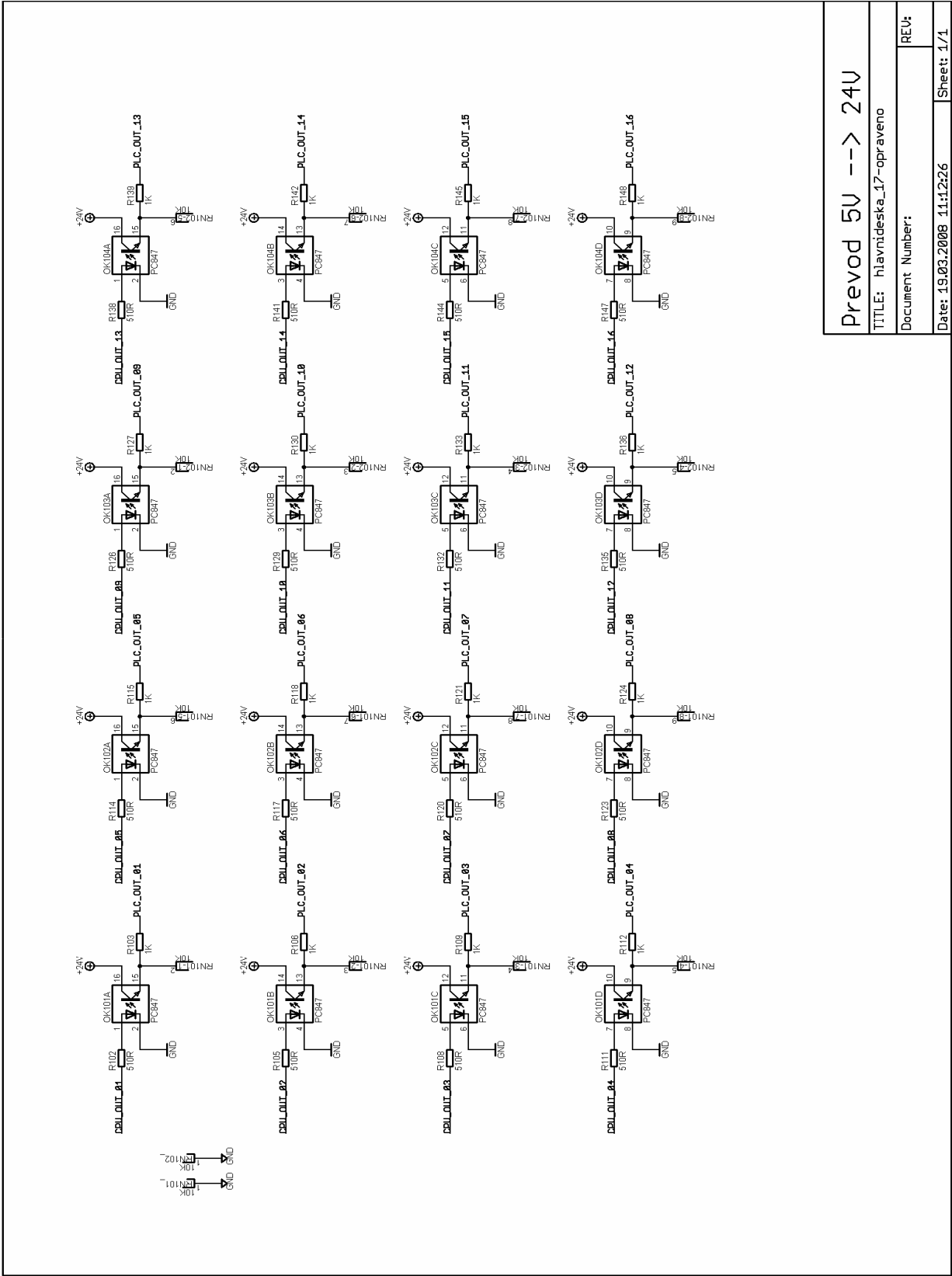
SEZNAM POUŽITÉ LITERATURY

- [1] ČERMÁK, M. *Moderní měřicí systémy*. Brno, 2003. 85 s., Diplomová práce na Přírodovědecké fakultě Masarykovy univerzity v Brně, Vedoucí diplomové práce RNDr. Zdeněk Bochníček, Dr.
- [2] KRÁL, J. *Základy programování jednočipových mikropočítačů v Assembleru a jazyku C*. 2006. 219 s., Skriptum SPŠE Rožnov pod Radhoštěm
- [3] VÁŇA, V. *ATMEL AVR Programování v jazyce C*. Praha, 2003. 216 s., Technická literatura BEN, ISBN 80-7300-102-0
- [4] WIKIPEDIA. *MultiSIM* [online]. 27.3.2008 [cit. 20.4.2008]. Dostupné z: <<http://en.wikipedia.org/wiki/MultiSIM>>.
- [5] KOČAJZ. *Výpočet odporového děliče napětí* [online]. 23.1.2007 [cit. 21.4.2008]. Dostupné z: <<http://kocajz.ic.cz/download/vypocet.html>>.
- [6] WIKIPEDIA. *TTL (logika)* [online]. 14.4.2008 [cit.22.4.2008]. Dostupné z: <http://cs.wikipedia.org/wiki/TTL_%28logika%29>.
- [7] SIEMENS s.r.o. *LOGO! Manuál* [online]. 8. vydání. 8.7.2005 [cit. 15.4.2008]. Dostupné z:<http://www1.siemens.cz/ad/current/content/data_files/automatizacni_systemy/mikrosystemy/logo/zakladni_pristroje/manual_logo_0ba5_2005_cz.pdf>.
