STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12.: Tvorba učebních pomůcek, didaktická technologie

Výukový model automatizační linky s PLC

Jan Novotný, Daniel Přibyl
Olomoucký kraje

Šumperk 2017

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12.: Tvorba učebních pomůcek, didaktická technologie

Výukový model automatizační linky s PLC

Teaching model, automation lines with PLC

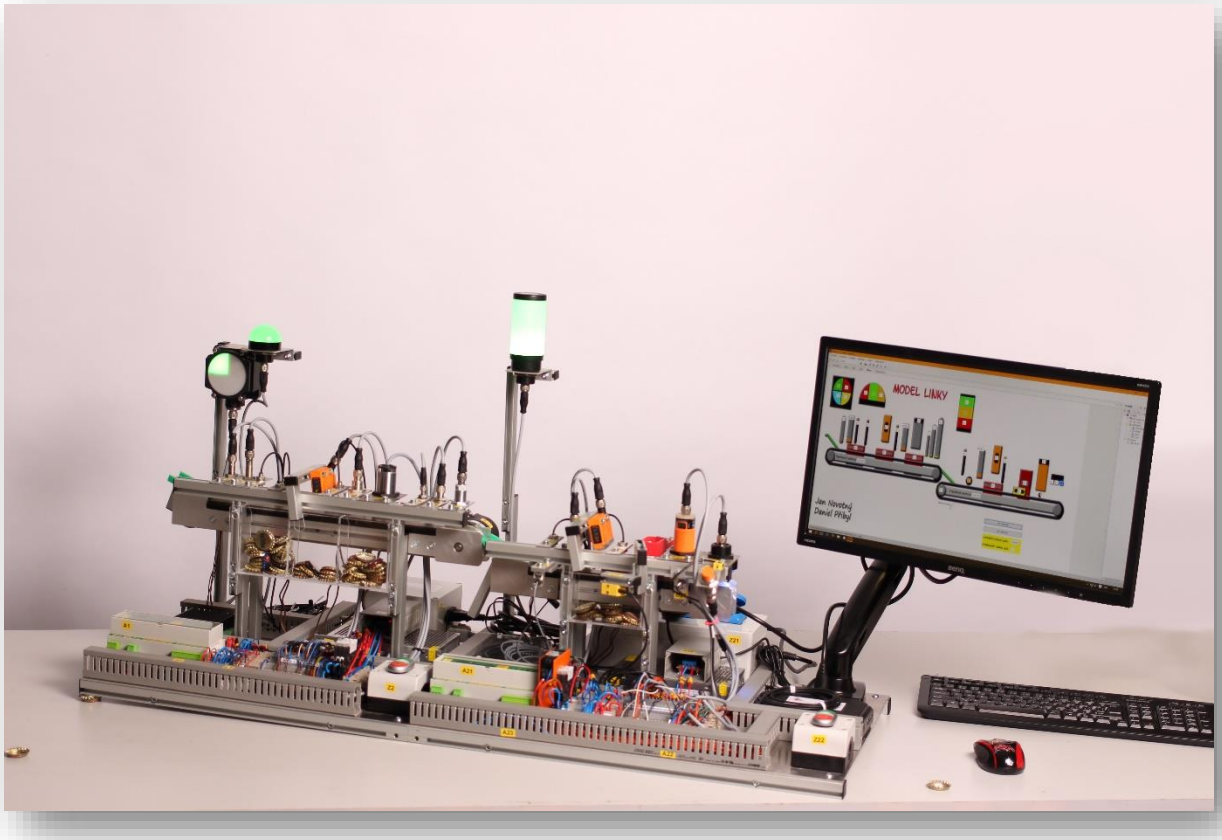
Autoři: Jan Novotný, Daniel Přibyl

Škola: Vyšší odborná a Střední průmyslová škola Šumperk, Gen. Krátkého 1, 787 29

Kraj: Olomoucký kraj

Konzultant: Ing. Roman Kubíček, Ing. Vít Krňávek

Šumperk 2017



Výukový model automatizační linky s PLC

Výukový modul

Jan Novotný, Daniel Přibyl

VOŠ a SPŠ Šumperk

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Šumperku dne _____ ……

Poděkování

Chtěli bychom poděkovat firmě Elzaco za poskytnutí elektromateriálu, konstrukčního materiálu a za obětavou pomoc při řešení a realizaci konstrukce linek.

Dále firmám IFM, TURCK, BALLUFF za poskytnutí senzorů.

A také třídnímu učiteli, Vítu Krňávkovi, který nás vedl a podporoval v práci.

Anotace

Tato práce se zabývá návrhem a tvorbou funkčního modelu automatizační linky, která bude využita ve výuce programování řídicích systémů. Programovatelná linka je rozdělena na dvě samostatně pracující části, které lze propojit. Zpracovává několik úkolů – pohon, třídění, plnění, kontrolu a signalizaci. Součástí projektu je i možnost její vizualizace.

Klíčová slova

Linka, PLC, AMiT, AMiNi2D, pivní uzávěry, dopravník, DetStudio. ViewDet, ALUtech, hliník, senzory, třídění, plnění, vizualizace

Annotation

This paper describes the design and creation of functional model of automation lines, which will be used in teaching programming of control systems. Programmable line is divided into two independently operating parts that can be connected. Handles several tasks - driving, sorting, filling, control and signaling. The project also includes the possibility of its visualization.

Keywords

Line, PLC, AMiT, AMiNi2D, beer caps, conveyor, DetStudio. ViewDet, Yellow, aluminum, sensors, sorting, loading, visualization

Obsah

Seznam obrázků

1. Úvod

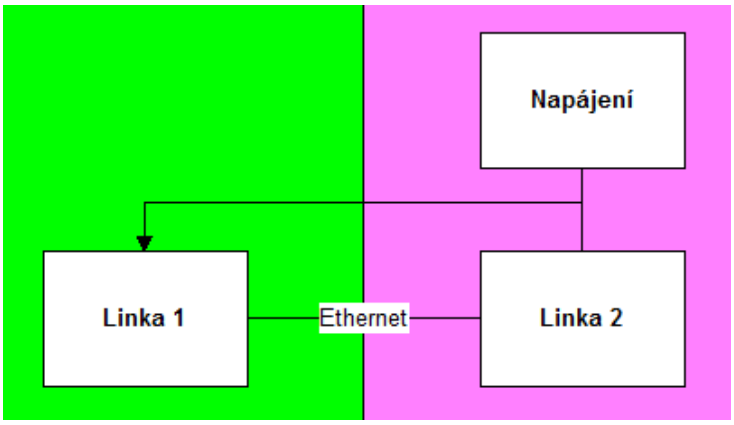
Cílem této práce je navrhnout a vytvořit funkční model automatizační linky, která bude využita ve výuce programování řídicích systémů. Práce se zabývá rozborem komponentů a jejich parametry (řídicí systémy, pohony, senzory, zobrazovače a ostatní); konstrukcí linky a jejím zapojením.

Těžiště práce se nachází v programování linky. Byl zvolen řídicí systém společnosti AMiT. Tato společnost dodává vývojové prostředí DetStudio a pro vizualizaci vývojové prostředí ViewDet, jež byly využity k programování PLC. Linka bude řízena pomocí řídicího systému AMiNi2D.

Automatizovaná linka se skládá ze dvou částí, které lze podle potřeby spojit. Úkolem linky bude třídit kovové uzávěry pivních lahví. Uzávěry budou vyhodnocovány jak digitálními, digitálními programovatelnými, ale také analogovými senzory. Bude se vyhodnocovat natočení, barva, přítomnost těsnění a kapacita uzávěrů. Pokročilejší funkcí linky bude plnění uzávěrů plastovými kuličkami o průměru 6 milimetrů a kontrola tohoto procesu. Pro vizualizaci bude na jedné lince umístěn osobní počítač, přes který se bude moci nahrávat program. Počítač a řídicí systémy budou propojeny pomocí Ethernetu do sítě LAN.

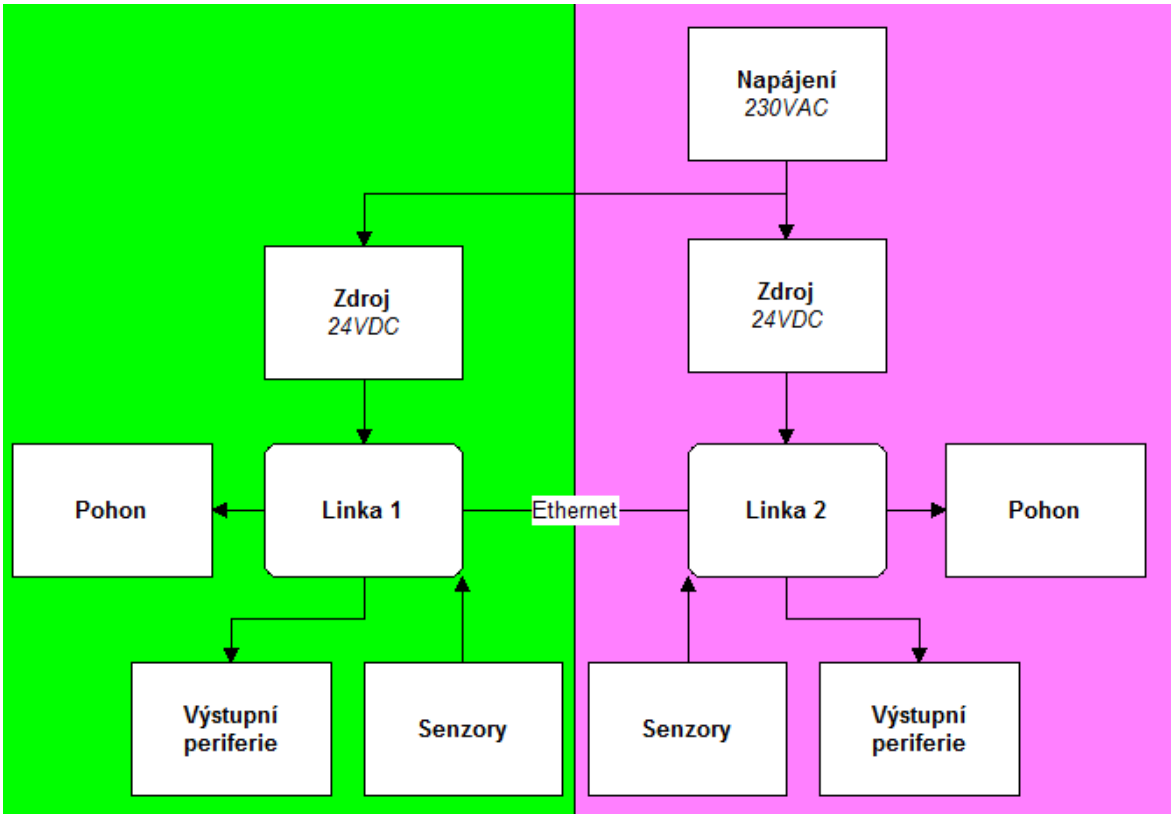
2. Úvod do problematiky práce

Cílem naší práce bylo vytvořit výukový model linky, který bude obsahovat základní úlohy pro úvod do programování automatizace a možnost složitějších úloh. Aby linka nebyla pouze jedna, tak z důvodu možností programování byla rozdělena na dvě části, které se dají spojit a navazují na sebe. Řídící systémy linek budou moci mezi sebou komunikovat pomocí ethernetu.



Obrázek 1: Jednoduché blokové schéma

Zvolili jsme krokové motory, kvůli přesnosti a jednoduchosti ovládání, které se řídí budičem. Použili jsme téměř všechny druhy senzorů (Indukční, analogové, optické, kapacitní) aby byla možnost vyzkoušet, jak jednotlivé senzory reagují v praxi a že není jedno, jaké senzory kam použijeme.



Obrázek 2: Principiální blokové schéma

2.1. PLC

PLC (**P**rogrammable **L**ogic **C**omputer) neboli již dnes **PAC** (**P**rogrammable **A**utomation **C**ontroller), jsou průmyslové počítače se vstupními a výstupními perifériemi. Nejčastěji pracují na 24V logice. Zkratka PLC je již zažitá, a proto se používá i dodnes, i když by se jim mělo říkat PAC. Jsou spousty výrobců PLC (Siemens, BEN, AMiT, …), každý výrobce má svoje programovací prostředí i jazyk. Může to být reléové schéma (RS – program se skládá z bloků, které jsou již před programované) anebo strukturovaný text, může se lišit pojmenovaní, (ST – psaní programu pomocí příkazů.

PLC můžeme rozdělit na:

- Kompaktní:
  Kompaktní PLC mají již v základu vše, co potřebujeme. Můžeme je také rozšířit, ale mají v základu již vstupy a výstupy. Mohou také obsahovat terminál.
- Modulární:
  Modulární PLC nemají v základu vstupně výstupní periférie, ty si musíme nakombinovat. Tyto PLC mají především komunikační linky a sběrnice pro komunikaci s moduly. Tyto modulu můžou být vstupní a výstupní periférie, ale i také terminály, obrazovky a komunikační jednotky, jako jsou třeba Ethernet či RS485, RS232

3. Rozbor použitých komponentů (základní parametry)

Všechny ostatní parametry najdete v přiložených datasheetech.

3.1. Řídící systém AMiNi2D [1]

CPU	SAB C167CR-LM
FLASH	512 KB
RAM	1024 KB
EEPROM	2 KB
Zálohování RAM	5 let bez napájení

Displej – pouze **AMiNi2D**	Podsvětlený, 4 × 20 znaků, * znak 5 × 7 bodů
Výška znaku	4 mm
Klávesnice – pouze **AMiNi2D**	Osm tlačítek

Číslicové vstupy	8 × Logická 0 min. 0 V, max. 5 V Logická 1 min. 16 V, max. 30 V
Typ vstupu	24 V ss / 24 V stř
Galvanické oddělení	Ano, společně s DO
Pevnost galvanického oddělení	300 V **

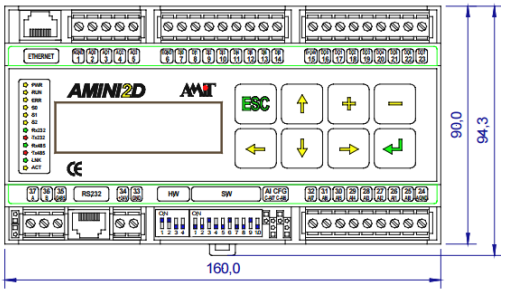
Číslicové výstupy	8 × Tranzistorové výstupy 24 V ss/0,3 A
Proud proudové ochrany	Typicky 1,5 A
Galvanické oddělení	Ano, společně s DI
Pevnost galvanického oddělení	300 V **

Analogové vstupy	6 ×
Typ vstupu	Ni1000
Analogové vstupy	2 ×
Typ vstupu	0 .. 5 V/ 0 .. 10 V/0 .. 20 mA/Ni1000
Galvanické oddělení	Ne
Ochrana analogových vstupů	Diody + odpor 10 KΩ

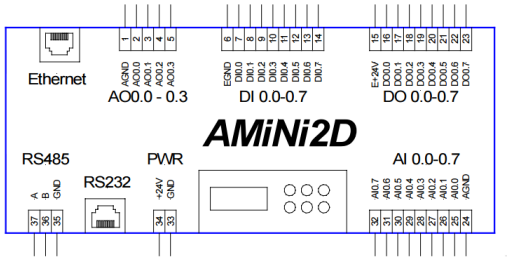
Sériový komunikační kanál	RS232 - bez galv. oddělení RS485 – s galv. oddělením 300 V **

Rozhraní Ethernet	10 Mbps Řadič RTL8019AS

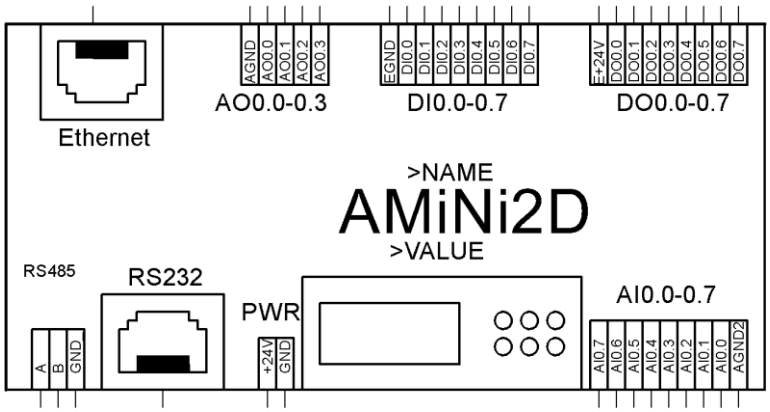
Mechanické provedení	Plastová krabička
Montáž	Na DIN lištu 35 mm
Krytí	IP20
Připojení signálů	Šroubovací konektory PA256 VE (5,08 mm)
Napájení	24 V ss ±20 %
Maximální odběr 24V	200 mA



Obrázek 5: Rozměry AMiNi2D



Obrázek 4: Doporučená schématická značka AMiNi2D



Obrázek 3: Naše schématická značka AMiNi2D

3.2. Pohony

3.2.1. Motor HS2231 (MAE) [2]

Podrobnosti k motoru najdete v produktovém listu (datasheetu).

Obrázek 6: Motor HS2231

3.2.2. Budič M542 (Leadshine) [3]

- Napájecí napětí do +50VDC, špičkový proud do 4,2A;
- Nastavení proudu fáze od 1 do 4,2A (8 hodnot);
- 15 volitelných rozlišení kroku až do 25000 kroků/ot;
- Maximální vstupní frekvence 300 KHz;
- Opticky izolované vstupní signály;
- Vhodný pro 2-fázové a 4-fázové krokové motory;
- Ochrana proti zkratu na výstupu, podpětí a přepětí napájecího napětí;
- Automatická redukce proudu po zastavení motoru;

Obrázek 7: Budič M542

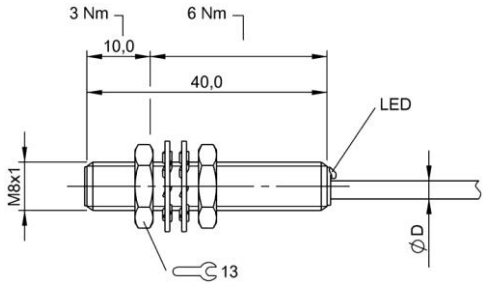
3.3. Senzory

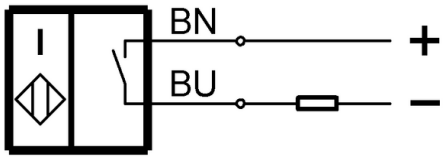
3.3.1. Indukční

3.3.1.1. Digitální

3.3.1.1.1. BES M08MG-GSC20B-BP03

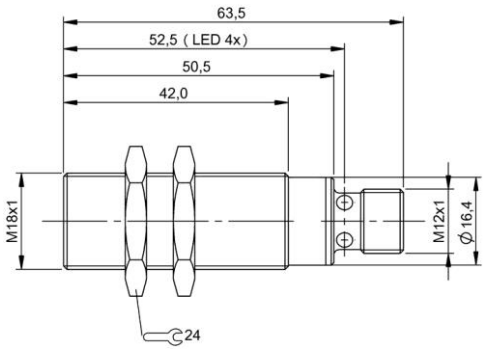
- Rozměry: M08x1;
- Připojení: Kabel, PUR, 3.00 m;
- Funkce výstupu: pólovaný, Spínací kontakt (NO);
- Provozní napětí: 10.0…36.0 V;
- Mechanická montážní podmínka: Vestavný v jedné rovině;
- Jmenovitá spínací vzdálenost Sn [mm]: 2.00 mm;

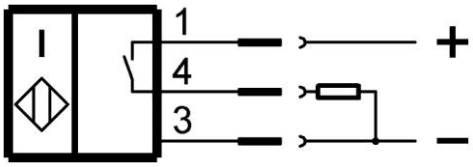




3.3.1.1.2. BES M18MG1-PSC12B-S04G

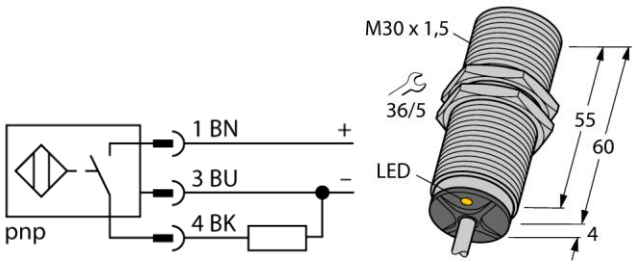
- Rozměry: M18x1;
- Připojení: Konektory, M12x1-S04;
- Funkce výstupu: PNP, Spínací kontakt (NO);
- Provozní napětí: 10.0…30.0 V;
- Zvláštní vlastnosti: velká spínací vzdálenost;
- Mechanická montážní podmínka Kvazivestavný;
- Jmenovitá spínací vzdálenost Sn [mm]: 12.00 mm;





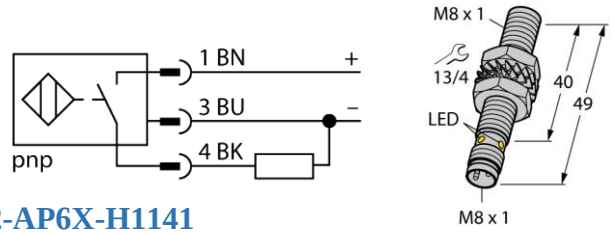
3.3.1.1.3. BI15U-M30-AP6X

- závitové pouzdro M30x1,5;
- chromovaná mosaz;
- faktor 1 pro všechny kovy;
- stupeň krytí IP68;
- odolnost vůči magnetickému poli;
- vysoká spínací vzdálenost;
- vestavná montáž možná;
- DC 3drát, 10…30 VDC;
- spínací PNP výstup;
- připojení kabelem;



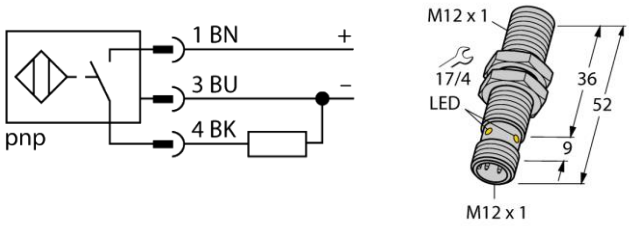
3.3.1.1.4. BI2-EG08-AP6X-V1131

- se zvýšenou spínací vzdáleností;
- závitové pouzdro M8x1;
- nerez 1.4427 SO;
- větší snímací rozsah;
- 3drát DC, 10…30 VDC;
- spínací PNP výstup;
- konektor M8x1;



3.3.1.1.5. BI4U-M12-AP6X-H1141

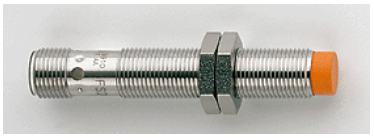
- závit M12 x 1;
- chromovaná mosaz;
- faktor 1 pro všechny kovy;
- stupeň krytí IP68;
- odolnost vůči magnetickému poli;
- vysoká spínací vzdálenost;
- vestavná montáž možná;
- 3drát DC, 10…30 VDC;
- spínací PNP výstup;
- konektor M12x1;



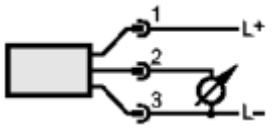
3.3.1.2. Analogové

3.3.1.2.1. Napěťové

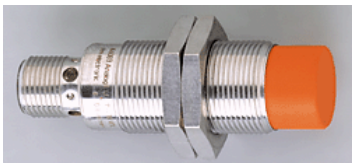
3.3.1.2.1.1. IF6031



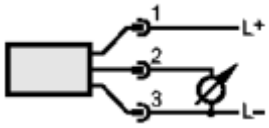
- Kovový závit M12 x 1;
- Konektorové provedení;
- Analogový výstup 0...10 V;
- (lineární, stoupání: 2,77 V/mm) *);
- Měřící rozsah: 0,4...4 mm;
- Elektrické provedení: DC analog.;
- Provozní napětí [V] : 15...30 DC;
- Proudový odběr [mA]: < 20;



3.3.1.2.1.2. IG6084

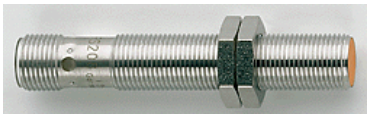


- Kovový závit M18 x 1;
- Konektorové provedení;
- Analogový výstup 0...10 V;
- (lineární, stoupání: 1,389 V/mm) *);
- Měřící rozsah 0,8...8 mm;
- Elektrické provedení: DC analog.;
- Provozní napětí [V]: 15...30 DC;
- Proudový odběr [mA]: < 20;

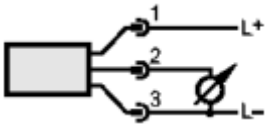


3.3.1.2.2. Proudové

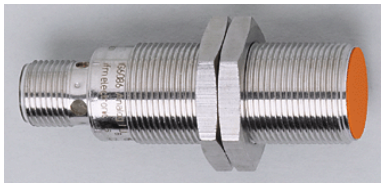
3.3.1.2.2.1. IF6028



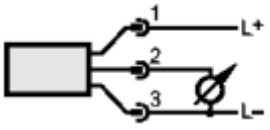
- Induktivní senzor;
- Kovový závit M12 x 1;
- Konektorové provedení;
- Analogový výstup 4...20 mA;
- (lineární, stoupání: 8,88 mA/mm);
- Měřící rozsah 0,2...2 mm;
- Elektrické provedení: DC analog.;
- Provozní napětí [V]: 15...30 DC;
- Proudový odběr [mA] : < 20;



3.3.1.2.2.2. IG6086

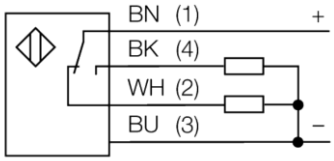


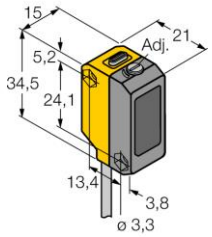
- Kovový závit M18 x 1;
- Konektorové provedení;
- Analogový výstup 4...20 mA;
- (lineární, stoupání: 3,56 mA/mm) *);
- Měřící rozsah 0,5...5 mm;
- Elektrické provedení: DC analog.;
- Provozní napětí [V]: 15...30 DC;
- Proudový odběr [mA]: < 20;



3.3.2. Optické

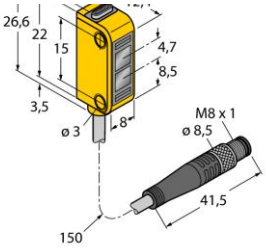
3.3.2.1. QS18VP6LAF



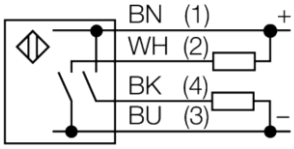


- **Laserový reflexní snímač s nastavitelným zacloněním pozadí;**
- PVC kabel 2 m;
- stupeň krytí IP67;
- dobře viditelné LED;
- mez zaclonění nastavitelná potenciometrem;

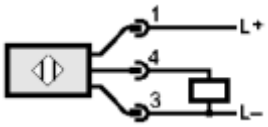
3.3.2.2. Q12AB6FF30Q

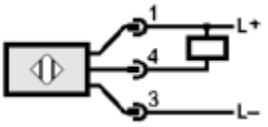


- **reflexní snímač s pevným zacloněním pozadí**
- miniaturní senzor;
- PVC kabel 150 mm s 4pinovým konektorem M8 x 1;
- stupeň krytí IP67;
- dobře viditelné LED;
- zobrazení nízké funkční rezervy;
- napájecí napětí: 10…30VDC;
- bipolární spínací výstup, spínání světlem;



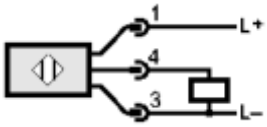
3.3.2.3. O5K500

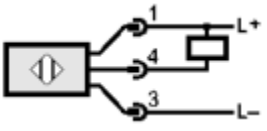




- **Kontrastní snímač;**
- Umělá hmota - kvádrový profil;
- Konektorové provedení;
- Automatické rozpoznání PNP/NPN;
- Teach funkce;
- Elektronický zámek;
- Snímací vzdálenost 18...22mm;
- Elektrické provedení: DC PNP/NPN;
- Provozní napětí [V]: 10...36 DC;
- Proudový odběr [mA]: 50 (24 V);
- Druh světla: červené světlo 625 nm; zelené světlo 525 nm; Modré světlo 465 nm;

3.3.2.4. O5C500



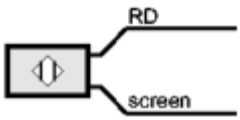


- **Senzor pro rozlišení barev;**
- Umělá hmota - kvádrový profil;
- Konektorové provedení;
- Rozpoznání barvy na principu difúzního odrazu;
- Nastavení rozlišení barev;
- Automatiké rozpoznání PNP/NPN;
- Teach funkce;
- Elektronický zámek;
- Snímací vzdálenost 15...19mm;
- Elektrické provedení: DC PNP/NPN;
- Provozní napětí [V]: 10...36 DC;
- Proudový odběr [mA]: 50 (24 V);
- Druh světla: červené světlo 625 nm; zelené světlo 525 nm; Modré světlo 465 nm;

3.3.2.5. Optozávora

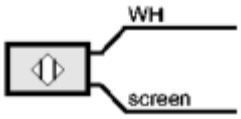
3.3.2.5.1. OE0003

- **Jednocestná světelná závora;**
- Rozměr závitu;
- M8 x 1;
- Kabelové provedení;
- Vysílač;
- Dosah 1m;
- Elektrické provedení:
  připojení na zesilovač OV5012 nebo OV0010;
- Druh světla: Infračervené světlo 950 nm;



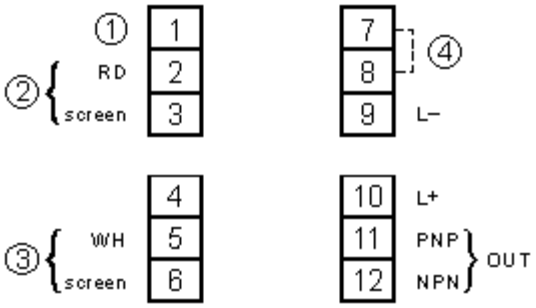
3.3.2.5.2. OE0002

- **Jednocestná světelná závora;**
- Rozměr závitu;
- M8 x 1;
- Kabelové provedení;
- Přijímač;
- Dosah 1m;
- Elektrické provedení:
  připojení na zesilovač OV5012 nebo OV0010;
- Druh světla: Infračervené světlo 950 nm;



3.3.2.5.3. Vyhodnocovací jednotka (OV5012)

- Vyhodnocovací jednotka pro optické senzory;
- Pro montáž na DIN lištu;
- Vstup pro uzamykací signál;
- 2 tranzistorové výstupy;
- Elektrické provedení: DC PNP/NPN;
- Provozní napětí [V]: 10...55 DC;
- Proudový odběr [mA]: 40 (24 V);



1: uzavírací signál

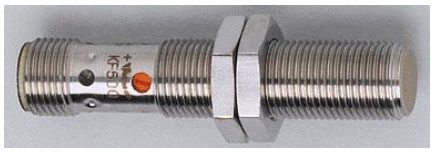
2: vysílač

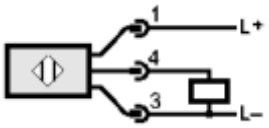
3: přijímač

4: výstupní fce

3.3.3. Kapacitní

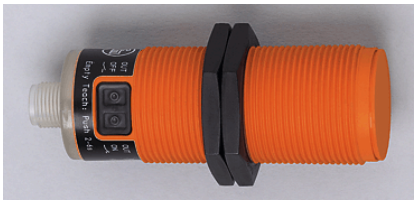
3.3.3.1. KF5001

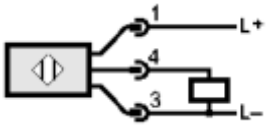




- Kovový závit M12 x 1;
- Konektorové provedení;
- Zvýšená odolnost proti rušení;
- Spínací vzdálenost 4 mm; nastavitelná 1...6 mm;
- Elektrické provedení: DC PNP;
- Provozní napětí [V]: 10...36 DC;
- Proudový odběr [mA] : < 12;
- Výstupní funkce: Spínač;

3.3.3.2. KI5083





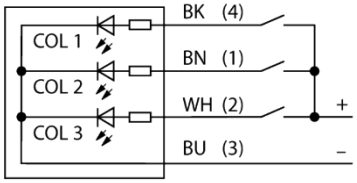
- závit v umělé hmotě M30 x 1,5;
- Konektorové provedení;
- Teach funkce;
- Elektronický zámek;
- Spínací vzdálenost 20 mm;
- Elektrické provedení: DC PNP;
- Provozní napětí [V]: 10...36 DC;
- Proudový odběr [mA]: < 20;
- Výstupní funkce: spínač / rozpínač programovatelný;

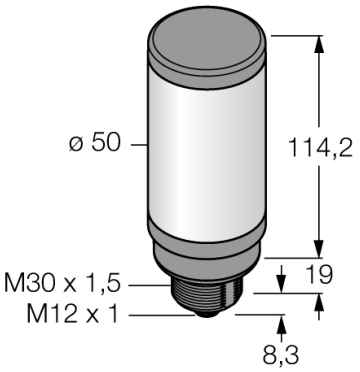
3.3.4. Inkrementální snímač

- Je použit inkrementální snímač pod označením IRC320/1024 od firmy LARM;
- Jedná se o snímač, který pracuje na základě světelné závory s LED;
- Vnější průměr hřídele je 10mm;
- Napájení je 10-30V v PNP provedení;
- Přesný popis najdete v datové listu (datasheetu);

3.4. Zobrazovače

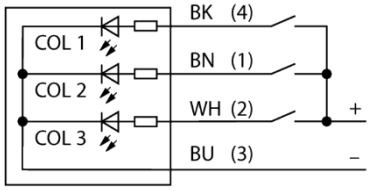
3.4.1. Světelný maják

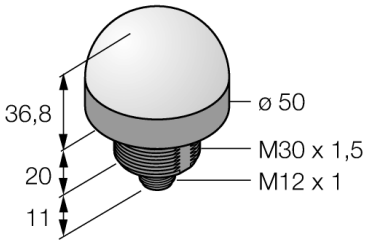




- LED signálka;
- lze nastavit;
- závit M30 x 1,5;
- stupeň krytí IP67;
- barvy: zelená (COL 1)
  červená (COL 2)
  žlutá (COL 3)
- konektor M12x1;
- napájecí napětí 18…30 VDC;

3.4.2. Poziční světlo





- LED signalizace kolem dokola;
- lze nastavit;
- závit M30 x 1,5;
- stupeň krytí IP67 / IP69K;
- barvy: modrá (COL 1)
  červená (COL 2)
  žlutá (COL 3)
- konektor M12x1;
- napájecí napětí 18…30 VDC;

3.4.3. Segment indikátor

- Housing Style: Flat;
- Primary Housing Material Polycarbonate;
- Feature: Multi-Color Segment Yes;
- Feature: Simultaneous Multi-Color Segment Yes;
- Supply Voltage 18-30 V dc;
- Color Option 1/Job Light Green;
- Color Option 2: Red;
- Color Option 3: Yellow;
- Color Option 4: Blue;
- Current - LED Color (mA) 35;
- Zapojení v obrázcích (str. 119): Obrázek 155: Zapojení čtyřbarevného indikátoru;

3.4.4. Denní viditelný indikátor

- Materiál: Polycarbonate;
- Feature: Multi-Color Segment: Yes;
- Napájecí napětí: 15-30 V dc;
- Barva 1: Zelená;
- Barva 2: Červená;
- Barva 3: Modrá;
- Proud na barvu (mA): 120;
- Zapojení v obrázcích (str. 119):Obrázek 154: Zapojení tříbarevného světla;

3.5. Ostatní

3.5.1. PC [4]

- Operační systém: Windows 10 64.bit;
- Procesor: Intel Cherry Trail Z8300 64-bit, 1,44 GHz (max. 1,84GHz);
- Grafika: Intel HD Graphics 8 gen.;
- Operační paměť: 2 GB DDR3L;
- FLASH: 32 GB eMMC;
- Video výstup: HDMI 1.4;
- Konektivita:
  - Dual Band WiFi 802.11abgn;
  - Bluetooth 4.0;
  - LAN 10/100;
  - Audio analogový výstup;
  - USB porty: 3 x USB 2.0 Host 1 x USB 3.0 Host;
  - Čtečka karet:
    - Ano, microSD slot (max. 64GB);
- Velikost: 15 cm x 12 cm x 3 cm;
- Hmotnost: 300 g;

3.5.2. Monitor [5]

- Displej o velikosti 21,5" a rozlišení 1920×1080 px;
- Technologie VA;
- Kontrast 20 000 000:1;
- Jas 250 cd/m²;
- Odezva 5 ms;
- Výstupy: HDMI, VGA;
- VESA kompatibilní;

- **Zdroj [6]**
- Vstupní napětí: 230V/50Hz;
- Výstupní napětí (DC): 24 V;
- Kalibrace: 21 - 26 V;
- Výstupní proud: 10 A;
- Výkon: 240 W;
- Rozměry [mm]: 111-51-199 mm;
- Pracovní teplota: -20 až 70 °C;
- Ochrany
  - proti přehřátí;
  - proti přepětí;
  - proti zkratu;
  - proti přetížení;
- Váha balení [kg]: 0.675 kg;

3.5.3. Svorkovnice

- Weidmuller;
- Wago 270(-560, -319, -417);

3.5.4. Relátka [7]

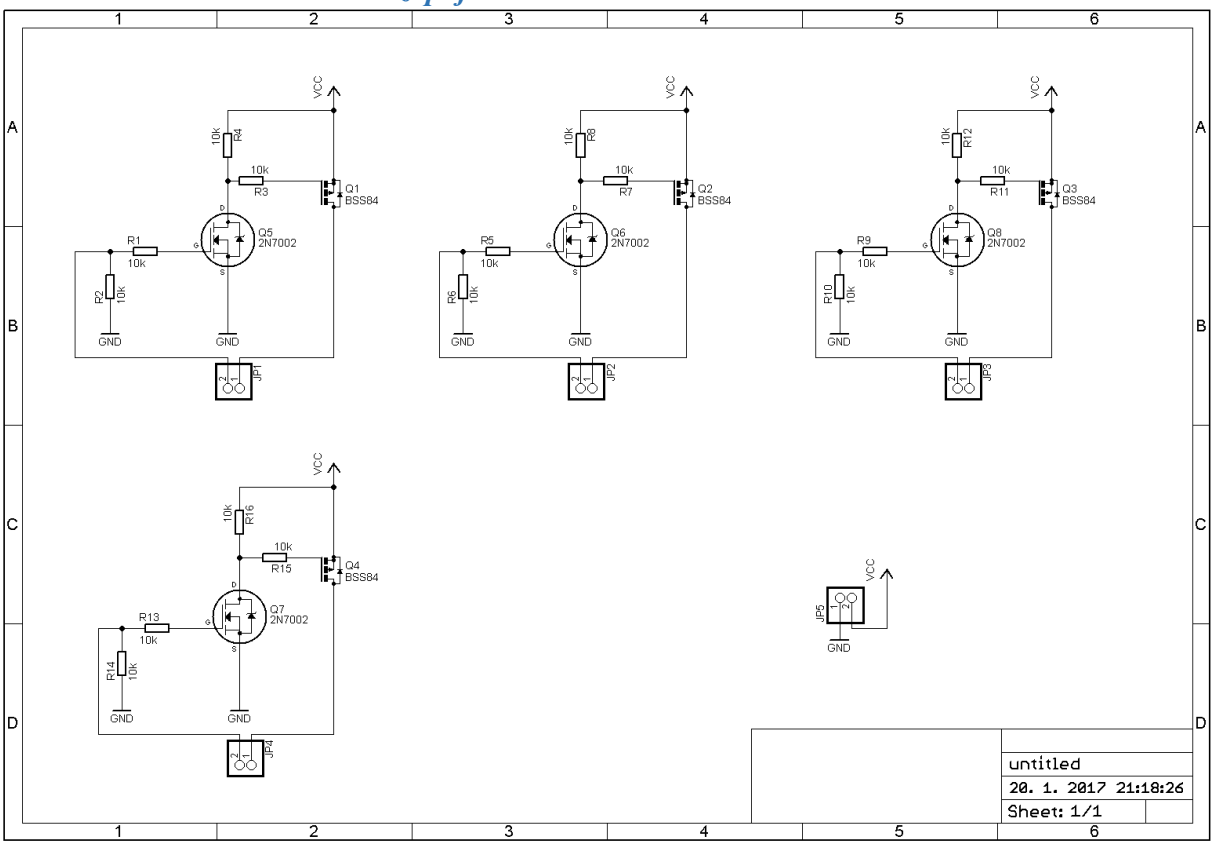
- WEIDMULLER 1122880000;

3.5.5. Měnič [8]

- 100W DC-DC conventor 10-32V to 12-60V;

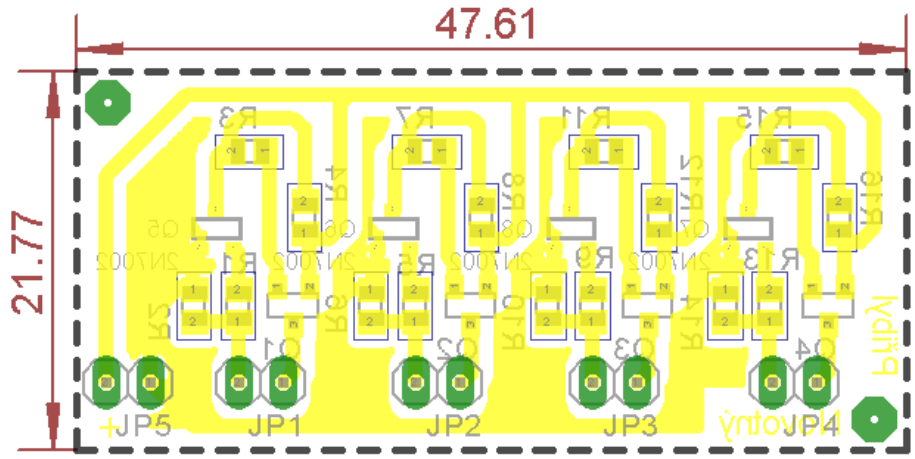
3.5.6. AO amplifier

3.5.6.1. *Schéma zapojení*



Obrázek 8: Schématické zapojení "AO amplifier"

3.5.6.2. *Plošný spoj*



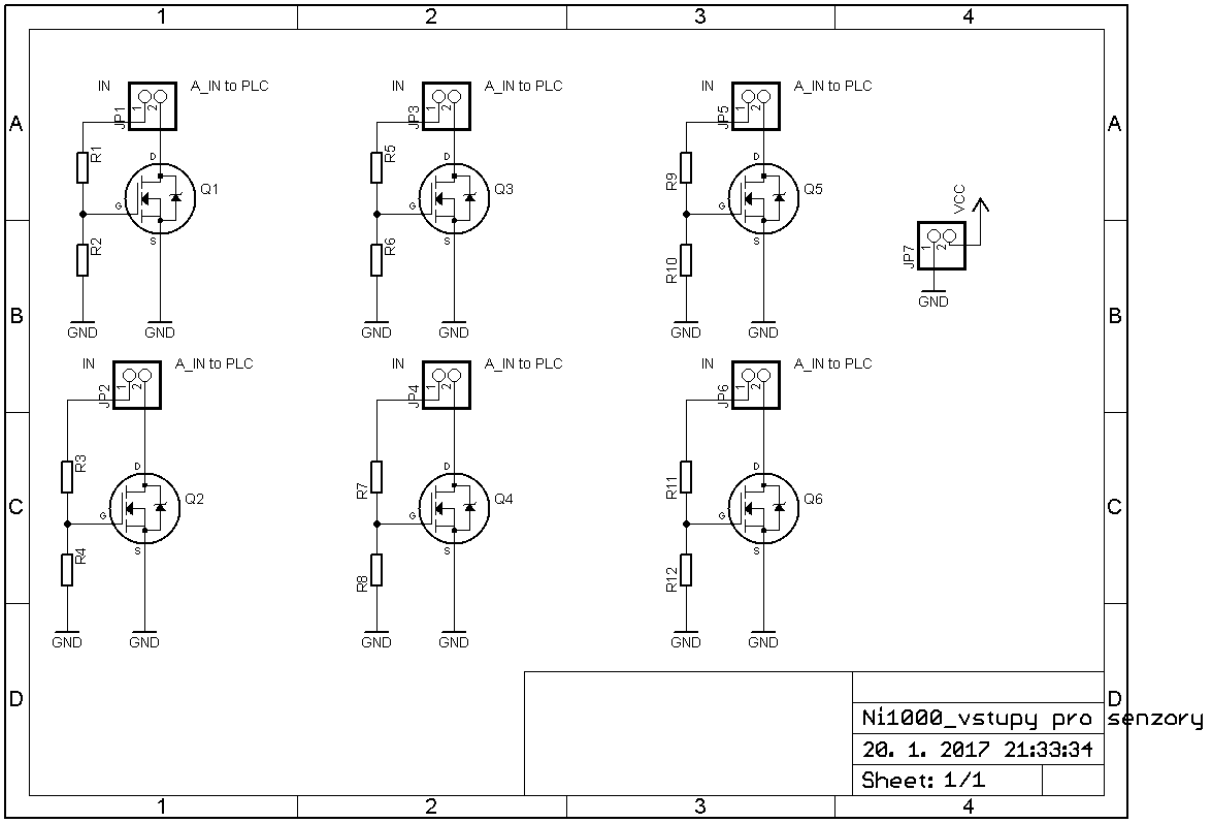
Obrázek 9: Návrh desky plošného spoje pro "AO amplifier"

3.5.6.3. *Popis*

PLC má 8 digitálních a 4 analogové výstupy. Jelikož 8 digitálních výstupů nestačilo a analogové výstupy mají maximálně 10V a 10mA, navrhli jsme zesilovač s mosfet tranzistory. Tyto výstupy byly použity pro zobrazovače, a kvůli jejich zapojení musí být použit mosfet s kanálem P. Pro spínání kanálu P byl použit ještě jeden mosfet kanálu N, který zajišťuje správné spínání tranzistoru. Tímto se dají použít i analogové výstupy na ovládání zobrazovačů.

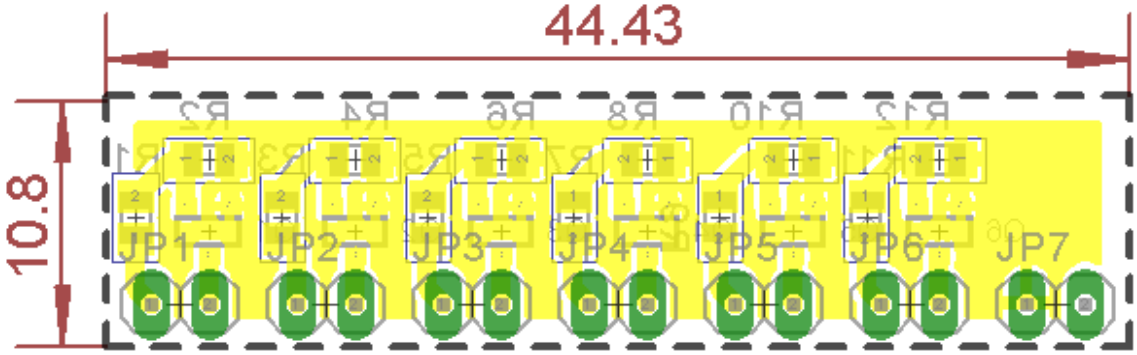
3.5.7. AI konvertor

3.5.7.1. *Schéma zapojení*



Obrázek 10: Schématické zapojení "AI konventoru"

3.5.7.2. *Plošný spoj*



Obrázek 11: Návrh desky plošného spoje pro "AI konventor"

3.5.7.3. *Popis*

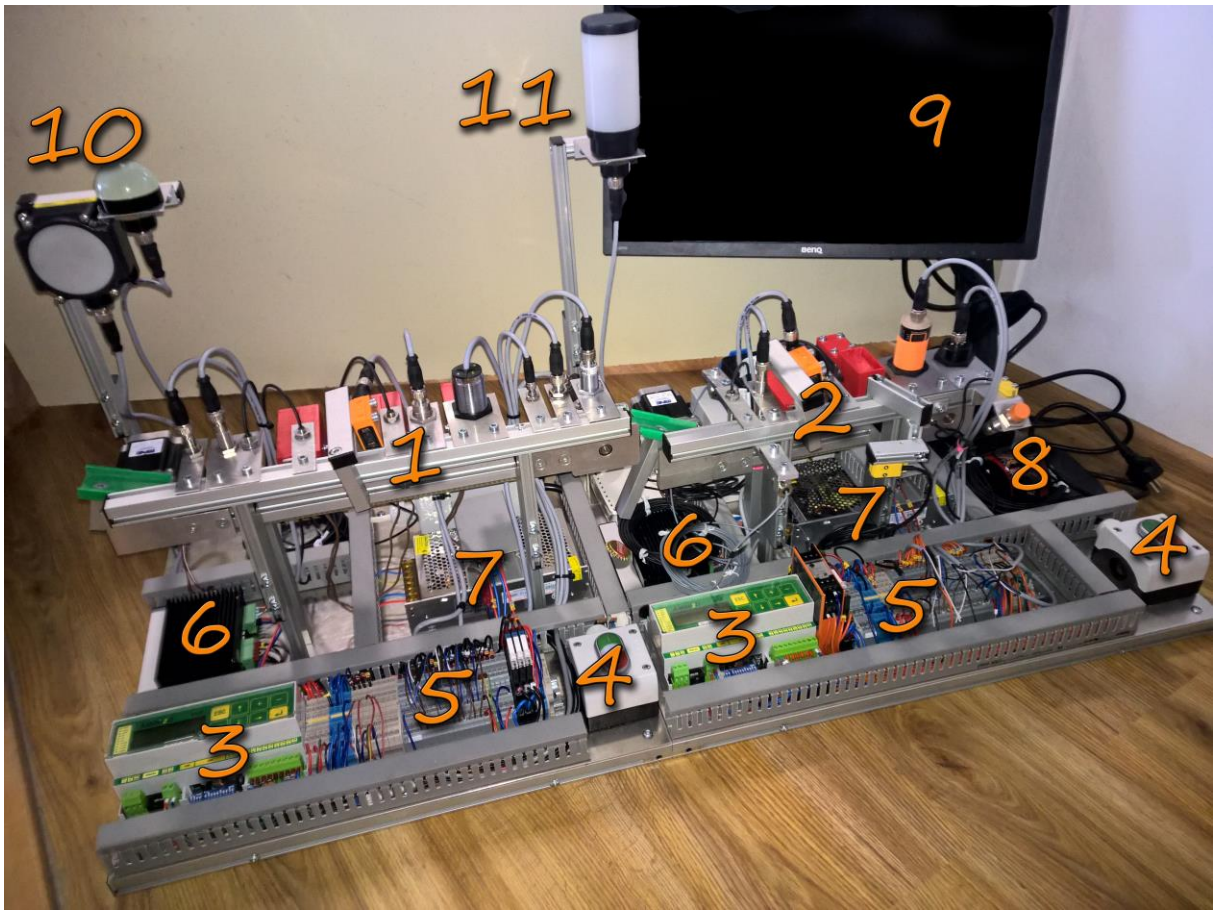
PLC má 6 analogových vstupů pro NI1000 a na lince žádný teplotní senzor není, proto bylo zapotřebí změnit tyto vstupy na digitální. Senzor NI1000 nahradí mosfet řízený výstupem senzoru. Tranzistor se bude chovat jako proměnný odpor a bude fungovat jako dělič napětí s vnitřním zapojením analogového vstupu pro NI1000.