- [8] SIEMENS s.r.o. *SIMATIC S7* [online]. 2006, 14.2.2007 [cit. 14.4.2008]. Dostupné z: <http://www1.siemens.cz/ad/current/index.php?nid=42&npt=doc_manualy&lnav=1&azletter=A Siemens S7>.
- [9] PHOENIX CONTACT. *ILC 150 starterkit* [online]. 15.4.2008 [cit. 15.4.2008]. Dostupné z: <<http://eshop.phoenixcontact.com/phoenix/treeViewClick.do;jsessionid=td0vHfnbQ1XdLvGm7hbxd94y2cJh0WBwdQBBg1MLVPYn9T1668kd!-1534911621!NONE?UID=2988955&parentUID=null&reloadFrame=true>>.
- [10] HORTEL, M. *Programování procesorů Atmel AVR* [online]. 28.12.2005 [cit. 25.4.2008]. Dostupné z: <<http://www.mlab.cz/Modules/AVR/ATmega801A/DOC/Programovani%20AVR.cs.pdf>>.
- [11] WIKIPEDIA. *Programovatelný logický automat* [online]. 2007, 3.5.2008 [cit. 7.5.2008]. Dostupné z: <http://cs.wikipedia.org/wiki/Programovateln%C3%BD_logick%C3%BD_automat>.
- [12] JANÍK, P. *Eagle – Návod* [online]. 9.4.2006, [cit. 9.4.2008]. Dostupné z: <http://paja-trb.unas.cz/elektronika/eagle/eagle_navod.html>.
- [13] PIRA. *Výroba desek plošných spojů* [online]. 31.9.2007 [cit. 2.5.2008]. Dostupné z: <<http://www.pira.cz/dps.htm>>.

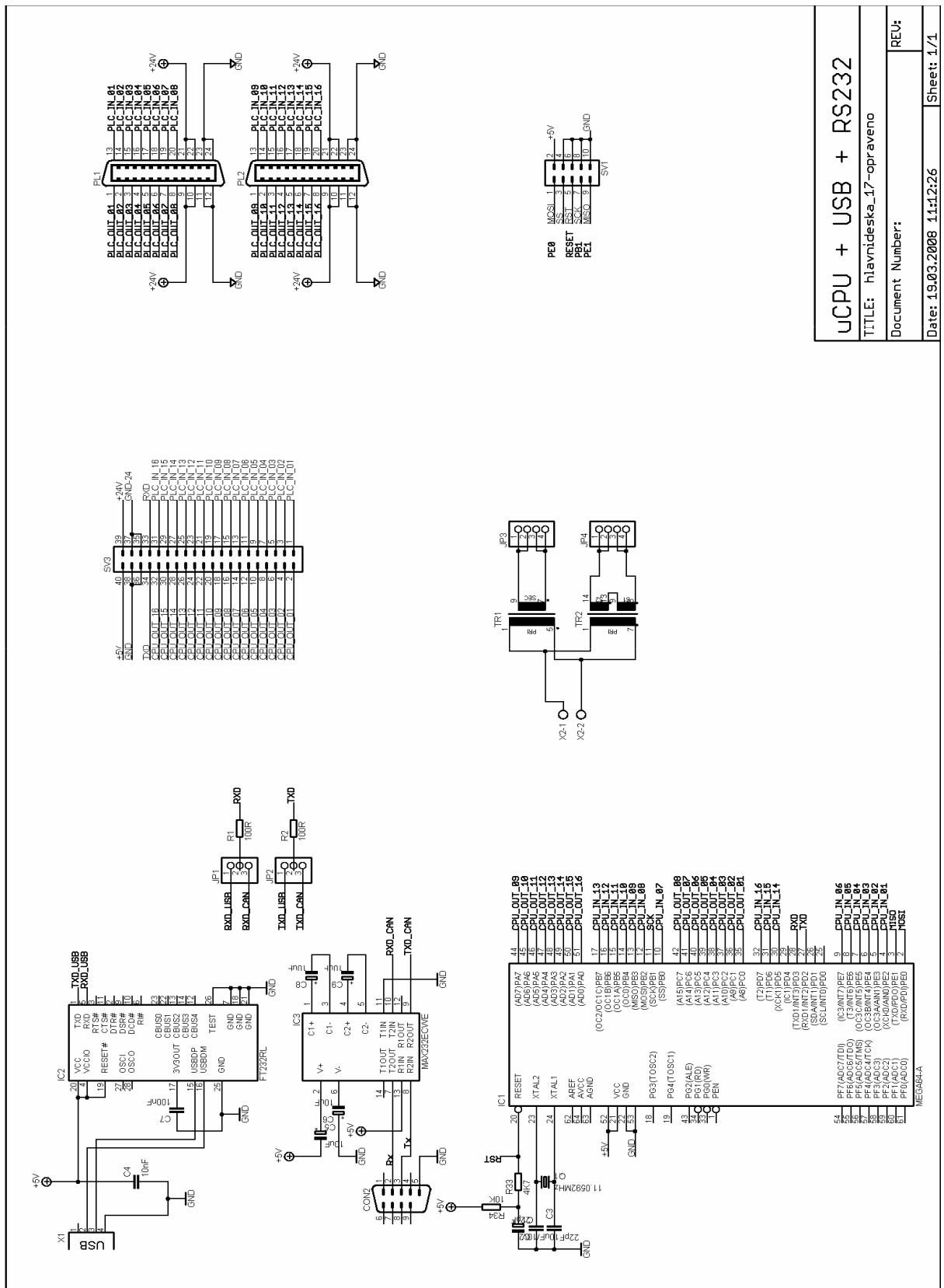
- [14] KOUHOUT, L. *EDUMAT* [online]. 2002, 10.5.2008 [cit. 11.5.2008]. Dostupné z: <www.edumat.cz>.