4. Konstrukce

Obrázek 12: Jednotlivé dopravníky s motory

Obrázek 13: Linka v průběhu výroby

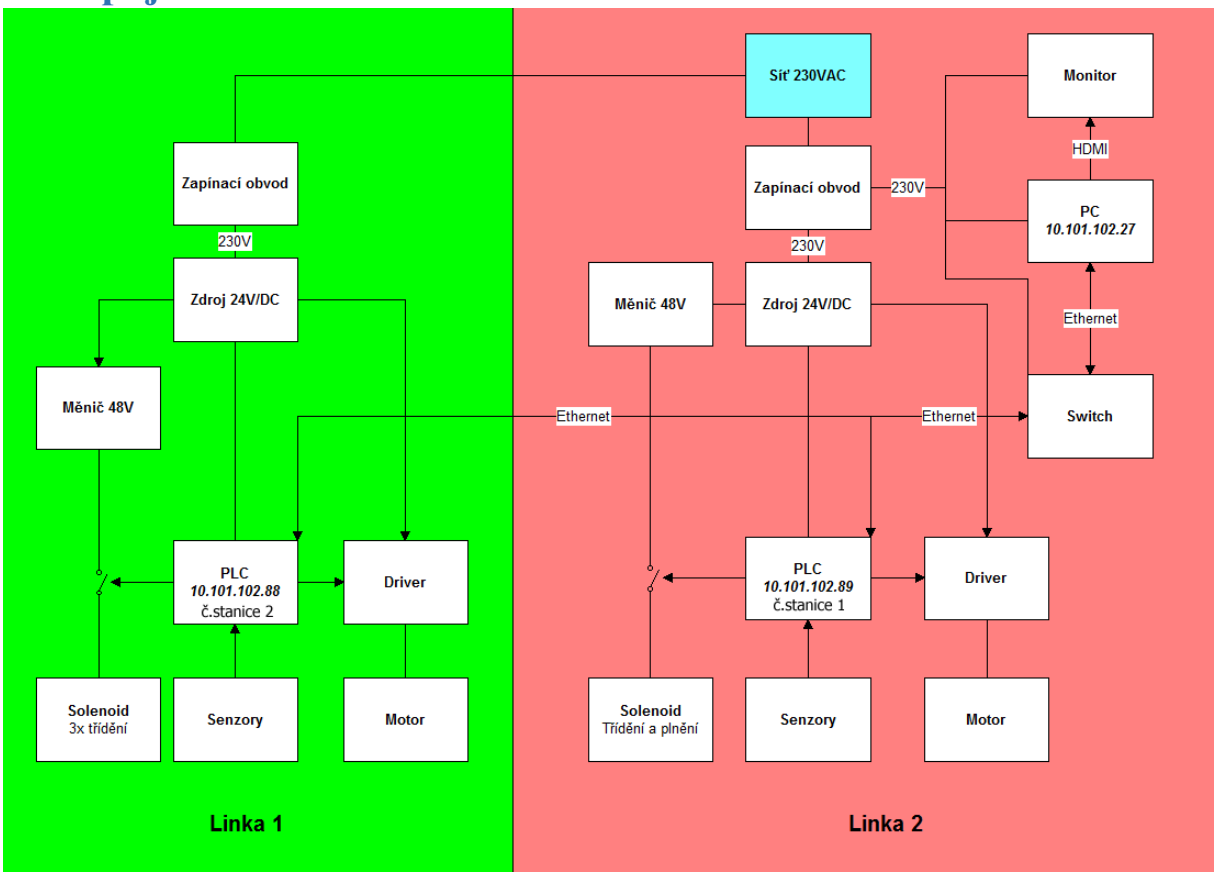
Bylo několik návrhů, jak linku vytvořit. Rozhodli jsme se, že použijeme hliník a hliníkové profily ALUTEC[13] drážky 6. Základ konstrukce tvoří rám pro zpevnění z profilu 21,5*21,5, na který je položen hliníková deska s tloušťkou 3 mm. Tato deska tvoří základ pro připevňování komponentů. Linky se spojují pomocí spojovacího materiálu na profily ALUTEC. Dále jsme použili profil 13*21,5 a 30*30. Od firmy ELZACO jsme dostali dopravníky, od kterých se následně odvíjela i konstrukce celé linky.



Obrázek 14: Jednotlivé části linky

1. *Linka 1*
2. *Linka 2*
3. *AMiNi2D (Řídící Systém)*
4. *Krabička dálkového ovládání pro zapínání silové části linky*
5. *Svorkovnice a komponenty na DIN lištu*
6. *Driver pro motor*
7. *Zdroj 24V/10A a měnič pro solenoidy (U_{IN}=24V, U_{OUT}=48V, P=100W, η=92%)*
8. *PC pro vizualizaci a programování*
9. *Monitor*
10. *Zobrazovače linky 1*
11. *Zobrazovače linky 2*

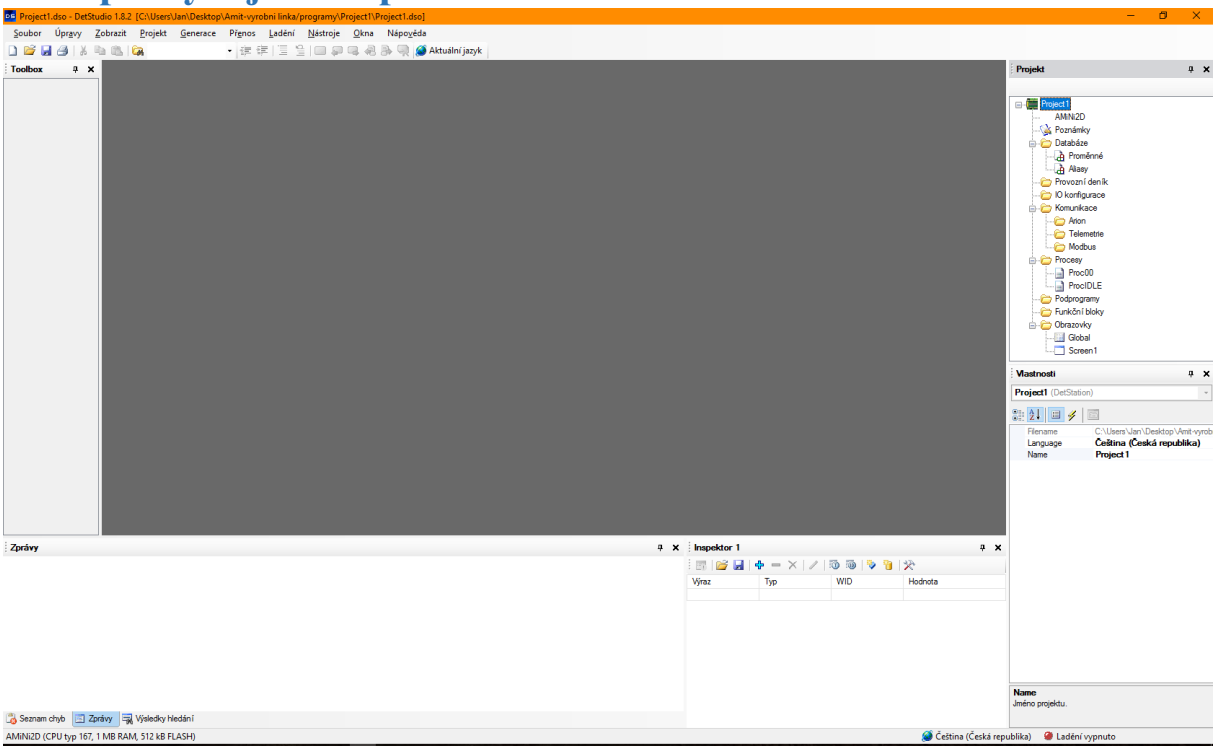
5. Zapojení linek



Obrázek 15: Zjednodušené blokové schéma zapojení obou linek

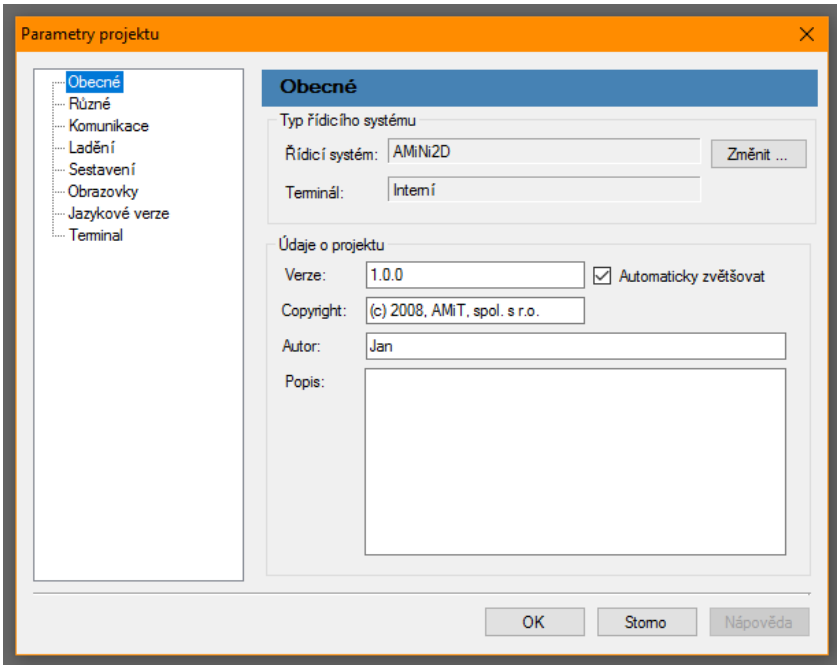
Podrobné zapojení linek najdete v kapitole Linka 1 (str.47) a Linka 2 (str. 77)

6. Popis vývojového prostředí DetStudio



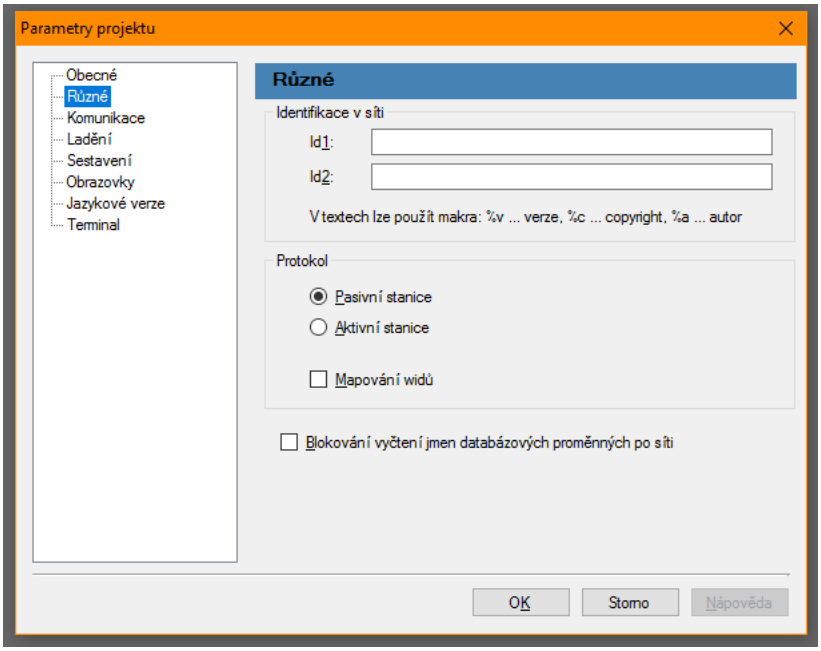
Obrázek 16: Prostředí det studio při vytvoření nového projektu

Při vytvoření nového projektu se nám zobrazí okno s možnostmi. Kdykoli později je možné je změnit.



Obrázek 17: Parametry projektu-obecné

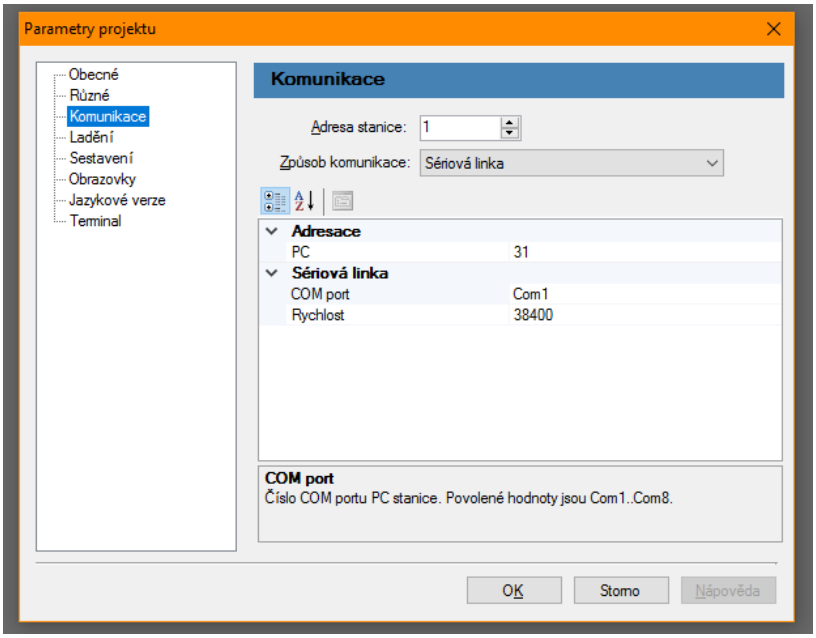
Nastavují se zde základní informace o projektu. Jejich nastavení není omezující software, jelikož jsou zde uloženy jen informace o projektu, jako je řídící systém s terminálem, verzi projektu, Copyright, autora a popis projektu.



Obrázek 18: Parametry projektu-různé

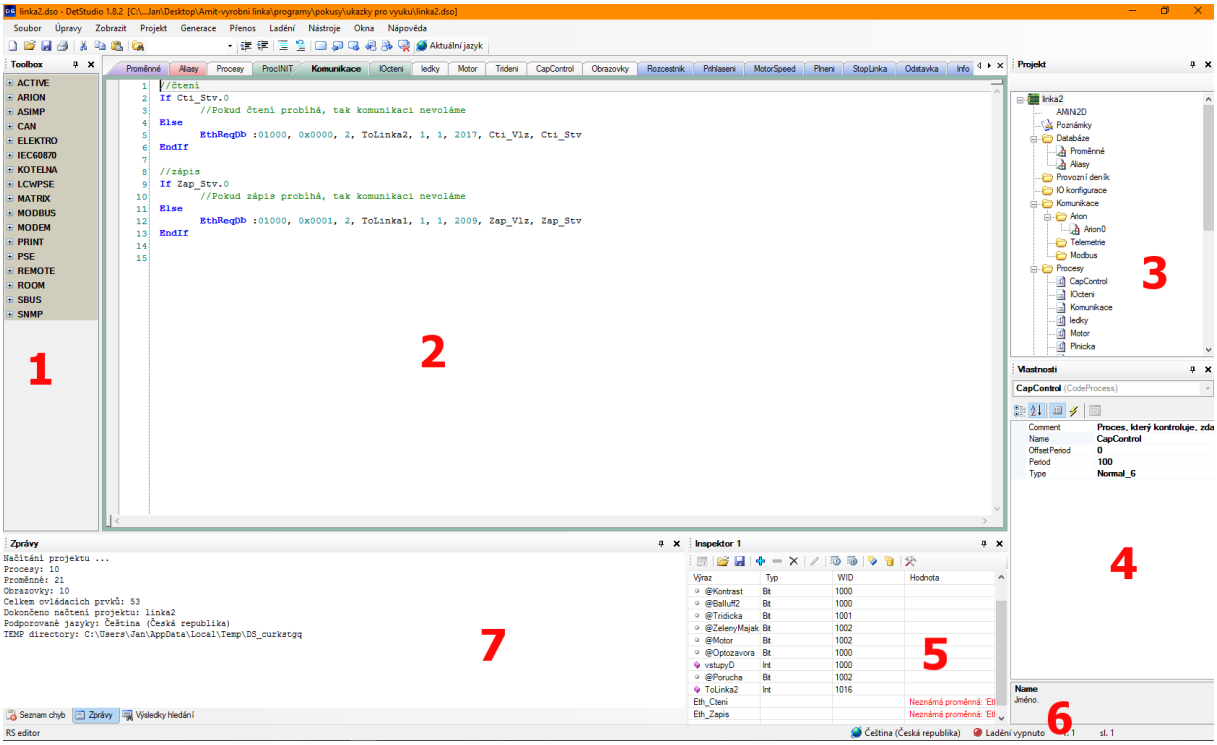
Zde můžeme nastavit, jak se bude řídící systém chovat v síti. Id1 a Id2 je identifikace v síti, kdy se nám řídící systém bude hlásit pod těmito názvy. Viditelný je při identifikace stanice.

V protokolu můžeme nastavit, zda bude stanice aktivní či pasivní. Pokud je stanice aktivní, tak komunikuje s ostatními a vytváří začátek komunikací a pasivní stanice jim pouze odpovídají.



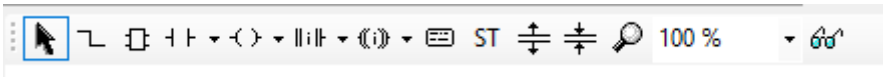
Obrázek 19: Parametry projektu-Komunikace

Zde nastavíme adresu stanice, v našem případě 1 nebo 2, způsob komunikace (Ethernet je nejrychlejší možnost) a parametry komunikace.



Obrázek 20: Otevřený projekt se strukturovaným textem

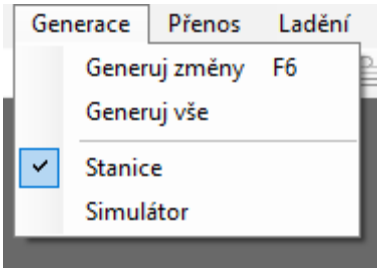
1) Okno s možnými příkazy
2) Okno s vlastním programem
3) Věci v projektu (proměnné, aliasy, obrazovky, procesy, …)
4) Vlastnosti jednotlivých modulů a objektů
5) Inspektor
6) Indikace, zda je aktivní inspektor
7) Stavový řádek (zprávy, seznam chyb, výsledky hledání)



Obrázek 21: Nástroje reléového schématu

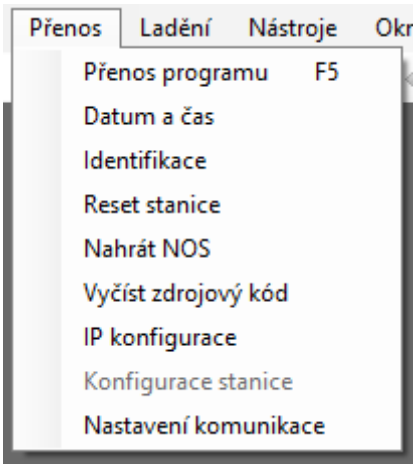
V nástrojích, které nabízí reléové schéma, najdeme normální ukazovatel, čáru, vkládání modulů, kontaktů, cívek, proměnných, strukturovaného textu. Poté je zde lupa a ladění.

!Pozor: Při používání ladění je nutné, aby byl zapnutý inspektor ladění. (F8)



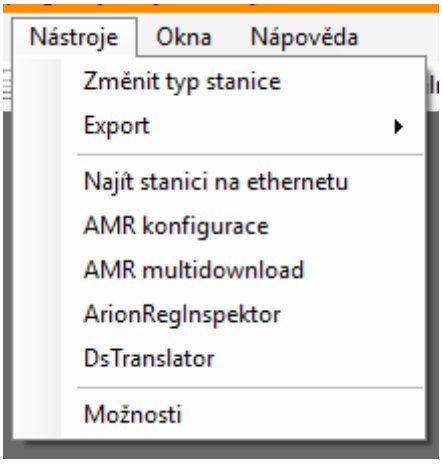
Obrázek 22: Záložka generace

Před nahráním programu je nutné generovat kód pro PLC, který se bude přenášet. Jedná se o kompilaci projektu.

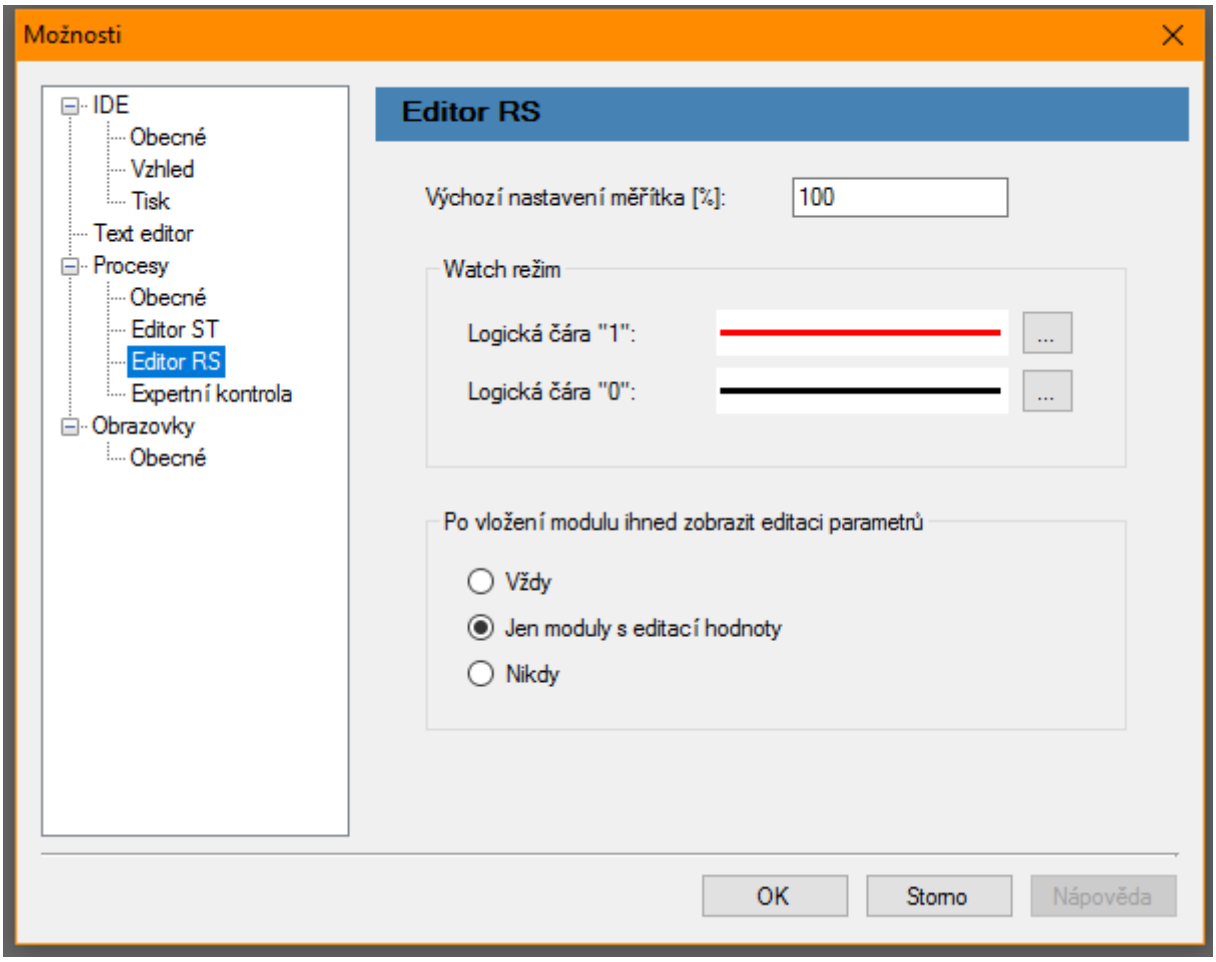


Obrázek 23: Záložka přenos

Po generaci je možné program nahrát. Důležitá je i verze NOSu. Pokud je v řídícím systému zastaralá verze, tak DetStudio upozorní na starší verzi. Průvodce nahrání NOSu vás provede její instalací.

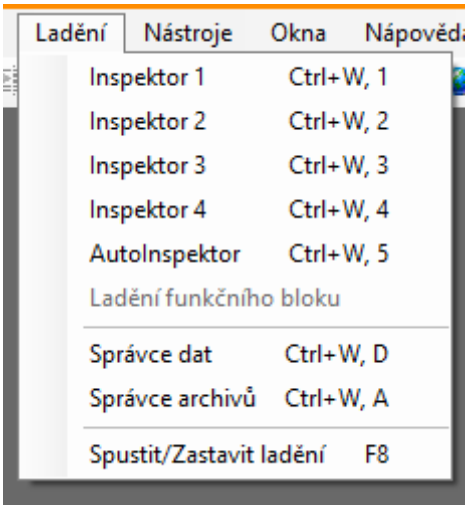


Obrázek 24: Záložka nástroje



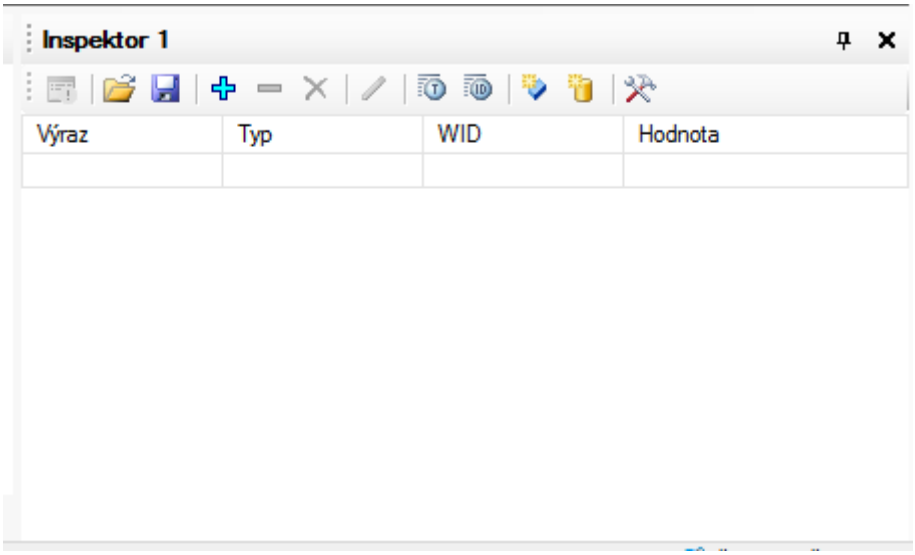
Obrázek 25: Okno možnosti-Editor RS

V možnostech změníme nastavení editoru RS. Pokud budeme chtít provádět ladění v RS, tak všechny čáry jsou černé a špatně se v něm orientuje. Je možné, že i ve starších verzích DetStudia se čáry nezobrazují. Pokud nastavíme na Log.1 červenou a na Log.0 černou, tak ladění bude mnohem přehlednější.



Obrázek 26: Záložka ladění

Zde můžeme vkládat inspektory, do kterých vložíme proměnné či aliasy, které chceme sledovat.

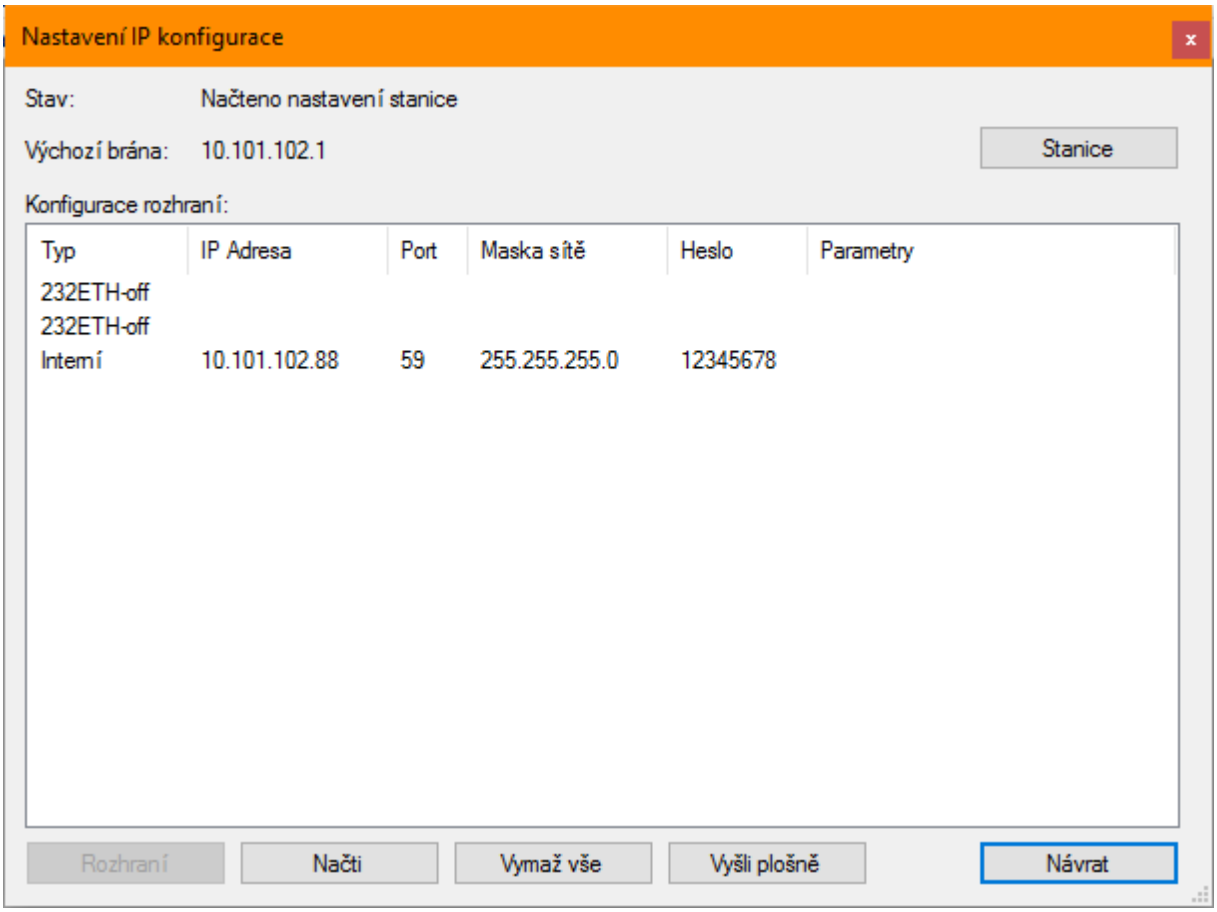


Obrázek 27: Inspektor 1

Do inspektora můžeme vkládat proměnné a aliasy, které neuvidíme třeba v ladění RS. Vložíme je poklepáním do okna výraz a následně vybereme proměnnou. Proměnné můžeme i přepisovat. Pokud je ale v procesu přepis proměnné, tak je vždy proměnná přepsána řídícím systémem a pro ladění je nutné toto přepisování zastavit, pokud ji chceme přepsat ručně a sledovat, co se děje.

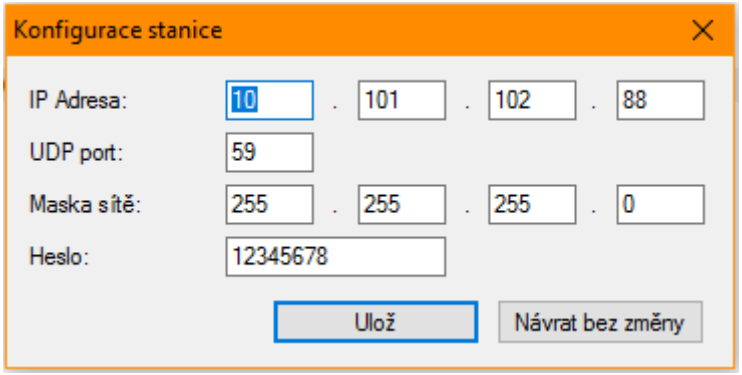
7. Konfigurace IP adresy PLC

Pro nahrávání programu do řídícího systému pomocí Ethernetu je zapotřebí nastavit IP adresu řídícího systému. Přednastavili jsme již IP adresy stanicím 10.101.102.88(89). Nastavení se provádí při připojení přes RS323. Zvolíme „Přenos“ a „IP konfigurace“. Následující okno bude vypadat následovně:



Obrázek 28: IP konfigurace stanice

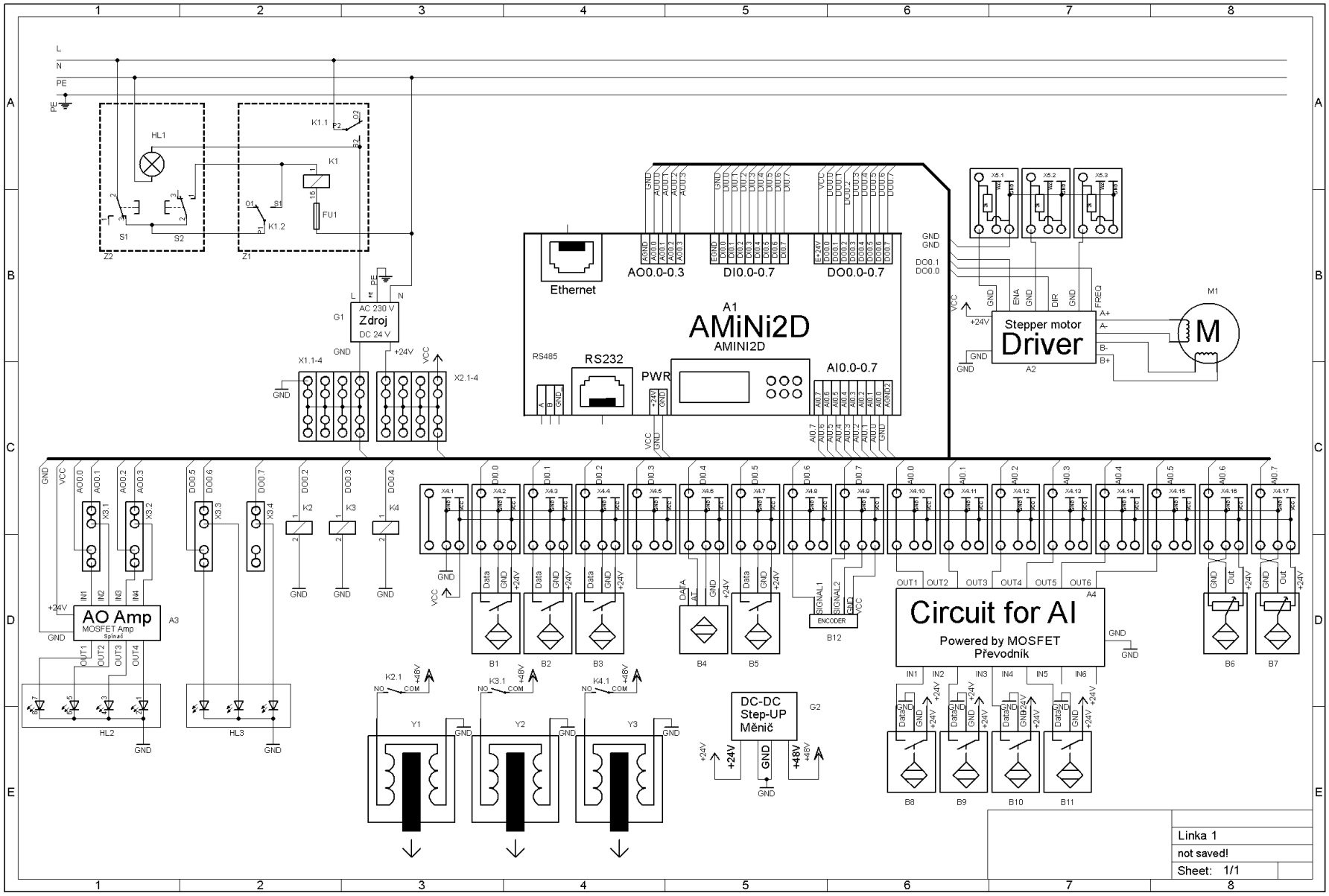
Rozklikneme požadovanou stanici a zobrazí se nastavovací okno:



Obrázek 29: Nastavení adresy stanice

Nastavíme požadovanou IP adresu, port, masku a heslo.

8. Linka 1



Obrázek 30: Schéma zapojení

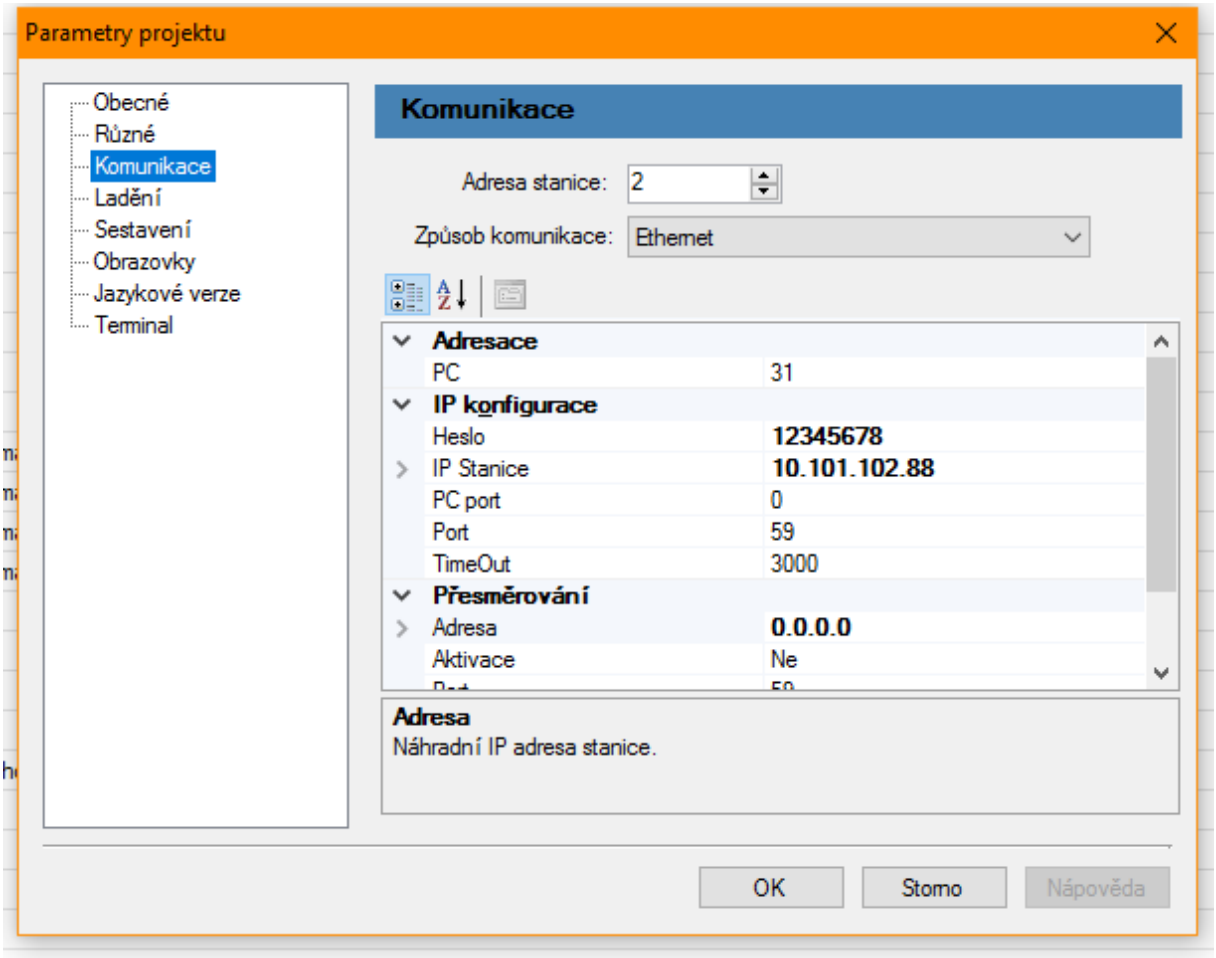
8.1. Vstupy, výstupy a použité senzory

PLC AMiNi2D				
Vstup	**Označení**	**Druh**	**Připojeno**	**Komentář**
AO0.0	HL2	Analogový výstup	Čtyř barevný indikátor (žlutá)	Digitální výstup realizován vnějším obvodem
AO0.1	HL2	Analogový výstup	Čtyř barevný indikátor (zelená)	Digitální výstup realizován vnějším obvodem
AO0.2	HL2	Analogový výstup	Čtyř barevný indikátor (modrá)	Digitální výstup realizován vnějším obvodem
AO0.3	HL2	Analogový výstup	Čtyř barevný indikátor (červená)	Digitální výstup realizován vnějším obvodem
DO0.0	A2	Digitální výstup	Motor (rychlost)	Řízení rychlosti motoru
DO0.1	A2	Digitální výstup	Motor (směr)	Řízení směru motoru
DO0.2	Y1	Digitální výstup	Solenoid 1	
DO0.3	Y2	Digitální výstup	Solenoid 2	
DO0.4	Y3	Digitální výstup	Solenoid 3	
DO0.5	HL3	Digitální výstup	Tříbarevný zobrazovač (zelená)	
DO0.6	HL3	Digitální výstup	Tříbarevný zobrazovač (žlutá)	
DO0.7	HL3	Digitální výstup	Tříbarevný zobrazovač (červená)	
DI0.0	B1	Digitální vstup	KF5001	
DI0.1	B2	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.2	B3	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.3	B4	Digitální vstup	O5C500 (výstup)	
DI0.4	B4	Digitální vstup	O5C500 (AT)	
DI0.5	B5	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.6	B12	Digitální vstup	Enkodér	
DI0.7	B12	Digitální vstup	Enkodér	
AI0.0	B8	Analogový vstup	BI15U-M30-AP6X	Digitální senzor na analogovém vstupu
AI0.1	B9	Analogový vstup	BI2-EG08-AP6X-V1131	Digitální senzor na analogovém vstupu
AI0.2	B10	Analogový vstup	BI4U-M12-AP6X-H1141	Digitální senzor na analogovém vstupu
AI0.3	B11	Analogový vstup	BES M18MG1-PSC12B-S04G	Digitální senzor na analogovém vstupu
AI0.4		Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.5		Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.6	B6	Analogový vstup	IF6028	
AI0.7	B7	Analogový vstup	IG6086	

Tabulka 1: Zapojení vstupů a výstupů řídícího systému na lince 1

8.2. Nastavení komunikace

Pro rychlé nahrávání programů je zvolen Ethernet (asi 6x rychlejší, nežli pře RS232), ale je nutné správné nastavení.



Obrázek 31: Parametrizace komunikace linky 1

- Zvolit adresu stanice – v našem případě 2
- Způsob komunikace – Ethernet
- Heslo – stanice má zprvopočátku heslo 12345678, ale jde změnit (viz. Nastavení IP adresy PLC – str.45)

IP stanice – stanici jsme nadefinovali, že má IP adresu 10.101.102.88

8.3. Výukové úlohy

8.3.1. Úloha 1.1: Čtení digitálních snímačů (vstupů)

8.3.1.1. Cíl

Naučit se číst digitální vstupy, na které jsou připojeny senzory s digitálním výstupem a následně přečtená data zobrazit.

8.3.1.2. Příklad

Navrhněte program pro čtení digitálních snímačů, to jsou vstupy DI.0-7 a AI.0-5, a hodnoty 4 libovolně zvolených snímačů zobrazte na terminálu AMiNi2D. Můžete použít seznam použitých snímačů viz strana 26

! Pozor: u AI0.0-5 čtení analogových vstupů jako digitálních

! Pozor: u AI0.0-5 je zapotřebí invertovat vstupy

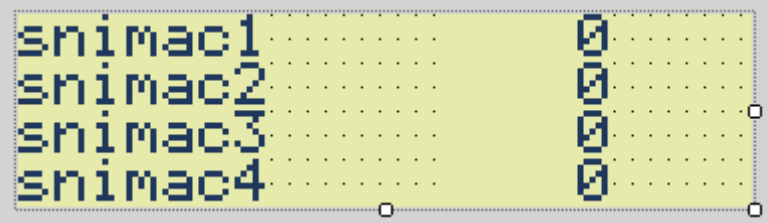
8.3.1.3. Realizace

Digitální vstupy jsou připojeny napřímo a vstupy AI0.0-5 jsou připojeny přes inventor (str.35), který obrací vstup, takže je zapotřebí číst tyto vstupy invertovaně.

8.3.1.4. Program

```
DigIn #0, Digital_in, 0x0000
DigIn #2, Analog_in, 0x000F
DigIn #0, vstupyD, 0x0000
```

DigIN načte vstupy na kanále 0 (DI0.0.7) a uloží je do proměnné „vstupyD“



Obrázek 32: Displej pro zobrazování senzorů

Vytvoříte obrazovku podle předlohy a můžete doplnit na jakém vstupu je připojen. Na holý text se používá „Label“ a pro zobrazení hodnoty „Numeric View“. Po vybrání zobrazované proměnné se zvolí i formát zobrazování. Jelikož budeme zobrazovat pouze bit (proměnnou bool), stačí jeden znak.

8.3.2. Úloha 1.2: Konfigurace a čtení analogových snímačů

8.3.2.1. Cíl

Naučit se správně nakonfigurovat vstupy řídícího systému a následně číst a zpracovávat přečtená data a zobrazovat je.

8.3.2.2. Příklad

Navrhněte program pro čtení analogových snímačů, to jsou vstupy AI.6-7 a hodnoty zobrazte na terminálu AMiNi2D. Můžete použít seznam použitých snímačů viz strana 26.

! Pozor: správné zvolení typu proměnné
! Pozor: převod měřené hodnoty na fyzikální jednotky

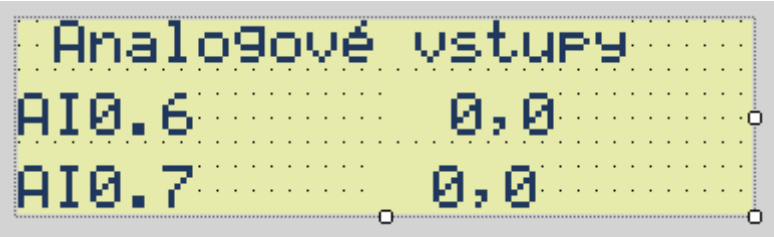
8.3.2.3. Realizace

Analogové senzory jsou připojeny na vstupy AI0.6 a AI0.7. Tyto vstupy jsou nastaveny jako proudové 4-20mA.

8.3.2.4. Program

```
AnIn #AI00_6, AI6, 10.000, 0.000, 10.000, 0.000, 100.000
AnIn #AI00_7, AI7, 10.000, 0.000, 10.000, 0.000, 100.000
```

Příkazy AnIn převádějí analogový vstup na fyzikální veličinu. Pokud budeme chtít pracovat s napětím, tak nadefinujeme ElMin na 0 a ElMax na 10, jelikož měříme napětí 0-10V. Toto nastavení je nadefinováno výrobcem senzoru. Mohou být senzory 2-8V a je to potřebí nastavit kvůli převodníku). PhysMin nastavíme na 0 a PhysMax na 10. Tyto parametry jsou dány pro převod z elektrické veličiny na fyzikální. My si však ponecháme napětí 0-10V. V případě, že bychom chtěli převádět napětí na vzdálenost, tak nastavíme podle měřené vzdálenosti senzoru. Napětí je lineárně závislé na vzdálenosti s tím, že senzor nezměří nulovou vzdálenost. V případě senzoru IG6084 nastavíme PhysMin na 0,8 a PhysMax na 8. Výstupem bude proměnná typu Float.



Obrázek 33: Displej pro zobrazování analogových vstupů

Podle obrázku vytvoříme obrazovku. Text je vložen pomocí „Label“ a proměnné pomocí „Numeric View“.

8.3.3. Úloha 1.3: Řízení solenoidů

8.3.3.1. Cíl

Naučit se řídit digitální výstupy a požívat časovače.

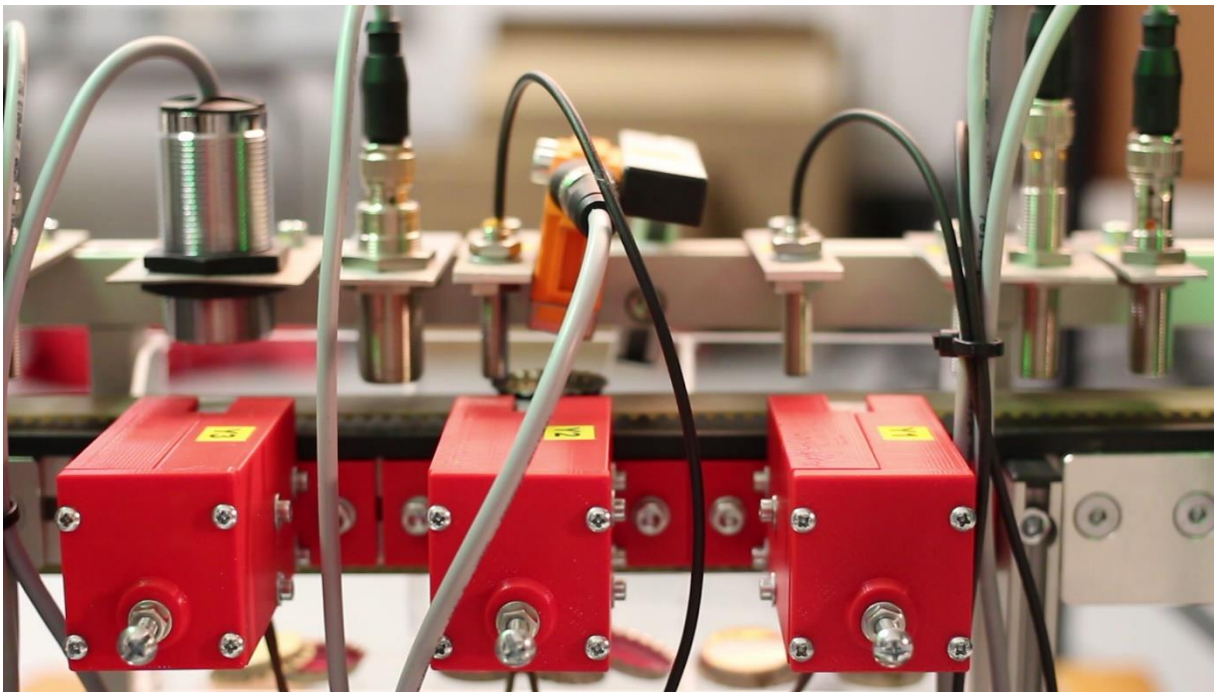
8.3.3.2. Příklad

Navrhněte program pro řízení solenoidů tak, abyste je mohli ovládat pomocí tlačítek na terminálu AMiNi2D, a tak aby solenoid po stisknutí tlačítka nezůstal ve vysunuté poloze, ale aby se vrátil do polohy klidové

! Pozor: dlouhé buzení solenoidů může vést k jejich zničení

8.3.3.3. Realizace

Solenoidy Y1, Y2 a Y3 jsou připojeny na výstupech DO0.2-4. Jelikož jsou připojeny na 48V spínány pomocí relátka. Solenoidy jsou vysunuté pouze tehdy, když je na nich napětí a vracejí se pomocí pružiny.



Obrázek 34: Solenoidy na lince

8.3.3.4. Program

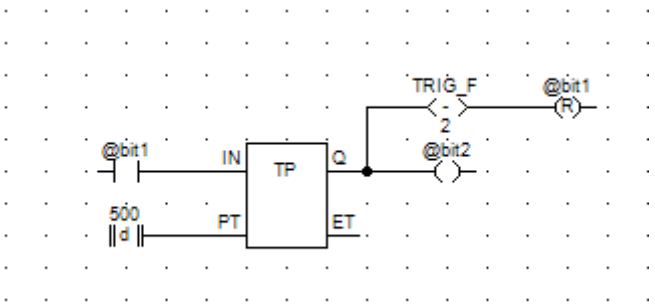
Pro ovládání pomocí terminálu je zapotřebí vytvořit obrazovku, kam se budou vkládat příkazy pro ovládání. Tyto příkazy by se mohly dát vložit i do obrazovky global, ale tato tlačítka by byla již pak nepoužitelná pro další obrazovky, jelikož obrazovka global běží neustále.