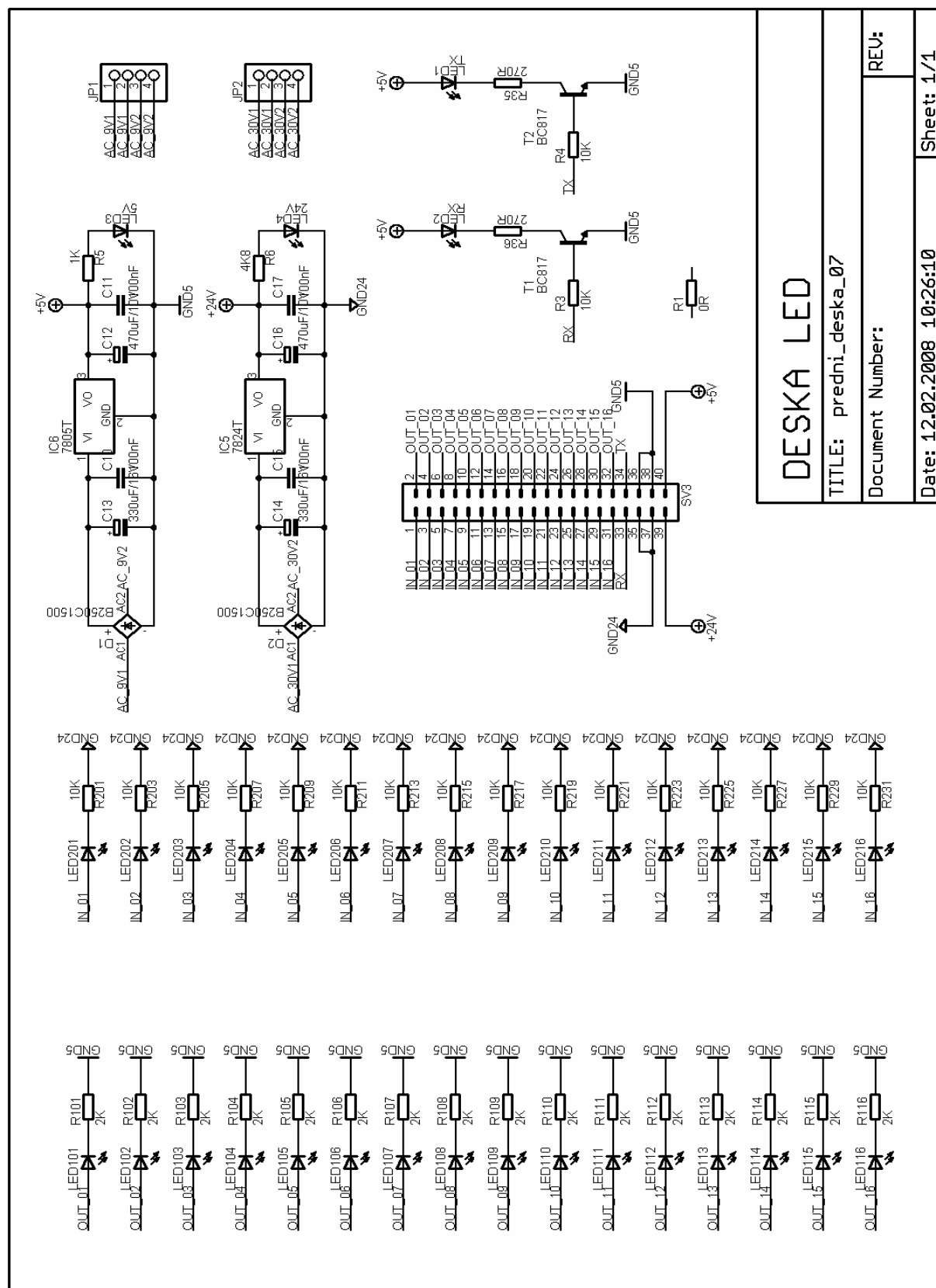
Príloha 1.: Schémata obvodů z PLC do PC



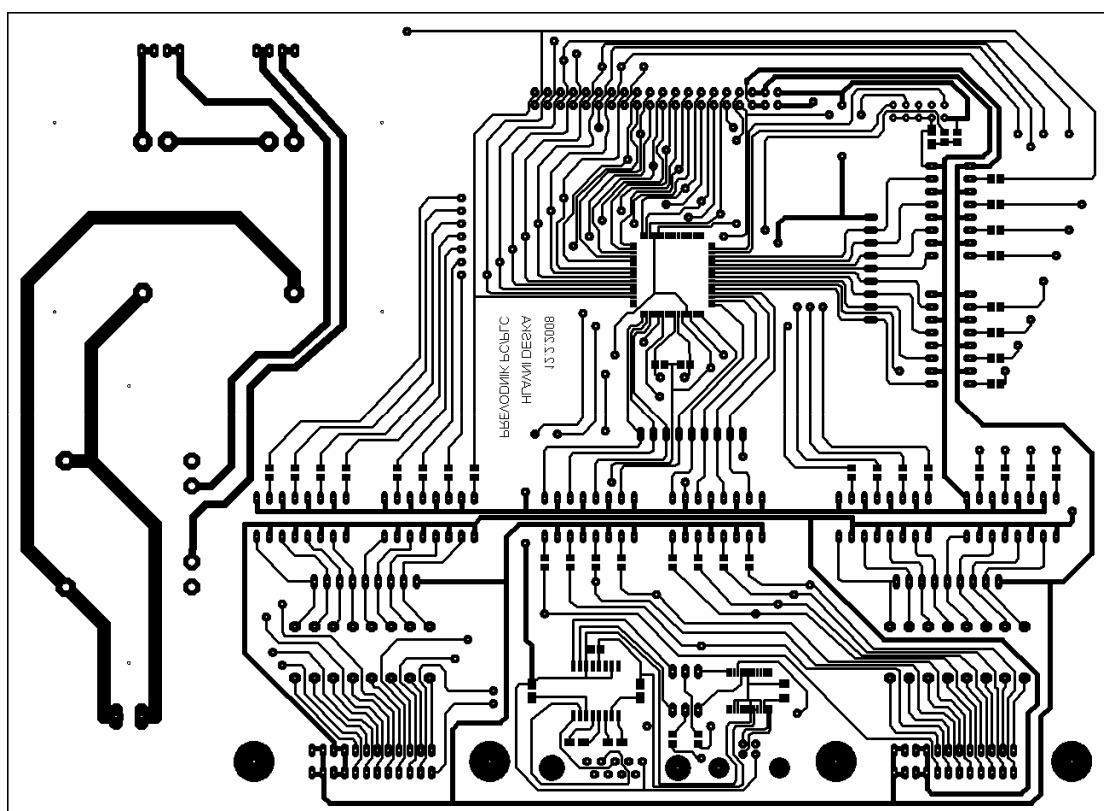
Príloha 2.: Schémata obvodů z PC do PLC



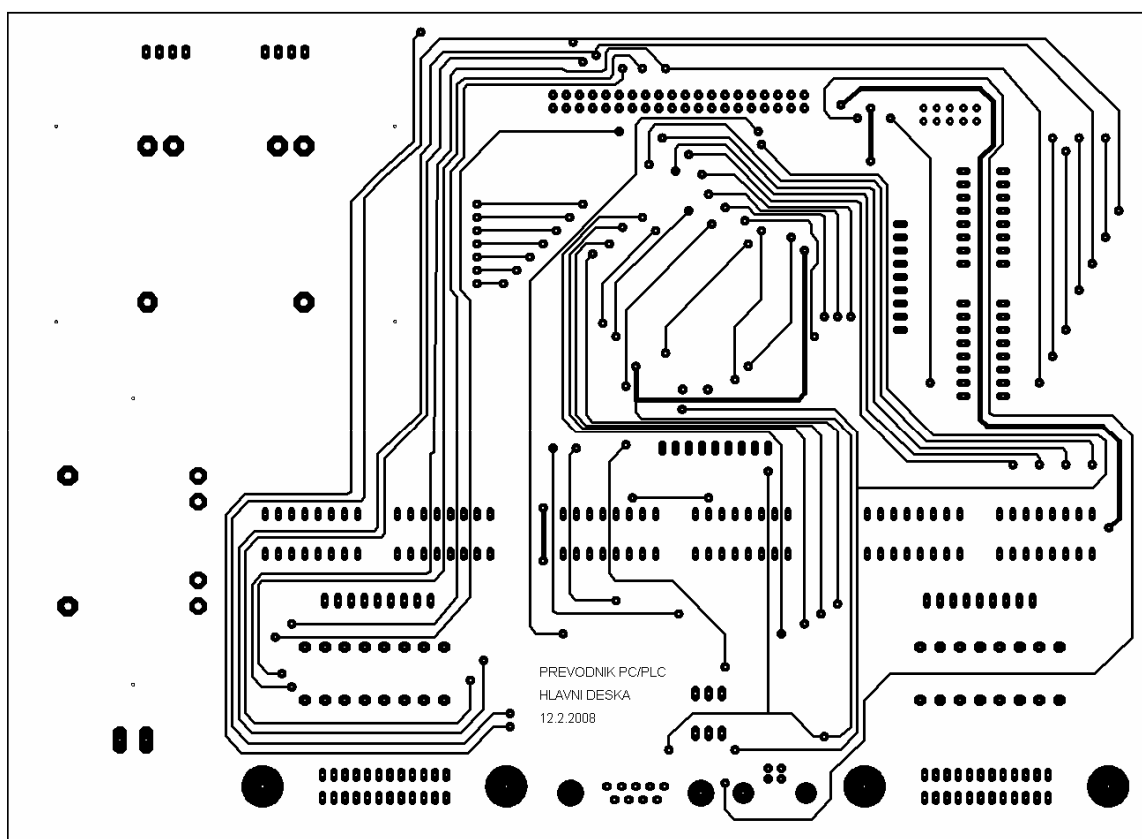
Příloha 3.: Hlavní deska-komunikace



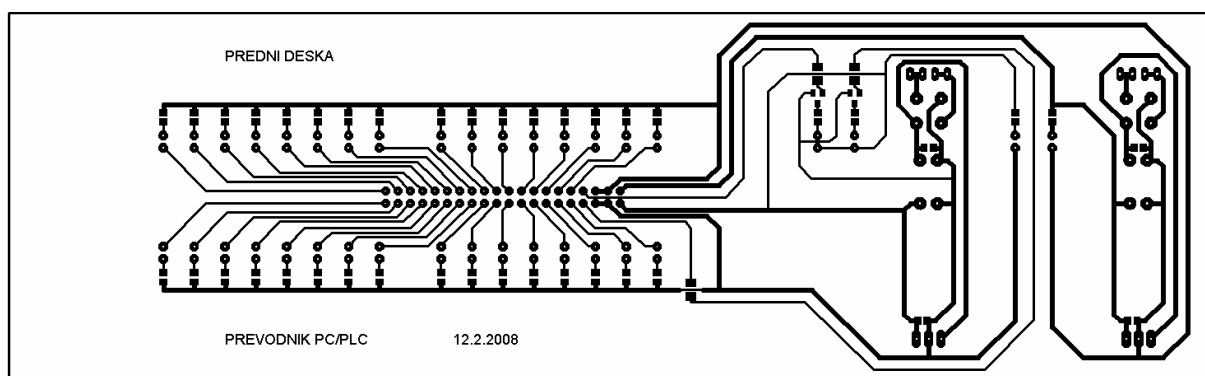
Příloha 4.: Přední deska



Příloha 5.:Spodní část Hlavní desky



Příloha 6.:Horní část Hlavní desky



Příloha 7.:DPS Pření deska

Příloha 8.: Kód Programu

```

#include <avr/io.h>                                // vložení knihovny io.h
#include <avr/interrupt.h>                          // vložení knihovny interrupt.h

#define setb(port,pin) port |= 1<<pin              // definuje nastavení bitu
#define clrb(port,pin) port &= ~(1<<pin)           // definuje vynulování bitu
#define negb(port,pin) port ^= 1<<pin              // definuje negování bitu

#define HLAVICKA1          120                      // komunikační hlavička první část
#define HLAVICKA2          200                      // komunikační hlavička druhá část

#define OUT_01              PORTC0                  // výstupy z převodníku do PLC
#define OUT_02              PORTC1                  // výstupy z převodníku do PLC