Pro ovládání solenoidů použijeme příkaz „KeyBit“

(Name)	**Plnicka**
Enabled	**True**
KeyCode	**Up**
Mode	**Set**
Permissions	**All**
Variable	**@bit 1**

Obrázek 35: Parametrizace objektu "KeyBit"

Při parametrizaci „KeyBit“ nastavíme, že má uložit do bitu „bit1“ log.1, se kterou následně pracujeme.



Obrázek 36: Řízení solenoidu

Pokud je tedy @bit1 log.1, tak se spustí časovač „TP“, který je nastaven na 500ms. Po tuto dobu má @bit2 logickou úroveň 1 a je tedy solenoid vysunutý. Po uplynutí doby 500ms je vytvořen se sestupnou hranou impulz, který vyresetuje (nastaví log.0) @bit1, aby mohlo dojít k dalšímu časování stiskem tlačítka, jinak by byl na vstupu časovače log.1 a časovač by tak nezačal časovat nikdy.

8.3.4. Úloha 2.1: Tříbarevný zobrazovač

8.3.4.1. Cíl

Naučit se obsluhovat tříbarevný zobrazovač.

8.3.4.2. Příklad

Navrhněte program pro tříbarevný zobrazovač HL3, to jsou výstupy DO.5-7, tak aby se postupně rozsvítila každá barva. Můžete použít libovolný snímač, nebo tlačítka řídicího systému jako spouštění libovolné barvy.

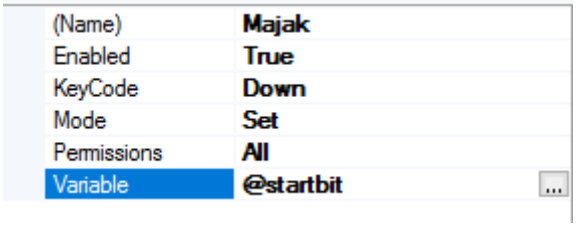
! Pozor: zobrazovač není schopen zobrazit více barev najednou

8.3.4.3. Realizace

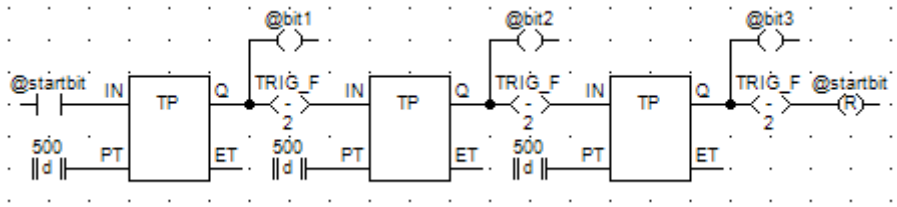
Maják je připojen na digitální výstupy DO0.0-3.

8.3.4.4. Program

Použijeme příkaz „KeyBit“, pro zápis hodnoty do bitu, kterým budeme spouštět sekvenci a nastavíme ho tak, aby při stisku setoval (nastavil log.1) na bit @startbit. Později se tento @startbit vyresetuje softwarově, aby nedocházelo, že se cykly budou spouštět několikrát.



Obrázek 37: Parametrizace objektu „KeyBit“



Obrázek 38: Řízení zobrazovače

@startbit zapne časování čítače „TP“, který je nastaven na dobu 500ms. Po tuto dobu bude na výstupu „TP“ log.1 a bude v bitu @bit1 uložena. Po tom, co první časovač „TP“ vypne, tak se změní @bit1 na log.0 a „TRIG_F“ nám vytvoří hranu se seběžnou hranou prvního čítače „TP“. Tento impulz nám nastaví čítání druhého čítače. A takto to pokračuje. Celý cyklus se nesmí spustit znovu, dokud nedoběhne, jelikož může svítit pouze jeden segment majáčku. Pokud je tedy @startbit nastaven do log.1, tak se první časovač nespustí znovu. Až dočasuje třetí čítač, tak s jeho seběžnou hranou je generován impulz, který resetuje bit @startbit a je možné pak spustit cyklus znovu. Pro přehlednost je možné spojit cyklus a zápis do proměnné pro „AnOut“.

8.3.5. Úloha 2.2: Čtyřbarevný zobrazovač

8.3.5.1. Cíl

Naučit se pracovat s analogovými výstupy.

8.3.5.2. Příklad

Navrhněte program pro řízení čtyřbarevného zobrazovače HL2, to jsou výstupy AO.0-3, můžete použít snímače, nebo tlačítka řídicího systému pro spouštění barev. Zobrazovač je schopen rozsvítit všechny barvy zároveň.

! Pozor: řídíme digitální prvek pomocí analogových výstupů

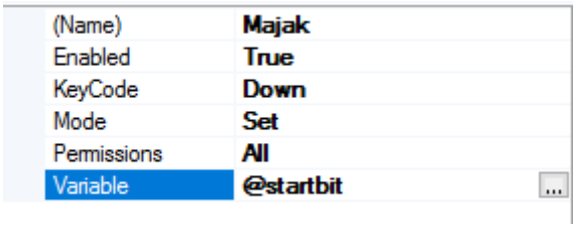
! Pozor: výstup je převodem z fyzikální veličiny

8.3.5.3. Realizace

Zobrazovač je připojen na výstupy AO0.0-3, které jsou převedeny na dig. podobu (viz AO amplifier str.34).

8.3.5.4. Program

Spuštění cyklu pro řízení barev budeme realizovat podobně jak v minulé úloze. Použijeme příkaz „KeyBit“ a nastavíme ho tak, aby při stisku setoval (nastavil log.1) na bit @startbit. Později se tento @startbit vyresetuje softwarově aby nedocházelo k tomu, že se cykly budou spouštět několikrát.

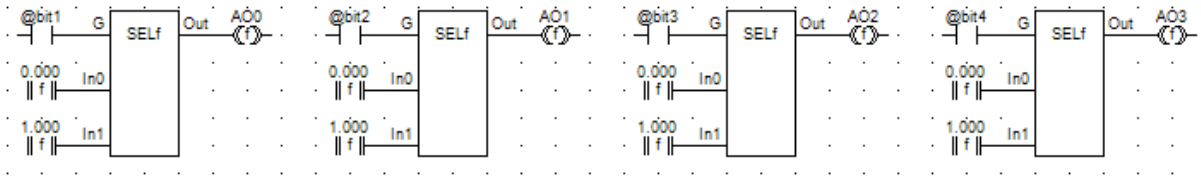


Obrázek 39: Parametrizace objektu "KeyBit"

Jelikož zapisujeme do AO, tak musíme použít příkaz AnOut. Buď použijeme AnOutLA pro RS (reléové schéma), anebo AnOut pro ST (strukturovaný text). V našem případě budeme mít dva procesy (RS – řízení, ST – zápis). AnOut převede fyzikální jednotku na elektrickou a zapíše ji na výstup. Zpracovává data z proměnné, ve které je hodnota, která se má zapsat.

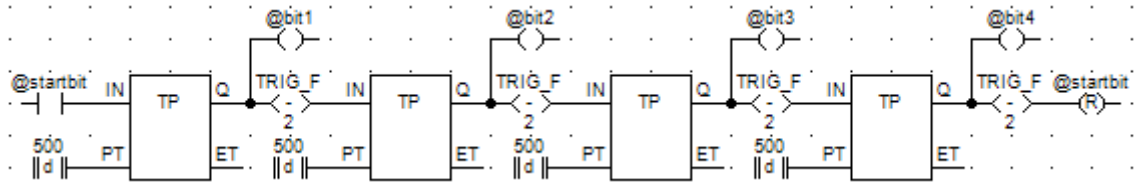
```
AnOut #AO00_0, yellow, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_1, green, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_2, blue, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_3, red, 10.000, 0.000, 10.000, 0.000, 1.000
```

Analogový výstup je realizován pomocí PWM, ale my si postačíme s MIN a MAX (0% a 100% střídy). Jelikož z cyklu pro zapínání barev, budeme mít digitální hodnotu, ale AnOut pracuje s analogovou hodnotou, použijeme modul SELf, který vybere jednu ze dvou hodnot pomocí řídícího bitu a uloží ji do proměnné, kterou si přečte AnOut.



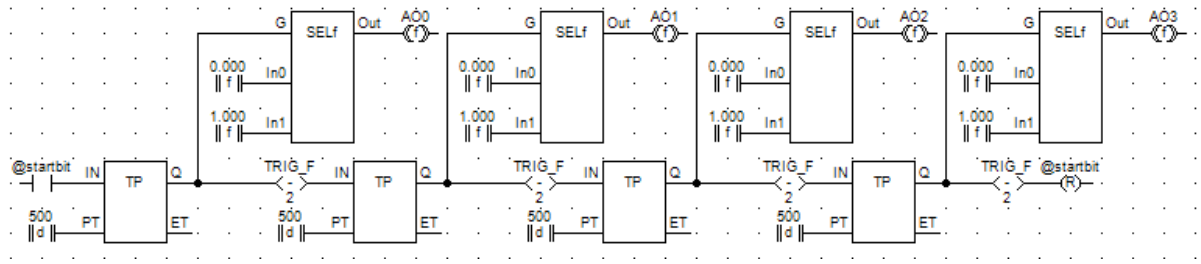
Obrázek 40: Zápis hodnoty do proměnné Float

Následně vytvoříme cyklus, který bude jednotlivé barvy spouštět, tedy bude přepínat hodnoty proměnných pro AnOut pomocí SELf.

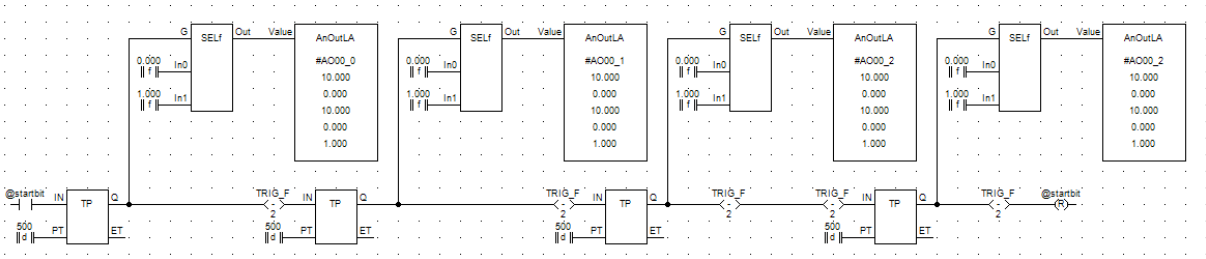


Obrázek 41: Cyklus řízení barev

@startbit zapne časování čítače „TP“, který je nastaven na dobu 500ms. Po tuto dobu bude na výstupu „TP“ log.1 a bude v bitu @bit1 uložena. Po tom, co první časovač „TP“ vypne, tak se změní @bit1 na log.0 a „TRIG_F“ nám vytvoří hranu se seběžnou hranou prvního čítače „TP“. Tento impulz nám nastaví čítání druhého čítače. A takto to pokračuje. Celý cyklus se nesmí spustit znovu, dokud nedoběhne, jelikož může svítit pouze jeden segment majáčku. Pokud je tedy @startbit nastaven do log.1, tak se první časovač nespustí znovu. Až dočasuje třetí čítač, tak s jeho seběžnou hranou je generován impulz, který resetuje bit @startbit a je možné pak spustit cyklus znovu. Pro přehlednost je možné spojit cyklus a zápis do proměnné pro „AnOut“.



Obrázek 42: Program spojený



Obrázek 43: Řízení celého majáku

Řízení celého majáku lze i docílit pomocí *jednoho pomocného bitu*, kde zapisování do proměnné nahradíme přímo zápisem do výstupu.

8.3.6. Úloha 3: Řízení krokového motoru

8.3.6.1. Cíl

Řídit krokový motor pomocí budiče a vyzkoušet si změnu směru otáčení a rychlost.

8.3.6.2. Příklad

Navrhněte program pro řízení krokového motoru M1 pomocí driveru A2, to jsou výstupy DO.0-1, vyzkoušejte si měnit rychlost, směr otáčení a zobrazte údaje na terminálu.

? Nápověda: modul „FreqOut“ tvoří frekvenci na výstupu digitálního výstupu

! Pozor: motor je řízen pomocí driveru, který je řízen frekvencí

! Pozor: maximální použitelná frekvence, kterou může AMiNi2D poskytnout, je 1500 Hz

8.3.6.3. Realizace

Motor je připojen pomocí budiče (Driveru), který řídí rychlost motoru v závislosti na frekvenci. Budič podporuje i mikrokrokování, které je nutné, jinak motor vibruje a jelikož jsou vršky velmi lehké, tak začnou všelijak jezdit po páse. Mikrokrokování a proud motorem se nastavuje přímo na driveru pomocí kódových přepínačů.

8.3.6.4. Program

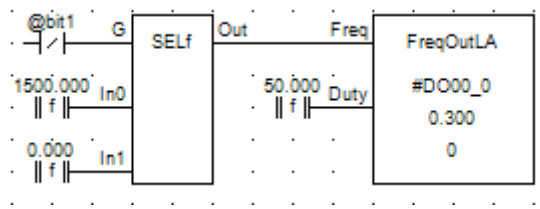
Vytvoříme obrazovku, ve které budou „KeyBit“ pro řízení. Jeden bude řídit směr, druhý brzdu a třetí rychlost. Pro „KeyBit“ nastavíme funkci toggle, abychom mohli pozorovat chování motoru v různých situacích.

(Name)	**Motor**
Enabled	**True**
KeyCode	**Plus**
Mode	**Toggle**
Permissions	**All**
Variable	**@bit 1**

Obrázek 44: Parametrizace "KeyBit"

Řízení směru a brzdy měníme bit přímo v proměnné pro zápis digitálních výstupů.

Pomocí bloku SELf budeme vybírat rychlost motoru. Rychlosti jsou nastaveny jako konstanty (0: motor stojí, 1500: maximální výstupní frekvence řídícího systému). Rychlosti se vybírají pomocí bitu @bit1. „Duty“ je střída výstupu v procentech a nastavíme ji na polovinu, tedy na 50 %.



Obrázek 45: Frekvenční výstup pomocí "FreqOutLA"

8.3.7. Úloha 4: Třídění vršků podle barvy

8.3.7.1. Cíl

Vytřídit vršky podle vnitřní barvy.

8.3.7.2. Příklad

Navrhněte program pro třídění vršků pomocí barevného senzoru O5C500. Výstup tohoto senzoru je zapojen na vstup DI0.3 a výstup AT je připojen na DI0.4. Kontrastní senzor nastavte tak, aby spínal jen tehdy, když vršek bude mít červenou barvu, pokud nebude mít červenou barvu, tak senzor nebude sepnutý. Budou se vyhazovat vršky s červenou barvou (DO0.3).

! Pozor: pokud vršek není tříděný, senzor vezme i druhou hranu vršku

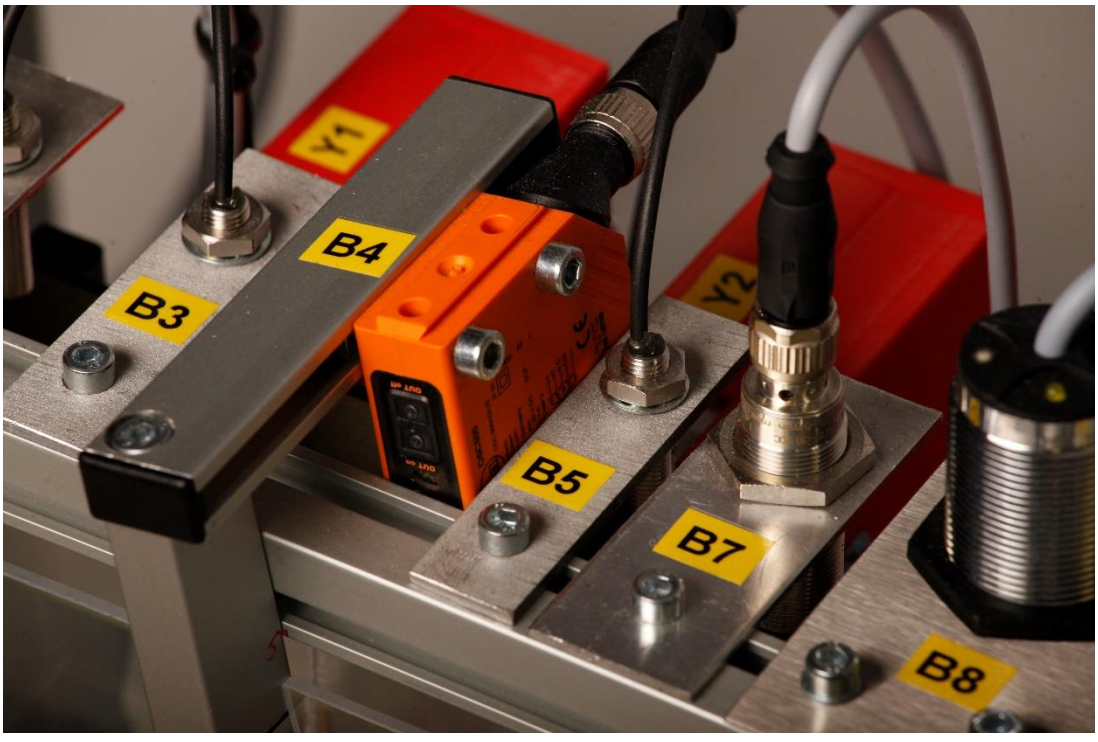
? Nápověda: pokud je vršek netříděný, počítejte hrany

? Nápověda: Učení senzoru se provádí vložením předmětu do snímacího prostoru. Podržením tlačítka ON dokud nezačne blikat (2Hz) (senzor bude spínat na tuto barvu), nebo se podrží tlačítko OFF dokud nezačne blikat led (2Hz)(senzor bude rozepínat na tuto barvu). Tlačítka musí být stisknuta po dobu 2s.

! Pozor: pokud signalizační led bliká rychle (8Hz), nastavení senzoru je chybné. Může být zapříčiněno malým rozdílem mezi kontrasty

8.3.7.3. Realizace

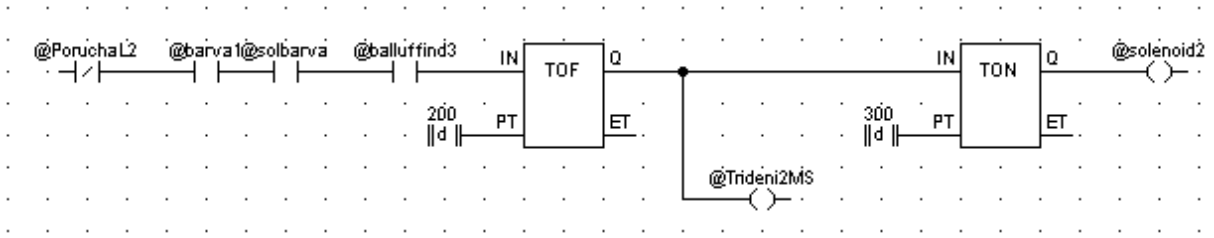
Solenoid (Y2) pro třídění podle barvy je spínaný pomocí relé, které je zapojeno na výstup DO0.3. Senzor „BES M08MG-GSC20B-BP03“ pro polohu vršku, zapojený na DI0.6, sepne, když je na místě vršek pro třídění (Vršek má ale dvě hrany). Senzor barvy snímá střed vršku, když je vršek v poloze pro třídění.



Obrázek 46: Senzor barvy na lince

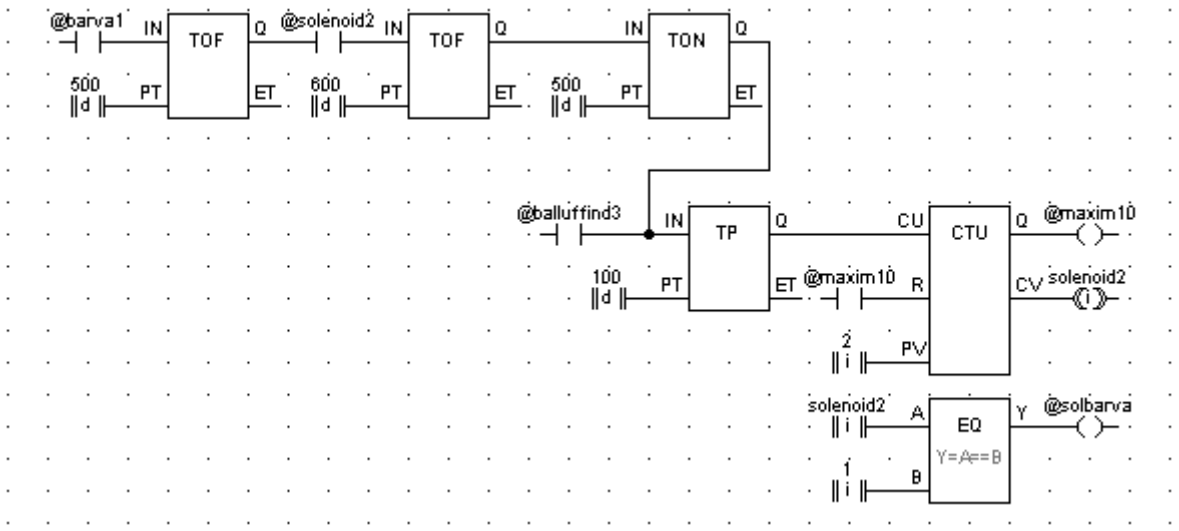
8.3.7.4. Program

Vytvoříme algoritmus (cyklus) pro třídění. Pokud bude mít vršek barvu, kterou jsme předem nastavili na snímači B4 (senzor B4 sepne), tak vršek bude vytříděn za krátké přestávky chodu dopravníku. Pokud však má vršek jakoukoliv jinou barvu, než kterou jsem předem naučili snímač B5, nic se neděje a vršek dále pokračuje po dopravníku.



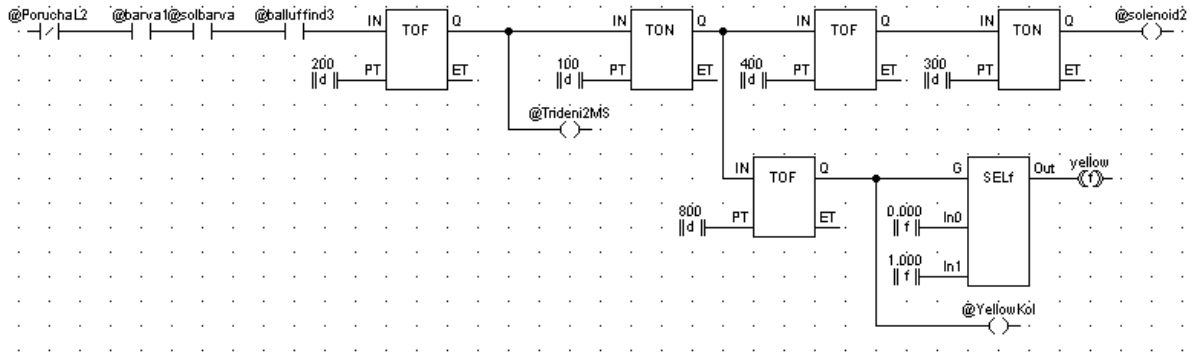
Obrázek 47: Cyklus třídění podle barvy

Nyní již třídíme vršky, které mají barvu, kterou jsme předem nastavili na snímači B5. Problém nastane tehdy, když je vršek s touto barvou a pojede dále. Senzor zaznamená i druhou hranu vršku, cyklus se spustí znovu a bude se třídit i s druhou hranou. Musíme tedy zajistit aby, když je vršek dobrý, nereagoval na druhou hranu.



Obrázek 48: Řešení dvou hran

Pro úplnost přidáme světelný signál při třízení na HL2 na kterém si vybereme kteroukoliv barvu jako signalizaci cyklu třídění. A cyklus třídění bude vypadat následovně



Obrázek 49: Cyklus třídění

8.3.8. Úloha 5: Třídění vršků podle otočení

8.3.8.1. Cíl

Zjistit otočení vršku a špatné vršky vytřídit.

8.3.8.2. Příklad

Navrhněte program pro třídění vršků. Snímač B6 (AI0.6) snímá natočení vršků a snímač B2 (DI0.1) udává, zda je vršek v pozici, kdy se může snímat otočení vršku. Otočené vršky se budou třídit (vyhazovat) (DO0.2).

! Pozor: mezi tříděním a rozpoznáváním vršků je mezera.

! Pozor: řídící systém si musí zapamatovat pořadí vršků a postupně je počítat. Použijte proměnnou, nebo matici pro ukládání pořadí.

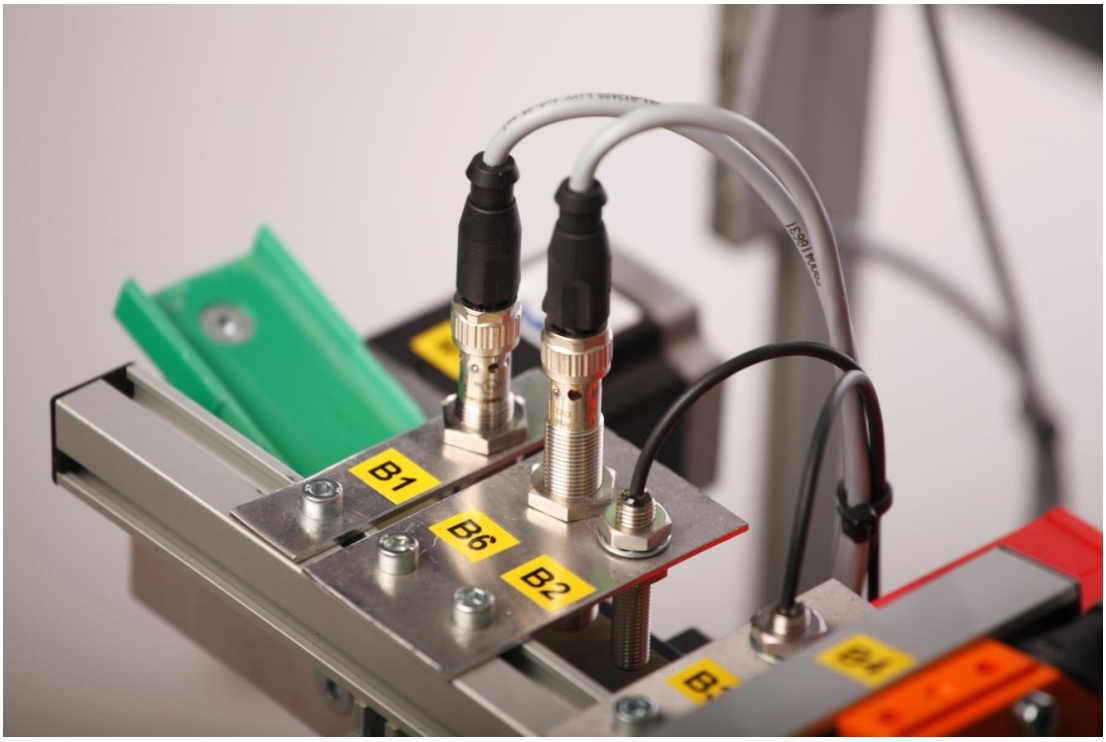
? Nápověda: můžeme se dotázat na jednotlivé bity proměnných (políček matic) pomocí jiné proměnné.

! Pozor: musí být provedení přetypování proměnné (příkaz „bool.()“) pokud ukládáme hodnotu do aliasu z políčka matice.

! Pozor: po zapnutí stanice musí dojít k nulování proměnných pro řízení zápisu a čtení z proměnné.

8.3.8.3. Realizace

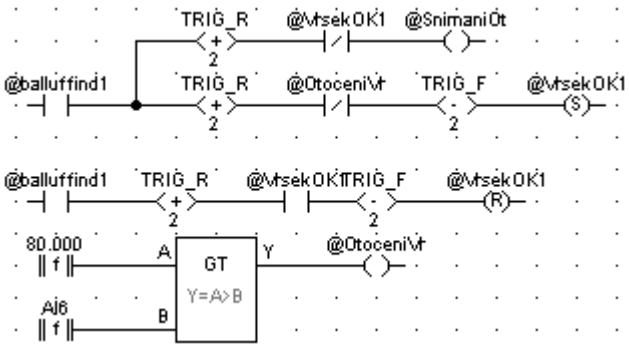
Solenoid Y1 pro třídění vršků podle natočení je spínaný pomocí relé, které je zapojeno na výstup DO0.2. Senzor B3 pro polohu vršku, zapojený na DI0.2, sepne, když je na místě vršek pro třídění (Vršek má ale dvě hrany). Analogový indukční senzor B6 snímá “vzdálenost“ indukčního materiálu od snímače, čímž můžeme zjistit, zda je vršek natočen správně či nikoliv. Snímání probíhá při sepnutí snímače B2, který zaznamená vršek (Vršek má ale dvě hrany).



Obrázek 50: Senzory pro třídění podle natočení na lince

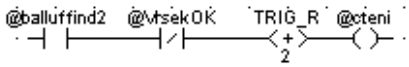
8.3.8.4. Program

Vytvoříme algoritmus (cyklus) pro třídění, který se bude skládat z několika částí. Hlavní částí programu je matice, do které se zapisují hodnoty, jelikož než je vršek vytříděn mohou se vyhodnocovat další. První částí tedy bude řízení, kdy bude program zapisovat do matice. Následovat bude samotná matice a nakonec bude proces pro samotný třídící proces pro solenoid Y1.



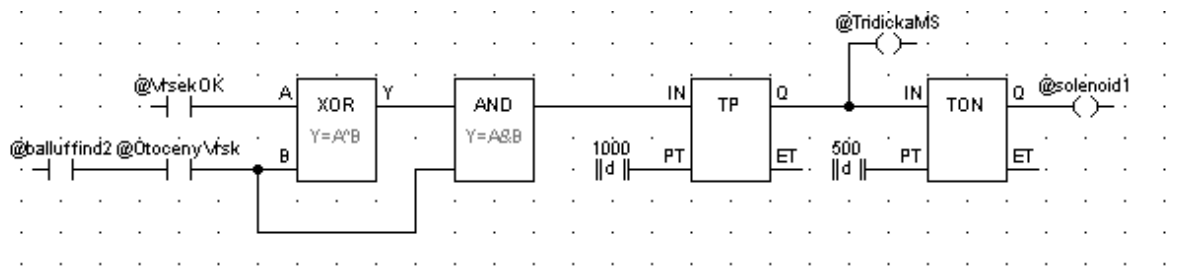
Obrázek 51:Cyklus, který řídí kdy má program zapisovat do matice

```
If @SnimaniOt
| Let MATridenil[0, MaWrVrOk] = @OtoceniVr
| Let MaWrVrOk = MaWrVrOk + 1
| If MaWrVrOk >= 16
| | Let MaWrVrOk = 0
| EndIf
EndIf
```




```
If @cteni
| Let @OtocenyVrsk = bool(MATridenil[0,MaRdVrOk])
| Let MaRdVrOk = MaRdVrOk + 1
| If MaRdVrOk >= 16
| | Let MaRdVrOk = 0
| EndIf
EndIf
```

Obrázek 52: Matice pro pamatování si více vršků



Obrázek 53: Cyklus třídění pro solenoid Y1

Obrázek 54: Ochrana dvou hran u solenoidu Y1

8.3.9. Úloha 6: Selekce vršků

8.3.9.1. Cíl

Třídit vršky podle pořadí, ve kterém jsou snímány senzorem B8.

8.3.9.2. Příklad

Navrhněte program pro selekci vršků. Selekce vršků spočívá v tom, že je vyhozen každý X-tý vršek. V našem případě bude vyhazován každý druhý vršek.

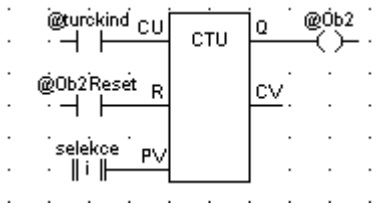
8.3.9.3. Realizace

Solenoid Y3 pro třídění vršků podle natočení je spínaný pomocí relé, které je zapojeno na výstup DO0.4. Senzor B8 pro polohu vršku, zapojený na AI0.0, sepne, když je na místě vršek. Senzor je umístěn nad solenoidem Y3 a zároveň plní funkci hlavního prvku pro selekci vršků. Díky své velikosti a parametrům je ideální pro snímaní a selekci vršků. Senzor sepne, právě tehdy, když je plocha snímaného vršku pod plochou snímače B8. Svými parametry tedy zamezuje tomu, že by mohl být vršek sejmut vícekrát, což zjednoduší program.

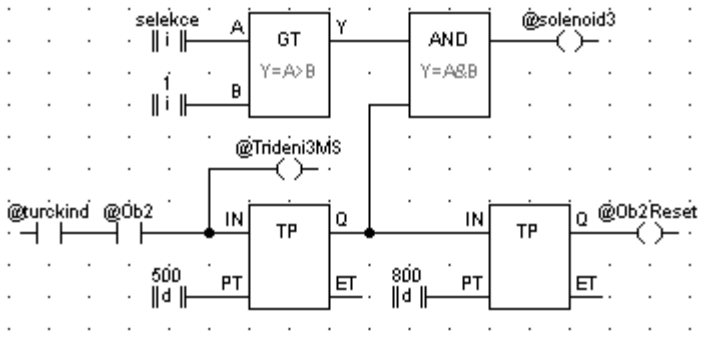
Obrázek 55: Senzor B8 pro selekci

8.3.9.4. Program

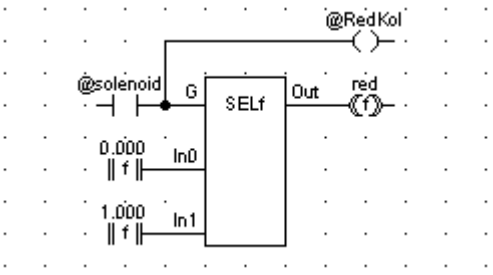
Vytvoříme algoritmus (cyklus) pro třídění. Hlavní částí programu bude čítač, u kterého budeme moci nastavit, který vršek v pořadí má být vytříděn. Následuje už pouze jen cyklus pro solenoid Y3. Místo selekce si doplníme kolikátý vršek v pořadí má být vytříděn.



Obrázek 56: Cyklus s čítačem



Obrázek 57: Cyklus třídění s ochranou proti vytřídění každého vršku



Obrázek 58: Světelná signalizace třídění

8.3.10. Úloha 7: Testování různých senzorů

8.3.10.1. Cíl

Zjistit, jaký je rozdíl mezi jednotlivými senzory různých typů, které fungují na stejném principu.

8.3.10.2. Příklad

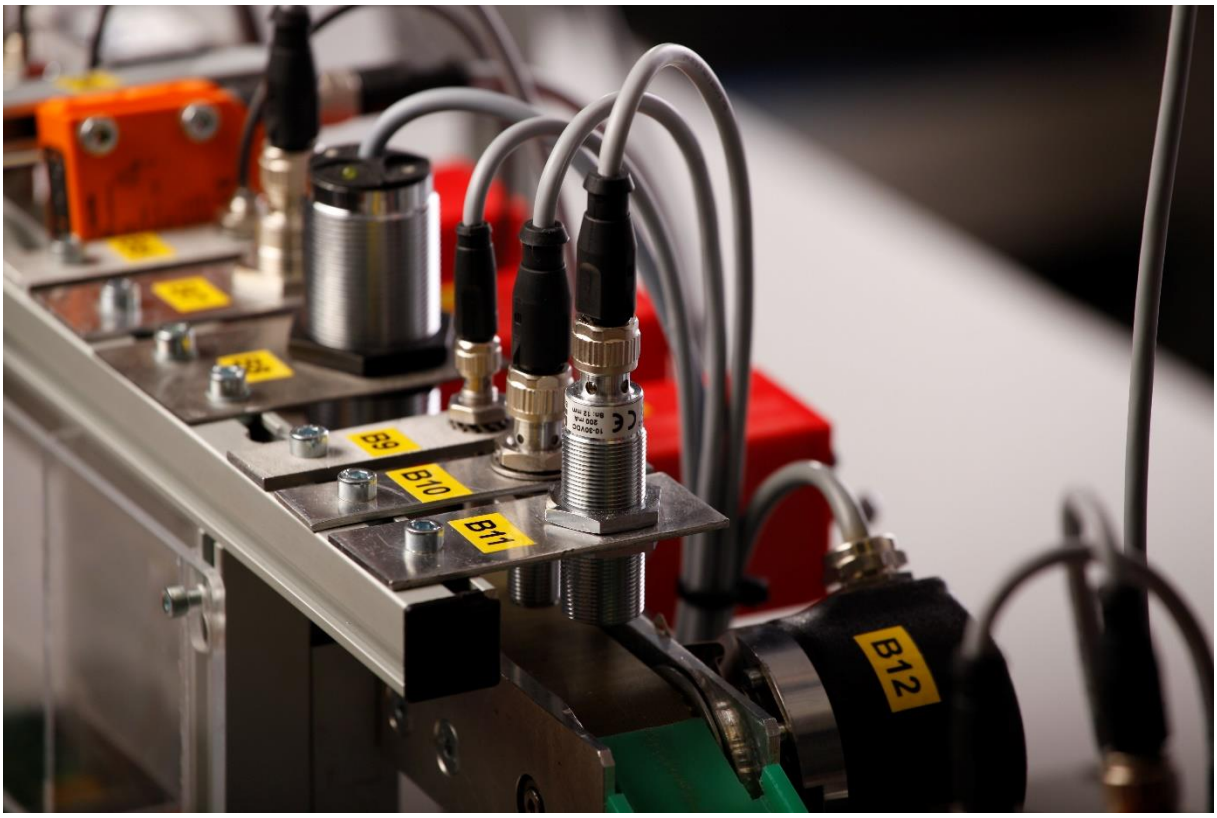
Vyzkoušejte si, jak fungují senzory B9, B10 a B11 různých velikostí, vstupy AI.1-3, a zjistěte, proč každý dokáže předmět snímat z jiné vzdálenosti.

? Nápověda: indukční senzory nereagují na magnetické předměty ale na vodivé předměty (železo, hliník, měď, …).

 # Extra: Zkuste fyzikálně vysvětlit, jak se měří indukce.

8.3.10.3. Realizace

Senzory B9, B10, B11 jsou připojeny na vstupy AI0.1-3. Senzory jsou od různých výrobců a mají různé vlastnosti, jako je například spínací vzdálenost.



Obrázek 59: Různé senzory na zkoušení funkce

8.3.11. Úloha 8: Inkrementální snímač

8.3.11.1. Cíl

Naučit se pracovat s inkrementálním snímačem.

8.3.11.2. Příklad

Navrhněte program pro snímání otočení vršku pomocí senzoru B7 (AI0.7). Otočené vršky vytřiďte. Použijte inkrementální snímač B12.

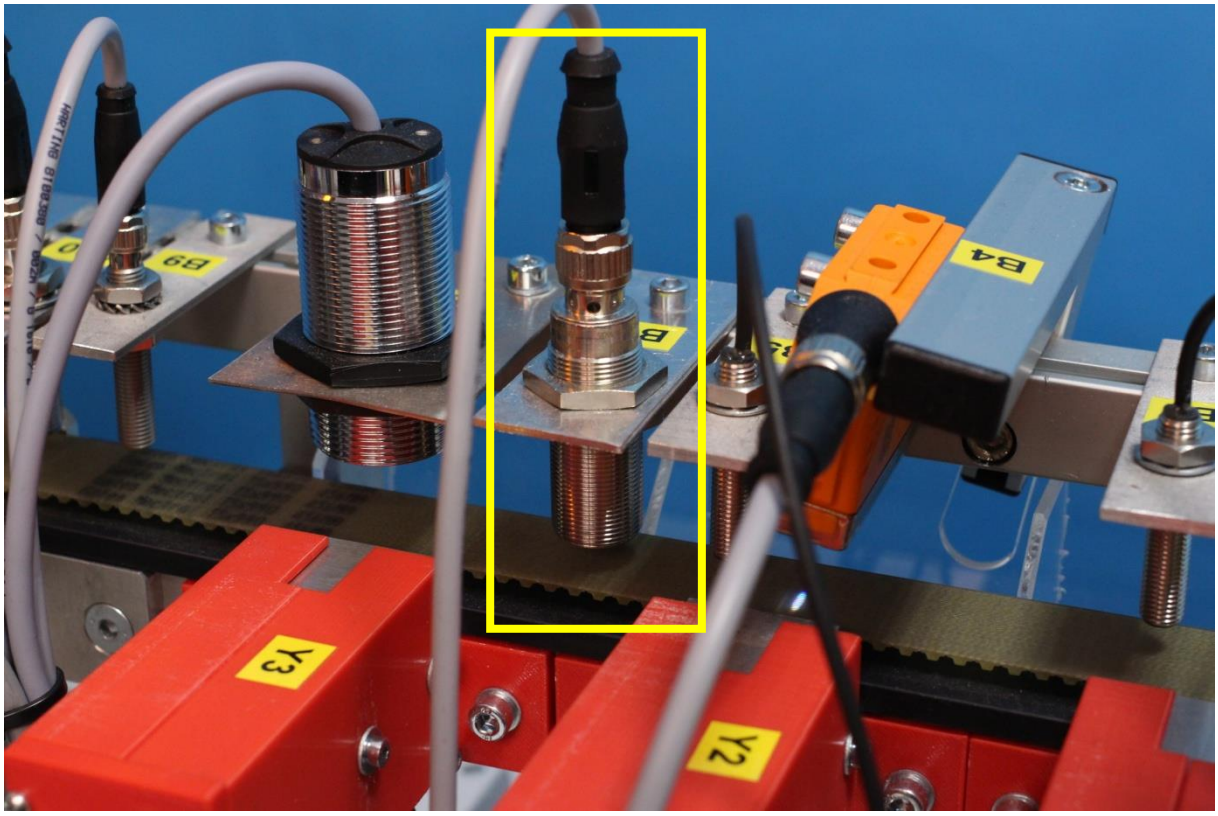
? Nápověda: použijte čítače pro určení kraje vršku a středu vršku.

 # Extra: snímejte rychlost enkodéru (stejný převodový poměr vůči motoru) a převeďte na jednotku RPM a zobrazte na terminálu AMiNi2D.

! Pozor: inkrementální snímač má 1024 pulzů na otáčku.

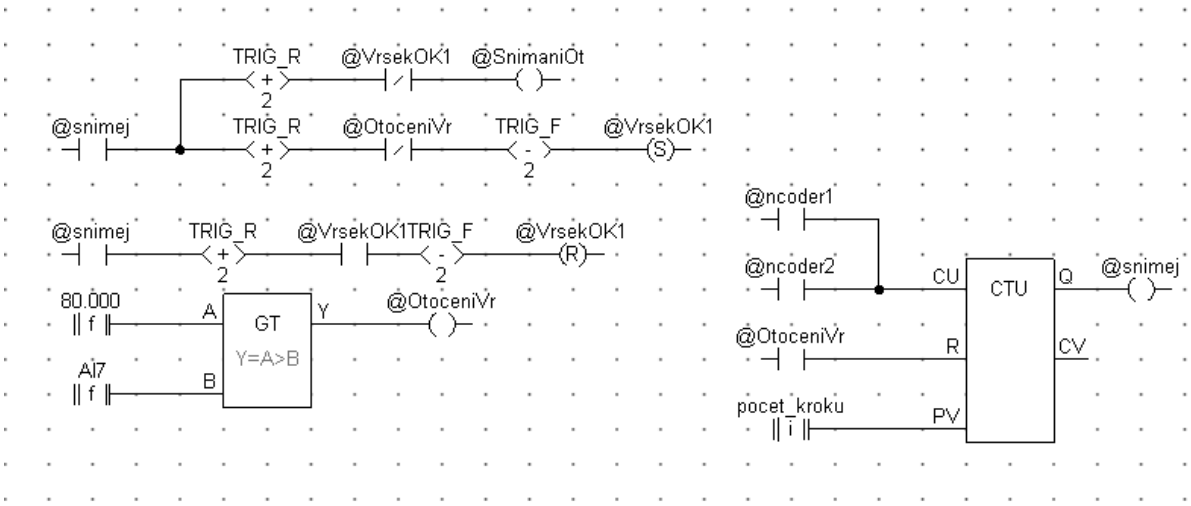
8.3.11.3. Realizace

Senzor B7 zapojený na vstup AI0.7 použijeme pro snímání vzdálenosti vodivé plochy pro určení natočení vršku. Snímač B12 připojený na vstupy DI0.6-7 použijeme pro zjištění, kde se vršek na dopravníku nachází. Pro proces třídění budeme muset zavést matici jako v úloze pět, která se ale bude řídit podle snímače B12. Nakonec použijeme blok pro převod signálu ze snímače B12 na jednotky RPM.



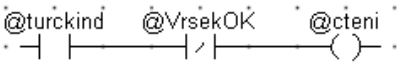
Obrázek 60: Snímač B7

8.3.11.4. Program



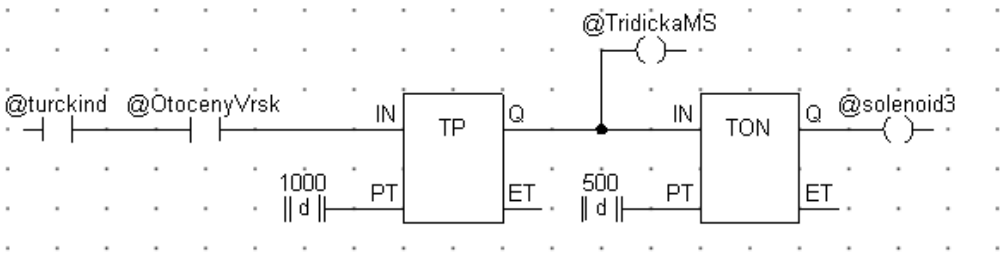
Obrázek 61: Cyklus, který řídí kdy má program zapisovat do matice

```
If @SnimaniOt
| Let MATrideni1[0, MaWrVrOk] = @OtoceniVr
| Let MaWrVrOk = MaWrVrOk + 1
| If MaWrVrOk >= 16
| | Let MaWrVrOk = 0
| EndIf
EndIf
```




```
If @cteni
| Let @OtocenyVrsk = bool(MATrideni1[0,MaRdVrOk])
| Let MaRdVrOk = MaRdVrOk + 1
| If MaRdVrOk >= 16
| | Let MaRdVrOk = 0
| EndIf
EndIf
```

Obrázek 62: Matice pro pamatování si více vršků



Obrázek 63: Cyklus třídění pro solenoid Y3

8.3.12. Úloha 9: Řízení celé linky

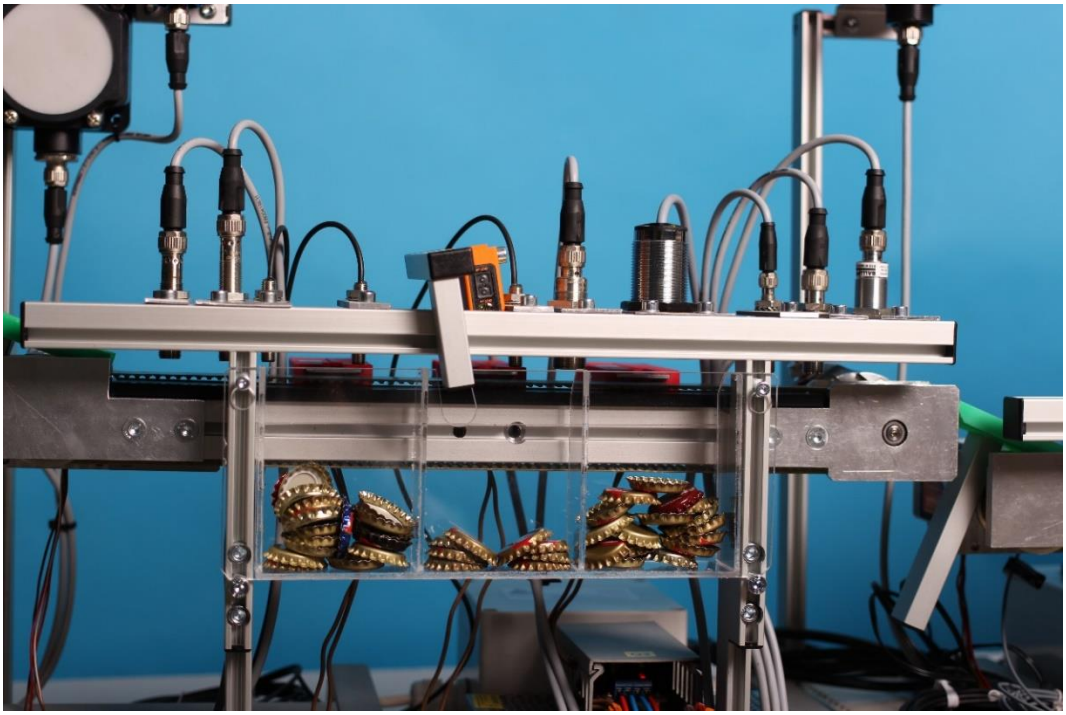
8.3.12.1. Cíl

Zkombinovat všechny možné úlohy a synchronizovat je.

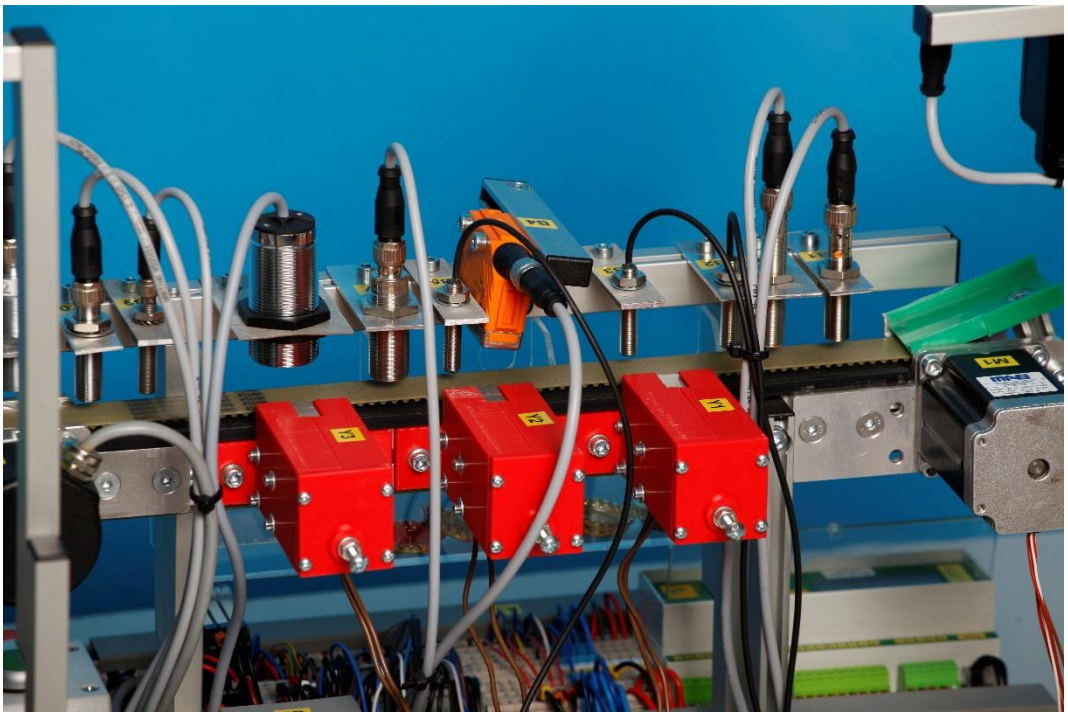
8.3.12.2. Příklad

Navrhněte program pro řízení celé linky. Využijte programů a znalostí z předchozích úloh. Linka bude třídit vršky podle natočení, třídit podle barvy a bude dělat selekci vršků.