#define OUT_03              PORTC2                  // výstupy z převodníku do PLC
#define OUT_04              PORTC3                  // výstupy z převodníku do PLC
#define OUT_05              PORTC4                  // výstupy z převodníku do PLC
#define OUT_06              PORTC5                  // výstupy z převodníku do PLC
#define OUT_07              PORTC6                  // výstupy z převodníku do PLC
#define OUT_08              PORTC7                  // výstupy z převodníku do PLC

#define OUT_09              PORTA7                  // výstupy z převodníku do PLC
#define OUT_10              PORTA6                  // výstupy z převodníku do PLC
#define OUT_11              PORTA5                  // výstupy z převodníku do PLC
#define OUT_12              PORTA4                  // výstupy z převodníku do PLC
#define OUT_13              PORTA3                  // výstupy z převodníku do PLC
#define OUT_14              PORTA2                  // výstupy z převodníku do PLC
#define OUT_15              PORTA1                  // výstupy z převodníku do PLC
#define OUT_16              PORTA0                  // výstupy z převodníku do PLC

#define IN_01               PORTE2                  // vstupy do převodníku z PLC
#define IN_02               PORTE3                  // vstupy do převodníku z PLC
#define IN_03               PORTE4                  // vstupy do převodníku z PLC
#define IN_04               PORTE5                  // vstupy do převodníku z PLC
#define IN_05               PORTE6                  // vstupy do převodníku z PLC
#define IN_06               PORTE7                  // vstupy do převodníku z PLC
#define IN_07               PORTB0                  // vstupy do převodníku z PLC
#define IN_08               PORTB2                  // vstupy do převodníku z PLC

#define IN_09               PORTB3                  // vstupy do převodníku z PLC
#define IN_10               PORTB4                  // vstupy do převodníku z PLC
#define IN_11               PORTB5                  // vstupy do převodníku z PLC
#define IN_12               PORTB6                  // vstupy do převodníku z PLC
#define IN_13               PORTB7                  // vstupy do převodníku z PLC
#define IN_14               PORTD5                  // vstupy do převodníku z PLC
#define IN_15               PORTD6                  // vstupy do převodníku z PLC
#define IN_16               PORTD7                  // vstupy do převodníku z PLC

```

```

#define RX                PORTD3    // Příjem dat
#define TX                PORTD2    // Odesílání dat

// ===== INIT PORTY =====
// (počáteční nastavení portů)

void port_init(void)      // datový typ bez hodnoty pro funkci port_init
{
    DDRA = 0B11111111;    // výstupní port
    PORTA = 0x00;         // výstup na 0

    DDRB = 0B00000000;    // vstupní port, používá se 0,2,3,4,5,6,7
    PORTB = 0xFF;         // zapnu Pull UP

    DDRC = 0B11111111;    // výstupní port
    PORTC = 0x00;         // výstup na 0

    DDRD = 0B00001000;    // vstupní port, používá se 5,6,7
    PORTD = 0xFF;         // zapnu Pull UP

    DDRE = 0B00000000;    // vstupní port, používá se 2,3,4,5,6,7
    PORTE = 0xFF;         // zapnu Pull UP

    DDRF = 0B00000000;    // vstupní port, NE-používá se
    PORTF = 0xFF;         // zapnu Pull UP

    DDRG = 0B000000;      // vstupní port, NE-používá se
    PORTG = 0x1F;         // zapnu Pull UP
}

// ===== READ INPUTS =====
// ( načtení vstupů )

unsigned int Read_Inputs(void) // absolutní celočíselný datový typ pro funkci Read_Inputs
{
    int output = 0;           // deklarace proměnné jako celočíselný datový typ

    unsigned char IN01 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN02 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN03 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN04 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN05 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN06 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN07 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN08 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN09 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN10 = 0;   // deklarace proměnné jako řetězec
    unsigned char IN11 = 0;   // deklarace proměnné jako řetězec
}

```

```

unsigned char IN12 = 0;      // deklarace proměnné jako řetězec
unsigned char IN13 = 0;      // deklarace proměnné jako řetězec
unsigned char IN14 = 0;      // deklarace proměnné jako řetězec
unsigned char IN15 = 0;      // deklarace proměnné jako řetězec
unsigned char IN16 = 0;      // deklarace proměnné jako řetězec

```

```

if ( !(bit_is_set(PINE,IN_01))) IN01 = 1; else IN01 = 0;
if ( !(bit_is_set(PINE,IN_02))) IN02 = 1; else IN02 = 0;
if ( !(bit_is_set(PINE,IN_03))) IN03 = 1; else IN03 = 0;
if ( !(bit_is_set(PINE,IN_04))) IN04 = 1; else IN04 = 0;
if ( !(bit_is_set(PINE,IN_05))) IN05 = 1; else IN05 = 0;
if ( !(bit_is_set(PINE,IN_06))) IN06 = 1; else IN06 = 0;
if ( !(bit_is_set(PINB,IN_07))) IN07 = 1; else IN07 = 0;
if ( !(bit_is_set(PINB,IN_08))) IN08 = 1; else IN08 = 0;
if ( !(bit_is_set(PINB,IN_09))) IN09 = 1; else IN09 = 0;
if ( !(bit_is_set(PINB,IN_10))) IN10 = 1; else IN10 = 0;
if ( !(bit_is_set(PINB,IN_11))) IN11 = 1; else IN11 = 0;
if ( !(bit_is_set(PINB,IN_12))) IN12 = 1; else IN12 = 0;
if ( !(bit_is_set(PINB,IN_13))) IN13 = 1; else IN13 = 0;
if ( !(bit_is_set(PIND,IN_14))) IN14 = 1; else IN14 = 0;
if ( !(bit_is_set(PIND,IN_15))) IN15 = 1; else IN15 = 0;
if ( !(bit_is_set(PIND,IN_16))) IN16 = 1; else IN16 = 0;

```

*Jestliže bit není jedna na
PINU x a PORTU x, tak
INxx = 1, jinak INxx = 0*

*Negace zde je kvůli otočené
logice použitého obvodu viz.
Kapitola 2.1.4*

```

output = ((IN01*1) + (IN02*2) + (IN03*4) + (IN04*8) + (IN05*16) + (IN06*32) +
(IN07*64) + (IN08*128) + (IN09*256) + (IN10*512) + (IN11*1024) + (IN12*2048) +
(IN13*4096) + (IN14*8192) + (IN15*16384) + (IN16*32768));

```

*// proměnná output se naplní jako celočíselná hodnota, které se dá převést na 16-bitovou
hodnotu pro komunikaci*

```

return output;      // návratová hodnota funkce
}

```

```

// ===== SET OUTPUTS =====
( natavení výstupů )