8.3.12.3. Realizace



Obrázek 64: Linka zepředu



Obrázek 65: Linka zezadu

8.3.12.4. Program

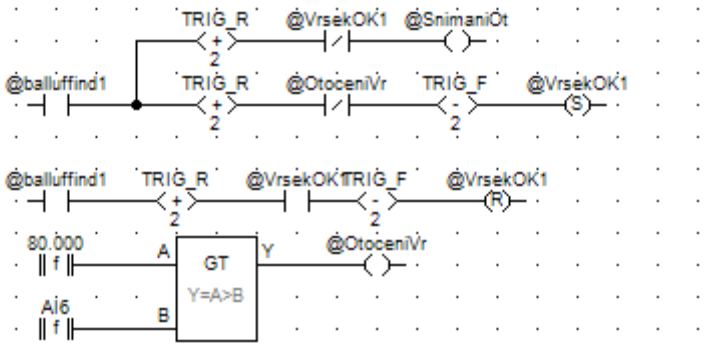
```
----------------------------------------------Proces INIT----------------------------------------------

let Motor = 1500
Let MaWrVrOk = 0
let MaRdVrOk = 0
let Selekce = 2
let @VrsekOK1 = false
let @PoruchaL2 = false
let @OdstavkaL2 = false

-----------------------------------------------Konec-----------------------------------------------
```

V procesu INIT dojde k nastavení potřebných hodnot do výchozích. Tento proces proběhne pouze jednou. Využijeme ho tedy i na resetování poruchy, selekce a nastavení rychlosti motoru.

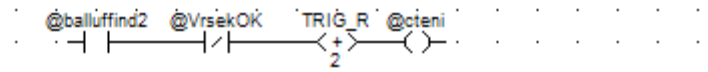
```
-------------------------Proces pro čtení vstupů a zápis výstupů--------------------------------

      DigOut Digital_out, #0, 0x0000
      DigIn #0, Digital_in, 0x0000
      DigIn #2, Analog_in, 0x000F

      AnOut #AO00_0, yellow, 10.000, 0.000, 10.000, 0.000, 1.000
      AnOut #AO00_1, green, 10.000, 0.000, 10.000, 0.000, 1.000
      AnOut #AO00_2, blue, 10.000, 0.000, 10.000, 0.000, 1.000
      AnOut #AO00_3, red, 10.000, 0.000, 10.000, 0.000, 1.000

      AnIn #AI00_6, AI6, 10.000, 0.000, 10.000, 0.000, 100.000
      AnIn #AI00_7, AI7, 10.000, 0.000, 10.000, 0.000, 100.000
-----------------------------------------------Konec-----------------------------------------------
```

Zde přečteme vstupy a zapíšeme je do proměnných a naopak.

----------**Proces třídění otočených vršků**----------

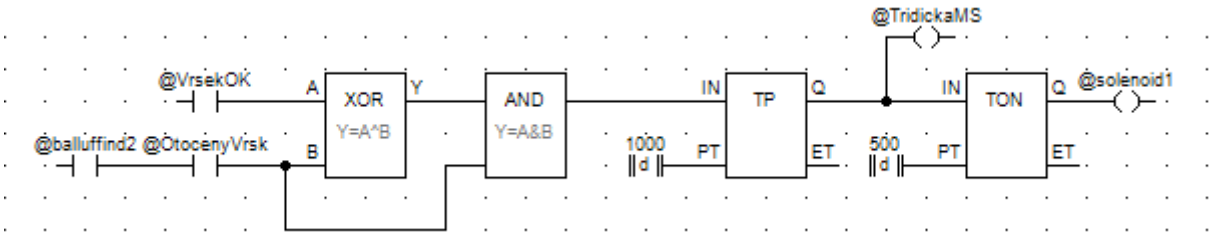



```
If @SnimaniOt
| Let MATrideni1[0, MaWrVrOk] = @OtoceniVr
| Let MaWrVrOk = MaWrVrOk + 1
| If MaWrVrOk >= 16
| | Let MaWrVrOk = 0
| EndIf
EndIf
```

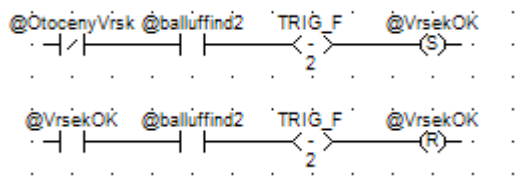



```
If @cteni
| Let @OtocenyVrsk = bool(MATrideni1[0,MaRdVrOk])
| Let MaRdVrOk = MaRdVrOk + 1
| If MaRdVrOk >= 16
| | Let MaRdVrOk = 0
| EndIf
EndIf
```

Obrázek 66: Snímaní natočení vršku



Obrázek 67: Algoritmus třídění

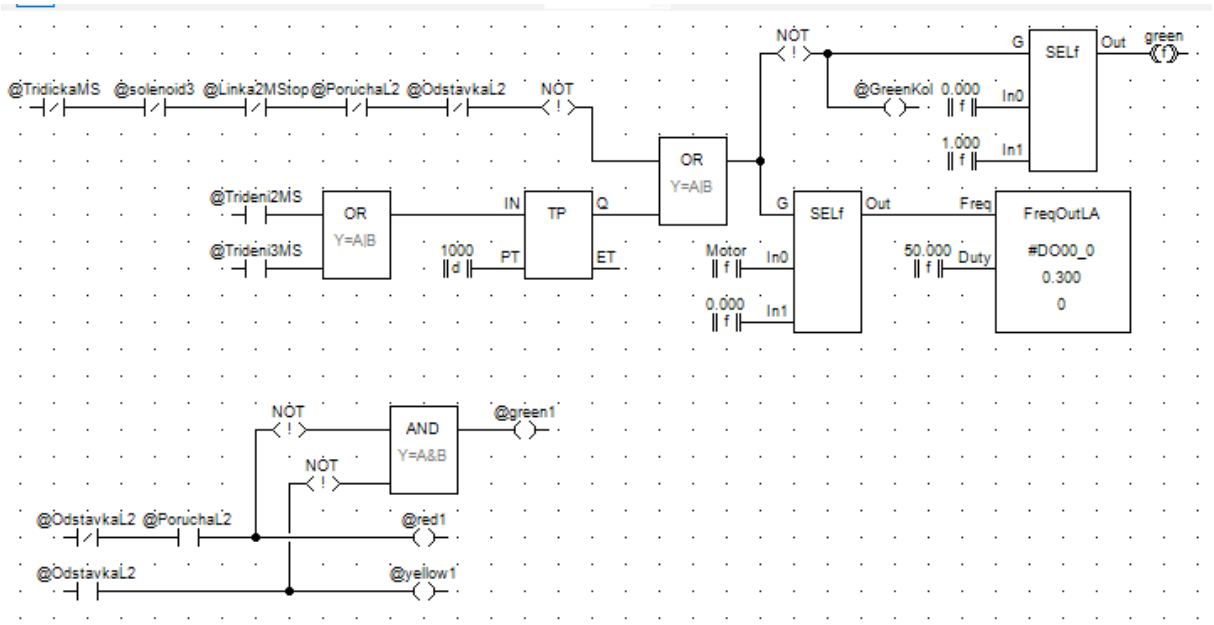


Obrázek 68: Ochrana druhé hrany

----------**Konec**----------

Nejdříve musíme zjistit, jak je vršek natočený. Nejedná se o okamžité vyhodnocení, a proto využijeme matic na zapamatování vršků. Poté musíme zabezpečit druhou hranu vršku.

----------Proces řízení motoru----------

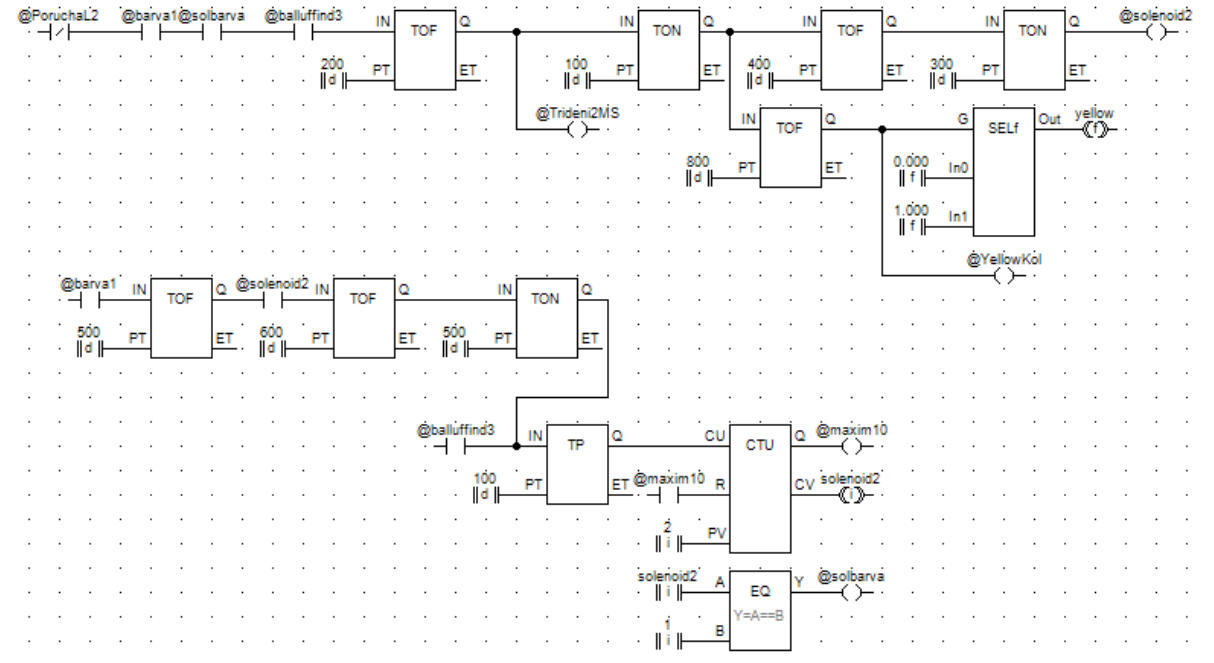


Obrázek 69: Řízení motoru

----------Konec----------

Proces pro řízení motoru.

----------Proces třídění podle barvy----------

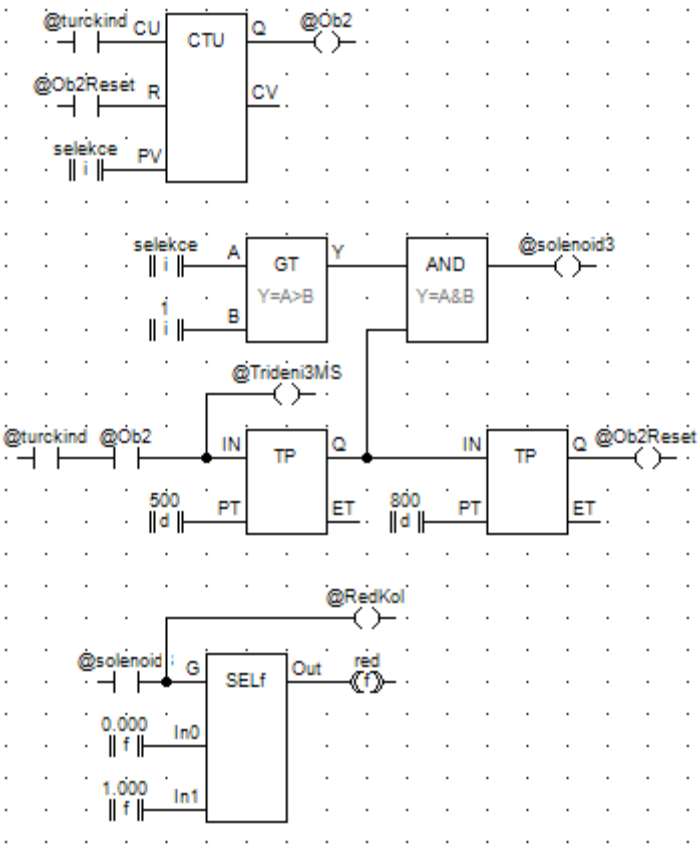


Obrázek 70: Třídění podle barvy

----------Konec----------

Třídění podle barvy probíhá okamžitě, takže stačí pouze vyhodnocovat. Musíme ochránit, když bude vršek netříděný.

----------Proces selekce vršků----------



Obrázek 71: Algoritmus pro selekci vršků

----------Konec----------

Porovnáváme proměnnou selekce s jedničkou. Pokud by byla hodnota v selekci 1, tak by byl vyhozen každý vršek. Pro ukázku jsme toto nepoužili a třídíme minimálně každý druhý vršek, anebo žádný, pokud je selekce 0. Nastavení selekce je v procesu INIT, ale jde změnit přes ladění (inspektora).

8.3.13. Úloha 10: Uživatelské rozhraní

8.3.13.1. Cíl

8.3.13.2. Příklad

Navrhněte program s uživatelským rozhraním. Použijte program z úlohy 9. Vytvořte harmonogram mezi obrazovkami, aby se dalo mezi nimi přepínat. Pro motor bude vytvořte obrazovku, kde se bude zobrazovat rychlost motoru a editace rychlosti. Selekce vršků se bude moci také editovat pomocí terminálu a bude se moci dát i vypnout. Jedna obrazovka bude zahrnovat informace.

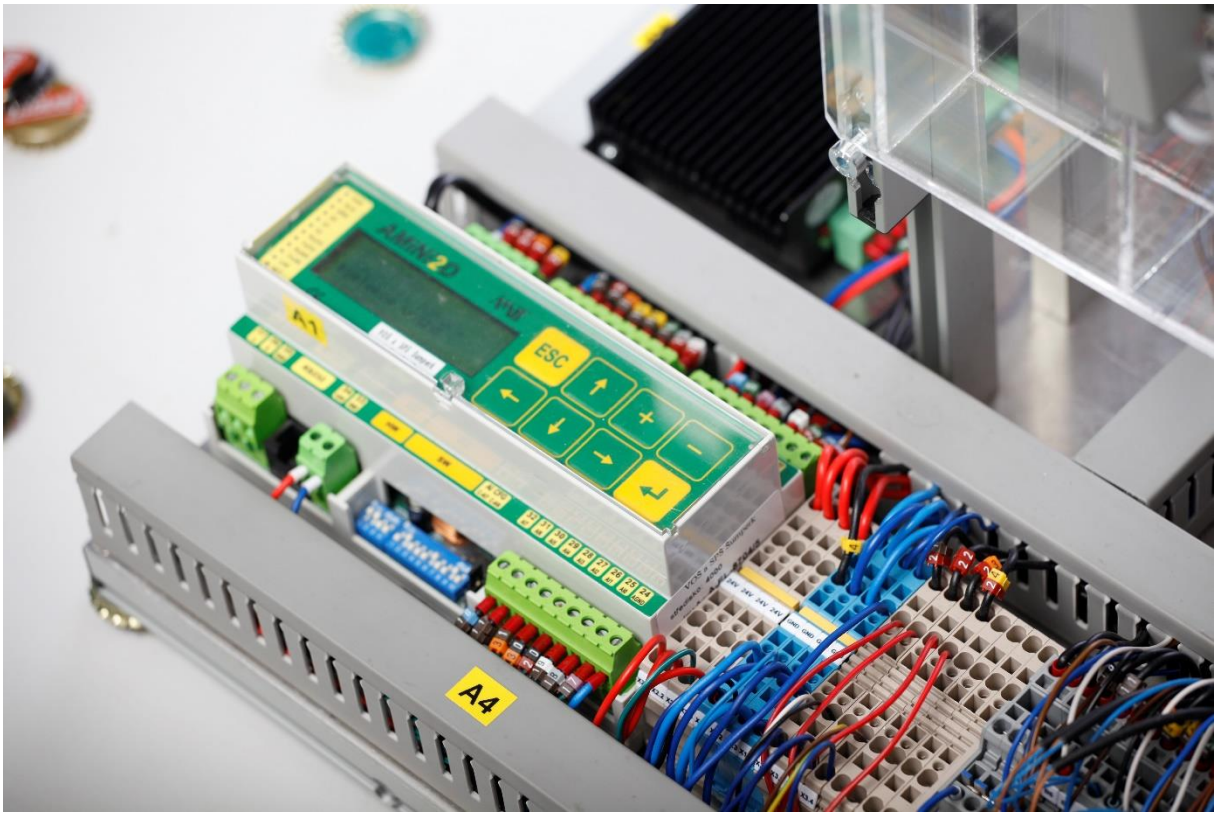
? Nápověda: vytvořte si hlavní obrazovku, do které umístíte „MenuScreen“, odkud se budete pohybovat mezi obrazovkami.

! Pozor: vytvořte na každé obrazovce, na kterou se odkazujete i zpáteční odkaz.

 # Extra: můžete použít přihlášení pro editaci proměnných.

! Pozor: je nutné nastavit u prvků, kdo může editovat a kdo může pouze vidět.

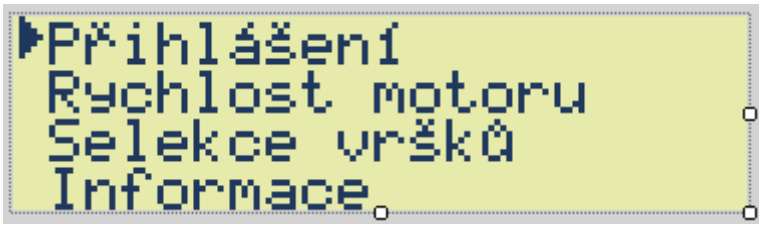
! Pozor: je nutné vytvořit pro „login“ skript, který zajistí výběr v modulu.



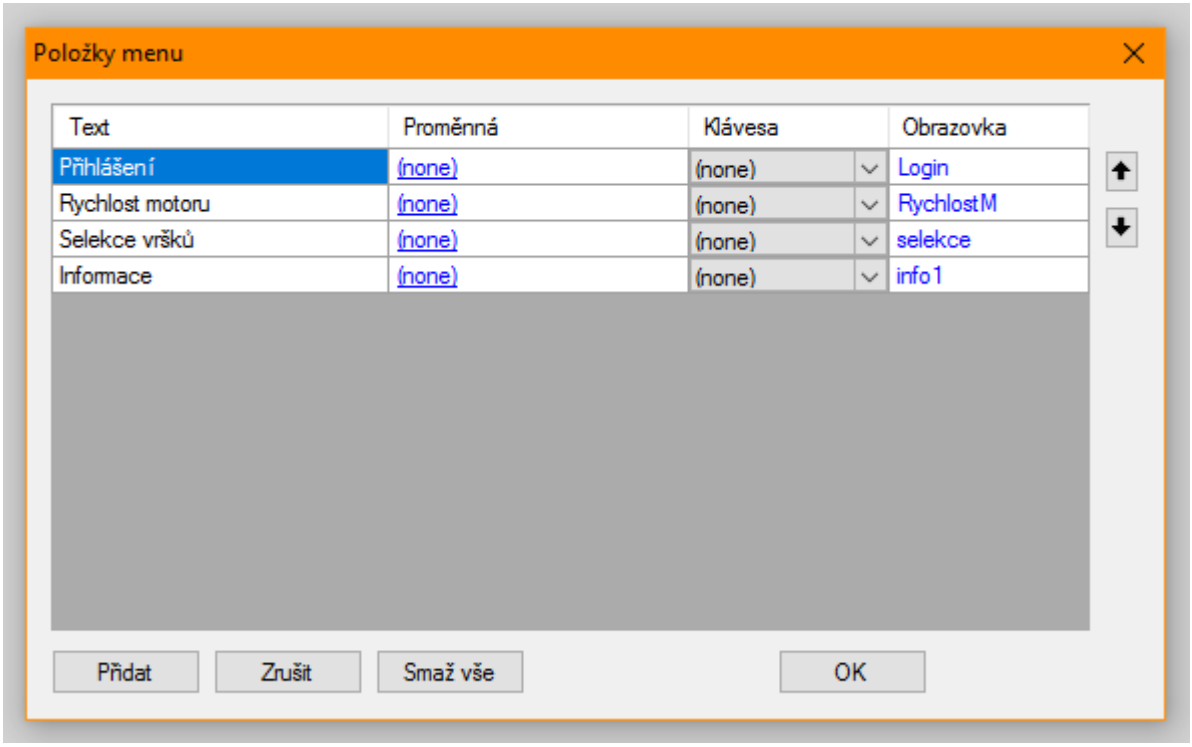
Obrázek 72: PLC s terminálem

8.3.13.3. Program

Pro přehlednost použijeme „MenuScreen“. Jedná se o menu, ve kterém vyberu obrazovku, na niž se chci odkázat.

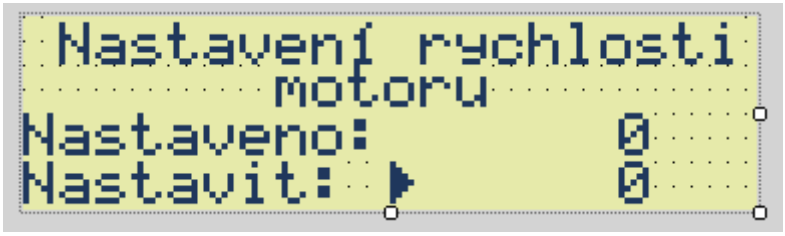


Obrázek 73: Obrazovka s "MenuScreen"

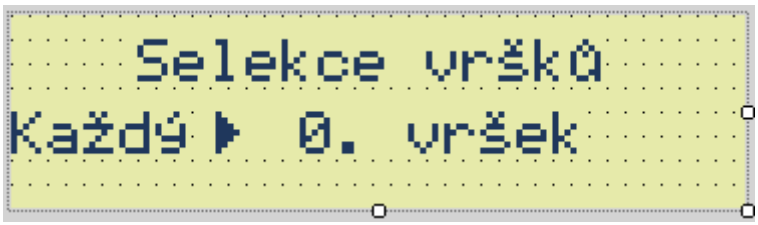


Obrázek 74: Nastavení "MenuScreen"

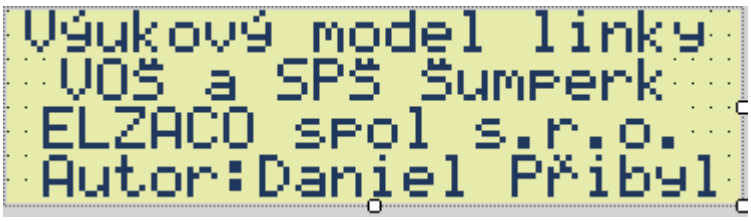
Poté vytvoříme obrazovky, kterým přiřadíme náležité funkce a vzhled. V každé obrazovce musí být zpáteční odkaz na rozcestník, jinak se nevrátíte do menu.



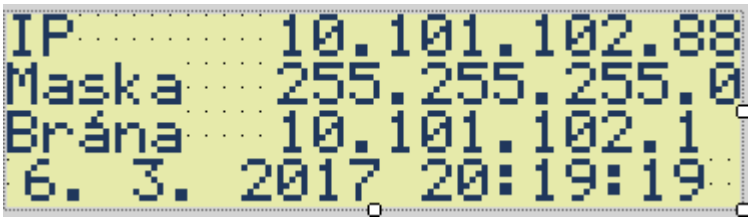
Obrázek 75: Obrazovka nastavení rychlosti motoru



Obrázek 76: Obrazovka nastavení selekce



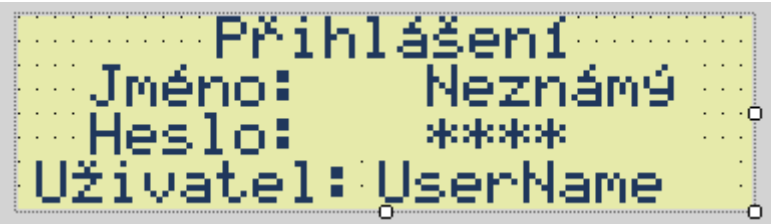
Obrázek 77: Obrazovka s informacemi



Obrázek 78: Obrazovka s informacemi

Extra: Přihlašovací obrazovka

Jako první je zapotřebí vytvořit uživatele (Projekt/Správa uživatelů): viz. Strana 123.