```

*int Set_Outputs(int port1, int port2) //funkce v celočíselném datovém typu, která má dva
parametry*

```

{
if (port1 & 0B00000001) setb(PORTC,OUT_01); else clrb(PORTC,OUT_01);
if (port1 & 0B00000010) setb(PORTC,OUT_02); else clrb(PORTC,OUT_02);
if (port1 & 0B00000100) setb(PORTC,OUT_03); else clrb(PORTC,OUT_03);
if (port1 & 0B00001000) setb(PORTC,OUT_04); else clrb(PORTC,OUT_04);
if (port1 & 0B00010000) setb(PORTC,OUT_05); else clrb(PORTC,OUT_05);
if (port1 & 0B00100000) setb(PORTC,OUT_06); else clrb(PORTC,OUT_06);
if (port1 & 0B01000000) setb(PORTC,OUT_07); else clrb(PORTC,OUT_07);
if (port1 & 0B10000000) setb(PORTC,OUT_08); else clrb(PORTC,OUT_08);

```

```

if (port2 & 0B00000001) setb(PORTA,OUT_09); else clrb(PORTA,OUT_09);
if (port2 & 0B00000010) setb(PORTA,OUT_10); else clrb(PORTA,OUT_10);
if (port2 & 0B00000100) setb(PORTA,OUT_11); else clrb(PORTA,OUT_11);
if (port2 & 0B00001000) setb(PORTA,OUT_12); else clrb(PORTA,OUT_12);
if (port2 & 0B00010000) setb(PORTA,OUT_13); else clrb(PORTA,OUT_13);
if (port2 & 0B00100000) setb(PORTA,OUT_14); else clrb(PORTA,OUT_14);
if (port2 & 0B01000000) setb(PORTA,OUT_15); else clrb(PORTA,OUT_15);
if (port2 & 0B10000000) setb(PORTA,OUT_16); else clrb(PORTA,OUT_16);

```

Příklad maskování:

*Jestliže port1 se shoduje s 0B00000001, tak nastav PORTC, OUT_01 jako logikou jedničku, jinak PORTC, OUT_01 vynuluj.(tzv. maskování)
(Porovnáváme např. 10010011 s 00000001, v tomto případě nastavíme 1)*

```

return(0); //návratová hodnota je nulová
}

// ===== UART INIT =====
// (inicializace sériové linky)

// UART1 initialize
// desired baud rate:115200 (nastavená rychlost přenosu dat)
// actual baud rate:115198 (0,0%) (aktuální rychlost přenosu dat)
// char size: 8 bit (velikost řetězce)
// parity: Disabled (parita: žádná)

void uart1_init(void) //funkce pro naplnění sériové komunikace- nastavení registrů
{
    UCSR1B = 0x00; //vypnutá když nastavujeme baudovou rychlost
    UCSR1A = 0x00;
    UCSR1C = 0x06;
    UBRR1L = 0x05; //nastavena baud rate na minimum
    UBRR1H = 0x00; //nastavena baud rate na maximum
    UCSR1B = 0x98;
}

/===== UART RECEIVE =====
// (Přijímání dat sériovou linkou)

int UART_Receive(void) // Definice funkce
{
    while ( !(UCSR1A & (1<<RXC)) ); // Čeká se než jsou data přijaty
    return UDR1; // Dostane a navrátí přijaté data z bufferu
}

```

V tomto případě se nepoužívá


```

else prijato = 0;           // jinak přijato = 0 a funkce se vrací na začátek
}

if (prijato == 3)           // jestliže přijato se rovná 3
    znak3 = UDR1;           // znak3 naplní registr UDR1

if (prijato == 4)           // jestliže přijato se rovná 4
{
    znak4 = UDR1;           // znak4 naplní registr UDR1
    Set_Outputs(znak3, znak4); // zavolá se funkce Set_Outputs a naplní se parametry
                                // znak3 a znak4
    prijato = 0;           // přijato se vynuluje
}
}

// ===== TIMER1 INIT =====
// (inicializace časovače)

//TIMER1 initialize - prescale:1024 //inicializace časovače s předděličkou 1024
// WGM: 0) Normal, TOP=0xFFFF
// desired value: 50mSec //požadovaná hodnota
// actual value: 49,908mSec (0,2%) //dosažená hodnota s 0,2% chybou

void timer1_init(void)       //nulová funkce
{
    TCCR1B = 0x00;           //registr pro zastavení časovače
    TCNT1H = 0xFD;           //nastavení časovače (maximum)
    TCNT1L = 0xE5;           //nastavení časovače (minimum)
    OCR1AH = 0x0A;
    OCR1AL = 0x8B;
    OCR1BH = 0x0A;
    OCR1BL = 0x8B;
    OCR1CH = 0x0A;
    OCR1CL = 0x8B;
    ICR1H = 0x0A;
    ICR1L = 0x8B;
    TCCR1A = 0x00;
    TCCR1B = 0x05;           //registr pro spuštění časovače
}

```

```
// ===== TIMER1 PRERUSENI =====
                                (Přerušeni časovače)

ISR(TIMER1_OVF_vect)
{
    //TIMER1 má přetéct
    TCNT1H = 0xFD;              // načtení největší hodnoty počítadla
    TCNT1L = 0xE5;              // načtení nejmenší hodnoty počítadla

    UART_Transmit(Read_Inputs()); //odešlu stavy portů
}

//===== MAIN =====
                                ( Hlavní program)

int main (void)
{
    asm volatile("CLI");        // vypne všechna přerušeni

    port_init();                 // inicializace portů
    uart1_init();                // inicializace sériové linky
    timer1_init();               // inicializace časovače

    MCUCR = 0x00;
    TIMSK = 0x04;                // zdroj přerušeni časovače

    asm volatile("SEI");        // znovu povolit přerušeni

    for(;;)
    {
        // NIC
    }
}

//===== END =====
```

Příloha 9.: Seznam Součástí

Kód	Název	Cena s DPH	Počet kusů	Celková cena s DPH
901-121	R0805 510R 5%	1,00 Kč	16	16,00 Kč
610-823	TRHEI541 - 2x15	194,00 Kč	1	194,00 Kč
610-800	TRHEI422-1X9	117,00 Kč	1	117,00 Kč
806-117	GSI-2	25,00 Kč	1	25,00 Kč
631-215	P-H8600VB01T	10,00 Kč	1	10,00 Kč
840-007	KONPC-SPK-8	2,00 Kč	2	4,00 Kč
123-137	E220M/25V	1,50 Kč	1	1,50 Kč
123-122	E100M/50V	1,50 Kč	1	1,50 Kč
802-001	CENTR.24V	50,00 Kč	2	100,00 Kč
623-066	DA5M3X08	2,50 Kč	6	15,00 Kč
623-076	KDA6M3X08	5,00 Kč	2	10,00 Kč
662-025	SPC306	3,20 Kč	2	6,40 Kč
662-005	SKM3X6	0,50 Kč	6	3,00 Kč
622-227	U-SP7771	165,00 Kč	1	165,00 Kč
661-042	CU-TA053	87,00 Kč	1	87,00 Kč
661-024	CU-TA214	204,00 Kč	1	204,00 Kč
330-009	T7824	8,00 Kč	1	8,00 Kč
620-061	DO3A	3,00 Kč	2	6,00 Kč
523-060	PC847	14,00 Kč	10	140,00 Kč
906-079	CK0805 22P NPO	2,50 Kč	2	5,00 Kč
906-089	CK0805 10N X7R	2,50 Kč	1	2,50 Kč
905-081	CK1206 100N X7R	2,50 Kč	1	2,50 Kč
958-107	ATmega128-16AU	145,00 Kč	1	145,00 Kč
959-303	FT232RL	140,00 Kč	1	140,00 Kč
131-077	QM 11.059MHZ	9,90 Kč	1	9,90 Kč
901-212	R0805 100R 1%	2,00 Kč	2	4,00 Kč
901-188	R0805 4K7 1%	2,00 Kč	1	2,00 Kč
901-176	R0805 10K 1%	1,00 Kč	33	33,00 Kč
901-066	R0805 2K 5%	1,00 Kč	32	32,00 Kč
959-027	MAX232CWE	23,00 Kč	1	23,00 Kč
907-112	CTS 10M/16V A	3,00 Kč	5	15,00 Kč
912-005	BC817-16 SMD	1,50 Kč	2	3,00 Kč
906-096	CK0805 100N X7R	2,50 Kč	4	10,00 Kč
900-001	R1206 0R	2,00 Kč	1	2,00 Kč
900-179	R1206 10K 1%	2,00 Kč	2	4,00 Kč
901-035	R0805 1K8 5%	2,00 Kč	1	2,00 Kč
901-178	R0805 1K 1%	2,00 Kč	1	2,00 Kč
901-063	R0805 270R 5%	2,00 Kč	2	4,00 Kč
123-153	E470M/10V	2,30 Kč	2	4,60 Kč
123-147	E330M/16V	2,50 Kč	1	2,50 Kč
110-073	RR 1K	0,80 Kč	16	12,80 Kč
111-266	RR 8X10K 2%	3,50 Kč	4	14,00 Kč
824-004	DIL16PZ	8,90 Kč	8	71,20 Kč
800-067	MLW40G	11,50 Kč	2	23,00 Kč
801-037	CAN 9 V 90	9,00 Kč	1	9,00 Kč
832-017	SIG20	4,00 Kč	1	4,00 Kč
802-003	CENTR.24Z 90	45,00 Kč	2	90,00 Kč
832-120	USB1X90B PCB	6,00 Kč	1	6,00 Kč
821-017	ARK500/2	5,00 Kč	1	5,00 Kč

Kód	Název	Cena s DPH	Počet kusů	Celková cena s DPH
800-010	PFL20	7,50 Kč	2	15,00 Kč
800-086	PFH02-04P	1,00 Kč	4	4,00 Kč
800-191	PFF02-01FG TAPE	0,60 Kč	20	12,00 Kč
800-170	PSH02-04PG	2,00 Kč	4	8,00 Kč
227-004	B250C1500	3,00 Kč	2	6,00 Kč
330-102	78T05	20,00 Kč	1	20,00 Kč
511-410	LED 3MM RED 200MCD	1,00 Kč	32	32,00 Kč
511-052	LED 3MM YELLOW	1,50 Kč	2	3,00 Kč
511-200	LED 3MM 2MA/G	14,00 Kč	4	56,00 Kč
				1 952,40 Kč