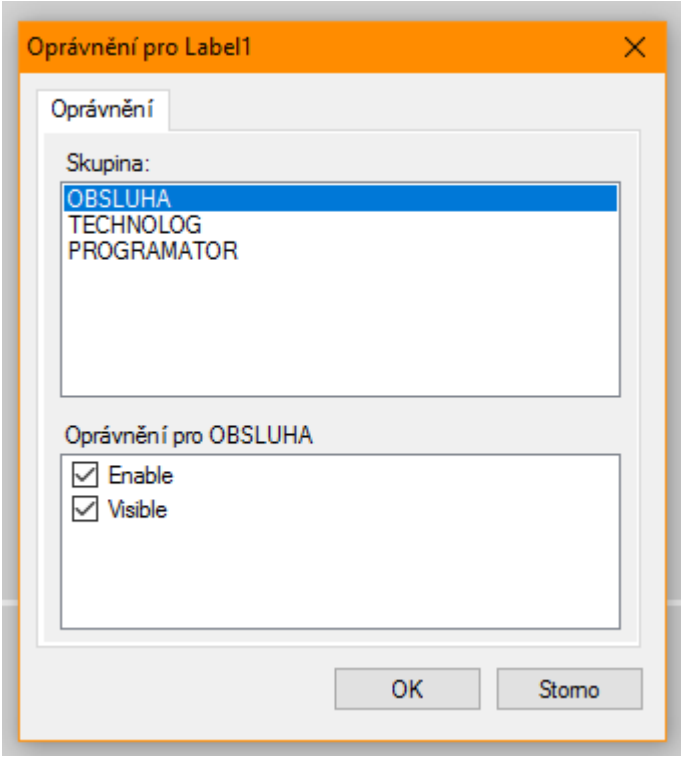


Obrázek 79: Přihlašovací obrazovka

Pro pohyb v modulu „LogIn“ je zapotřebí napsat skript. Skript modulu „LogIn“ bude vypadat následovně:

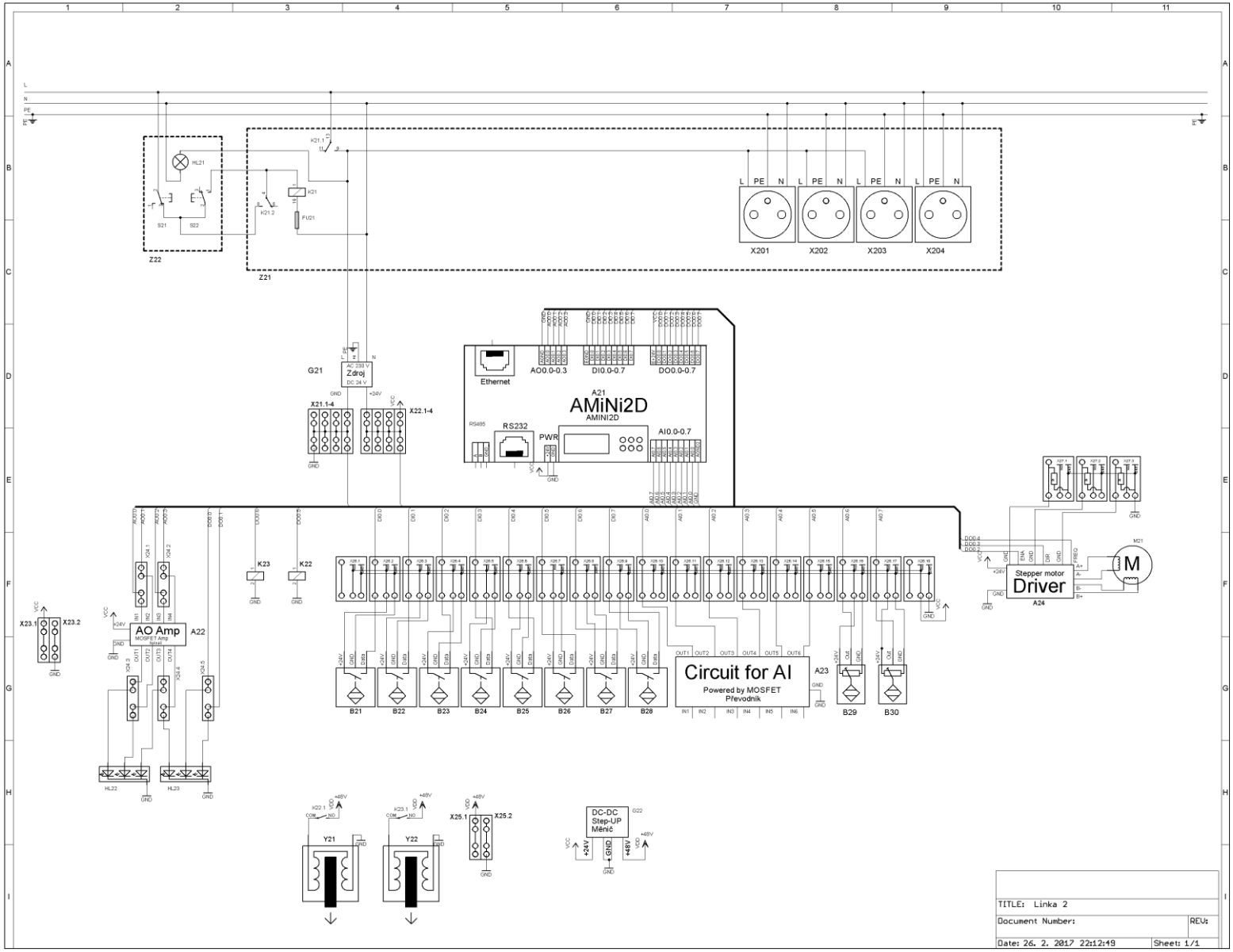
```
event Login1_OnAccessGrant()
      Login1.Focus();
      Application.SendKey(keys.up);
      Login1.Focus();
      Application.SendKey(keys.down);
end;
```

Přihlášení by bylo k ničemu, když by se pak nevyužívala práva uživatelů. Je tedy zapotřebí pak nastavit u každého bloku (najdete názvem „Permission“ a někdy je ještě pod záložkou „Advanced“), a pro každý typ uživatele jde nastavit viditelnost a zda ho sní používat.



Obrázek 80: Nastavení oprávnění

9. Linka 2



Obrázek 81: Schéma zapojení

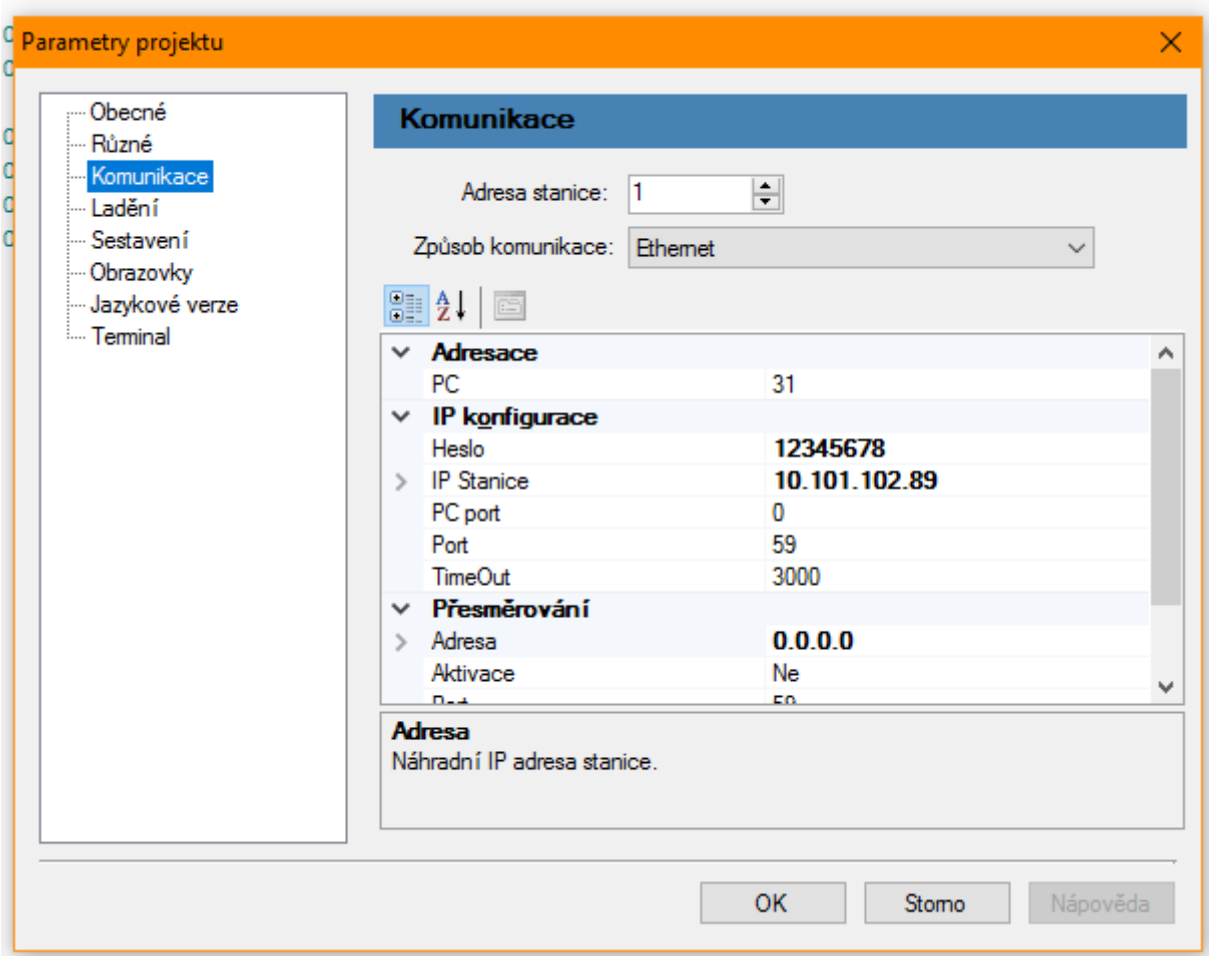
9.1. Zapojení vstupů a výstupů a použité senzory

PLC AMiNi2D				
Vstup	**Označení**	**Druh**	**Připojeno**	**Komentář**
AO0.0	HL22	Analogový výstup	Tříbarevný majáček (zelená)	Digitální výstup realizován vnějším obvodem
AO0.1	HL22	Analogový výstup	Tříbarevný majáček (červená)	Digitální výstup realizován vnějším obvodem
AO0.2	HL22	Analogový výstup	Tříbarevný majáček (žlutá)	Digitální výstup realizován vnějším obvodem
AO0.3	HL23	Analogový výstup	Tříbarevný zobrazovač (zelená)	Digitální výstup realizován vnějším obvodem
DO0.0	HL23	Digitální výstup	Tříbarevný zobrazovač (modrá)	
DO0.1	HL23	Digitální výstup	Tříbarevný zobrazovač (červená)	
DO0.2	A24	Digitální výstup	Motor (povolení)	Řízení brzdy motoru
DO0.3	A24	Digitální výstup	Motor (směr)	Řízení směru motoru
DO0.4	A24	Digitální výstup	Motor (rychlost)	Řízení rychlosti motoru
DO0.5	Y21	Digitální výstup	Solenoid třídění	
DO0.6	Y22	Digitální výstup	Solenoid plnění	
DO0.7		Digitální výstup	-	
DI0.0	B21	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.1	B22	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.2	B23	Digitální vstup	O5K500	
DI0.3	B24	Digitální vstup	Optozávora (Výstup OV5012)	
DI0.4	B25	Digitální vstup	QS18VP6LAF	
DI0.5	B26	Digitální vstup	KI5083	
DI0.6	B27	Digitální vstup	BES M08MG-GSC20B-BP03	
DI0.7	B28	Digitální vstup	Q12AB6FF30Q	
AI0.0	B29	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.1	B30	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.2	B31	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.3	B32	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.4	B33	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.5	B34	Analogový vstup	-	Digitální senzor na analogovém vstupu
AI0.6	B35	Analogový vstup	IF6031	
AI0.7	B36	Analogový vstup	IG6084	

Tabulka 2: Zapojení vstupů a výstupů řídícího systému na lince 2

9.2. Nastavení komunikace

Pro rychlé nahrávání programů je zvolen Ethernet (asi 6x rychlejší, nežli pře RS232), ale je nutné správné nastavení.



Obrázek 82: Parametrizace komunikace linky 2

- Zvolit adresu stanice – v našem případě 1
- Způsob komunikace – Ethernet
- Heslo – stanice má zprvopočátku heslo 12345678, ale jde změnit (viz. Nastavení IP adresy PLC – str.45)
- IP stanice – stanici jsme nadefinovali, že má IP adresu 10.101.102.89

9.3. Výukové úlohy

9.3.1. Úloha 1.1: Čtení digitálních snímačů

9.3.1.1. Cíl

Naučit se číst digitální vstupy, na které jsou připojeny senzory s digitálním výstupem a následně přečtená data zobrazit.

9.3.1.2. Příklad

Navrhněte program pro čtení digitálních snímačů, to jsou vstupy DI.0-7 a hodnoty 4 libovolně zvolených snímačů zobrazte na terminálu Amini2D. Můžete použít seznam použitých snímačů viz strana 26.

! Pozor: u AI0.0-5 čtení analogových vstupů jako digitálních.

! Pozor: u AI0.0-5 je zapotřebí invertovat vstupy.

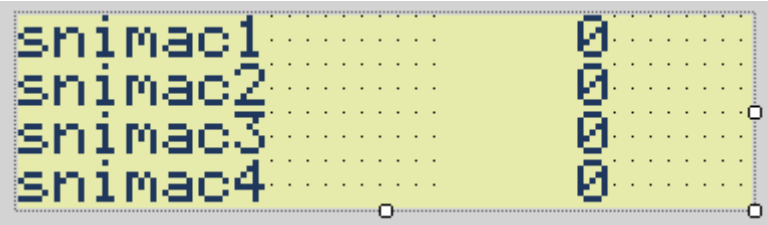
9.3.1.3. Realizace

Digitální vstupy jsou připojeny napřímo a vstupy AI0.0-5 jsou připojeny přes převodník (str.26), který obrací vstup, takže je zapotřebí číst tyto vstupy invertovaně.

9.3.1.4. Program

```
DigIn #0, vstupyD, 0x0000
```

DigIN načte vstupy na kanále 0 (DI0.0.7) a uloží je do proměnné „vstupyD“.



Obrázek 83: Displej pro zobrazování senzorů

Vytvoříte obrazovku podle předlohy a můžete doplnit na jakém vstupu je připojen. Na holý text se používá „Label“ a pro zobrazení hodnoty „Numeric View“. Po vybrání zobrazované proměnné se vybere i formát zobrazování. Jelikož budeme zobrazovat pouze bit (proměnnou bool), tak stačí jeden znak.

9.3.2. Úloha 1.2: Konfigurace a čtení analogových snímačů

9.3.2.1. Cíl

Naučit se správně nakonfigurovat vstupy řídícího systému a následně číst a zpracovávat přečtená data a zobrazovat je.

9.3.2.2. Příklad

Navrhněte program pro čtení analogových snímačů B29 a B30, to jsou vstupy AI.6-7, a hodnoty zobrazte na terminálu AMiNi2D. Můžete použít seznam použitých snímačů viz strana 26.

! Pozor: správná konfigurace analogových vstupů dle typu senzoru.

! Pozor: správné zvolení typu proměnné.

! Pozor: převod měřené hodnoty na fyzikální jednotky.

 # Extra: převeďte hodnotu z AI0.7 na vzdálenost a zobrazte ji na displeji.

! Pozor: Senzor neměří od nulové vzdálenosti a snímá 0,8-8mm.

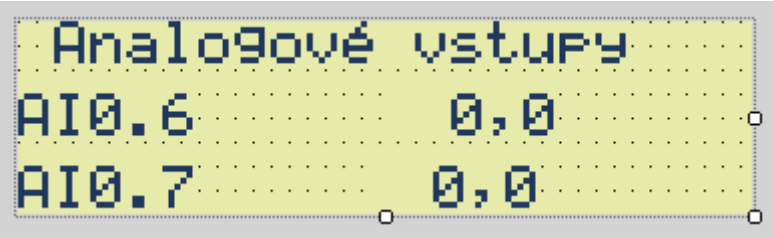
9.3.2.3. Realizace

Analogové senzory jsou připojeny na vstupy AI0.6 a AI0.7. Tyto vstupy jsou nastaveny jako napěťové 0-10V.

9.3.2.4. Program

```
AnIn #AI00_6, AI6, 10.000, 0.000, 10.000, 0.000, 10.000
AnIn #AI00_7, AI7, 10.000, 0.000, 10.000, 0.000, 10.000
```

Příkazy AnIn převádějí analogový vstup na fyzikální veličinu. Pokud budeme chtít pracovat s napětím, tak nadefinujeme ElMin na 0 a ElMax na 10, jelikož měříme napětí 0-10V. Toto nastavení je nadefinováno výrobcem senzoru. Mohou být senzory 2-8V a je to potřebí nastavit kvůli převodníku). PhysMin nastavíme na 0 a PhysMax na 10. Tyto parametry jsou dány pro převod z elektrické veličiny na fyzikální. My si však ponecháme napětí 0-10V. V případě, že bychom chtěli převádět napětí na vzdálenost, nastavíme podle měřené vzdálenosti senzoru. Napětí je lineárně závislé na vzdálenosti s tím, že senzor nezměří nulovou vzdálenost. V případě senzoru IG6084 nastavíme PhysMin na 0,8 a PhysMax na 8. Výstupem bude proměnná typu Float.



Obrázek 84: Displej pro zobrazování analogových vstupů

Podle obrázku vytvoříme obrazovku. Text je vložen pomocí „Label“ a proměnné pomocí „Numeric View“.

9.3.3. Úloha 1.3: Řízení solenoidů

9.3.3.1. Cíl

Naučit se řídit digitální výstupy a požívat časovače.

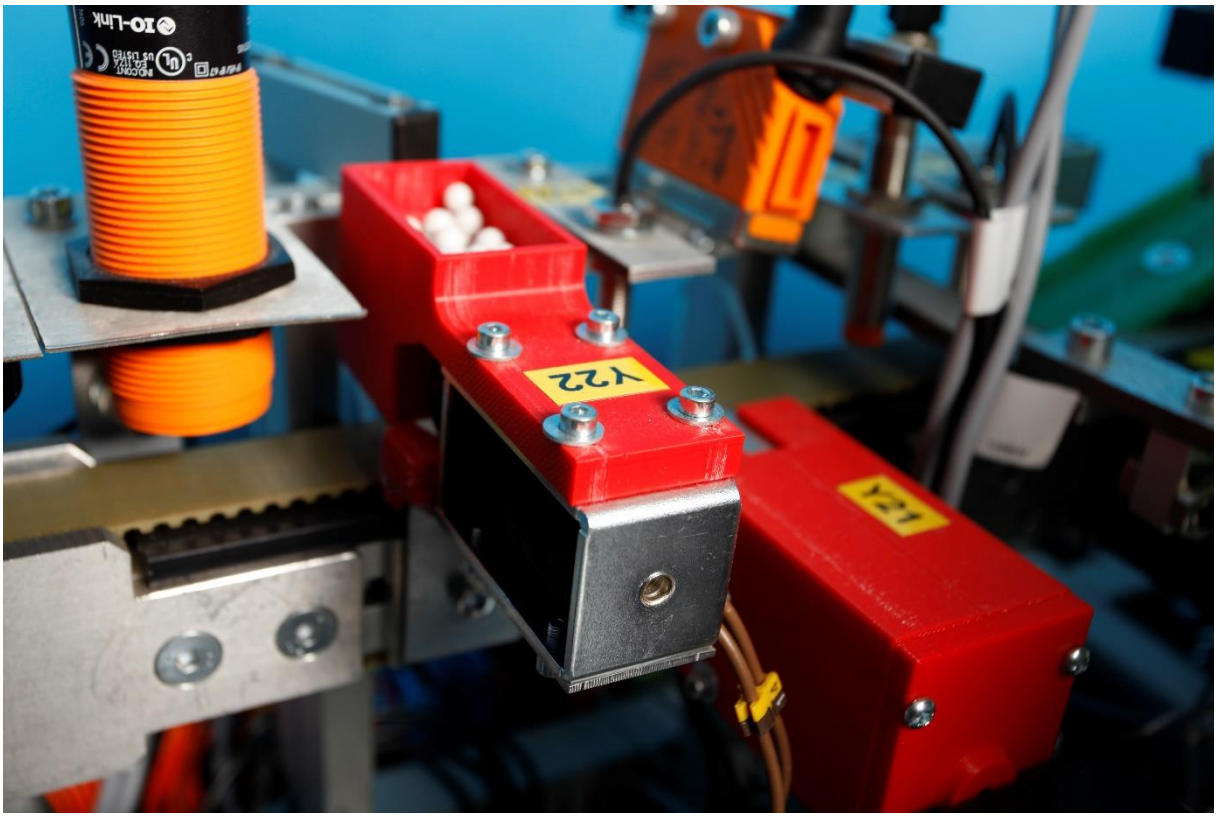
9.3.3.2. Příklad

Navrhněte program pro řízení solenoidů Y21 a Y22, to jsou výstupy DO0.5 a DO0.6, tak abyste je mohli ovládat pomocí tlačítek na terminálu AMiNi2D tak, aby solenoid po stisknutí tlačítka nezůstal ve vysunuté poloze, ale aby se vrátil do klidové polohy.

! Pozor: dlouhé buzení solenoidů může vést k jejich zničení.

9.3.3.3. Realizace

Solenoidy jsou připojeny na výstupech DO0.5 a DO0.6. Jelikož jsou připojeny na 48V spínány pomocí relátka. Solenoidy jsou vysunuté pouze tehdy, když je na nich napětí a vracejí se pomocí pružiny.



Obrázek 85: Solenoidy na lince

9.3.3.4. Program

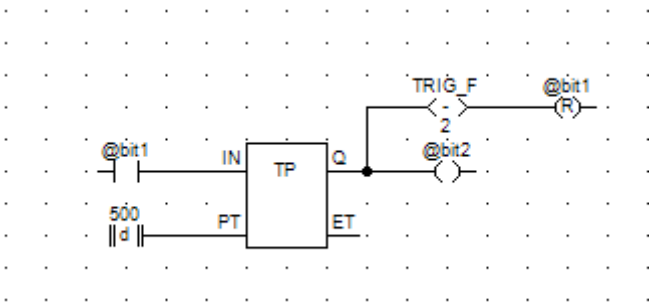
Pro ovládání pomocí terminálu je zapotřebí vytvořit obrazovku, kam se budou vkládat příkazy pro ovládání. Tyto příkazy by se mohly dát vložit i do obrazovky global, ale tato tlačítka by byla již pak nepoužitelná pro další obrazovky, jelikož obrazovka global běží neustále.

Pro ovládání solenoidů použijeme příkaz „KeyBit“

(Name)	**Plnicka**
Enabled	**True**
KeyCode	**Up**
Mode	**Set**
Permissions	**All**
Variable	**@bit 1**

Obrázek 86: Parametrizace objektu "KeyBit"

Při parametrizaci „KeyBit“ nastavíme, že má uložit do bitu „bit1“ log.1, se kterou následně pracujeme.



Obrázek 87: Řízení solenoidu

Pokud je tedy @bit1 log.1, tak se spustí časovač „TP“, který je nastaven na 500ms. Po tuto dobu má @bit2 logickou úroveň 1 a je tedy solenoid vysunutý. Po uplynutí doby 500ms je vytvořen se sestupnou hranou impulz, který vyresetuje (nastaví log.0) @bit1, aby mohlo dojít k dalšímu časování stiskem tlačítka, jinak by byl na vstupu časovače log.1 a časovač by tak nezačal časovat nikdy.

9.3.4. Úloha 2.1: Tříbarevný maják

9.3.4.1. Cíl

Naučit se pracovat s analogovými výstupy.

9.3.4.2. Příklad

Navrhněte program pro tříbarevný zobrazovač HL22, to jsou výstupy AO.0-2 tak, aby se postupně rozsvítila každá barva. Můžete použít libovolný snímač či tlačítko na řídícím systému jako spouštění libovolné barvy.

! Pozor: zobrazovač není schopen zobrazit více barev najednou.

! Pozor: řídíme digitální prvek pomocí analogových výstupů.

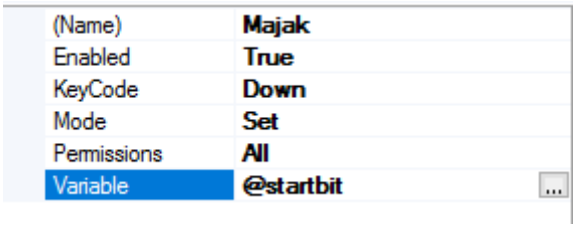
! Pozor: výstup je převodem z fyzikální veličiny.

9.3.4.3. Hardwarové řešení úloh

Maják je připojen na výstupy AO0.0-2, které jsou převedeny na dig. podobu (viz AO amplifier str.34).

9.3.4.4. Softwarové řešení úloh

Spuštění cyklu pro řízení barev budeme realizovat podobně jako v minulé úloze. Použijeme příkaz „KeyBit“ a nastavíme ho tak, aby při stisku setoval (nastavil log.1) na bit @startbit. Později se tento @startbit vyresetuje softwarově, aby nedocházelo, že se cykly budou spouštět několikrát.

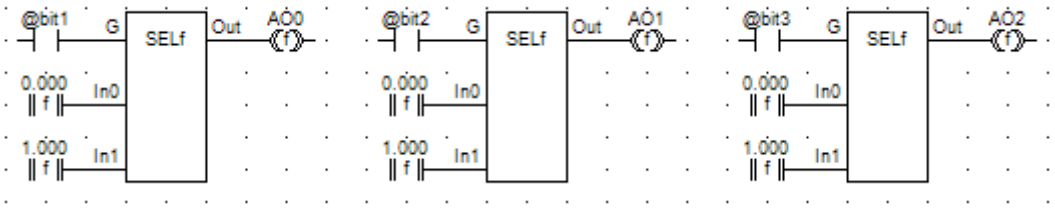


Obrázek 88: Parametrizace objektu "KeyBit"

Jelikož zapisujeme do AO, tak musíme použít příkaz AnOut. Buď použijeme AnOutLA pro RS (reléové schéma), anebo AnOut pro ST (strukturovaný text). V našem případě budeme mít dva procesy (RS – řízení, ST – zápis). AnOut převede fyzikální jednotku na elektrickou a zapíše ji na výstup. Zpracovává data z proměnné, ve které je hodnota, co se má zapsat.

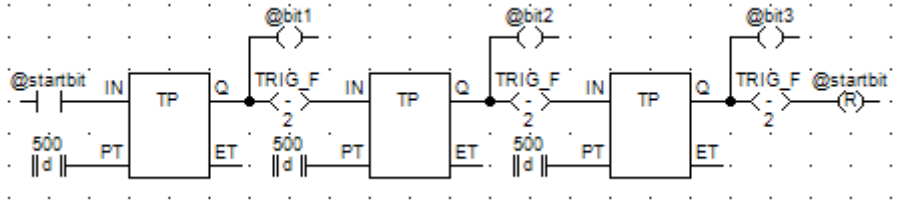
```
AnOut #AO00_0, AO0, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_1, AO1, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_2, AO2, 10.000, 0.000, 10.000, 0.000, 1.000
```

Analogový výstup je realizován pomocí PWM, ale my si postačíme s MIN a MAX (0% a 100% střídy). Jelikož z cyklu pro zapínání barev, budeme mít digitální hodnotu, ale AnOut pracuje s analogovou hodnotou, použijeme modul SELf, který vybere jednu ze dvou hodnot pomocí řídícího bitu a uloží ji do proměnné, kterou si přečte AnOut.



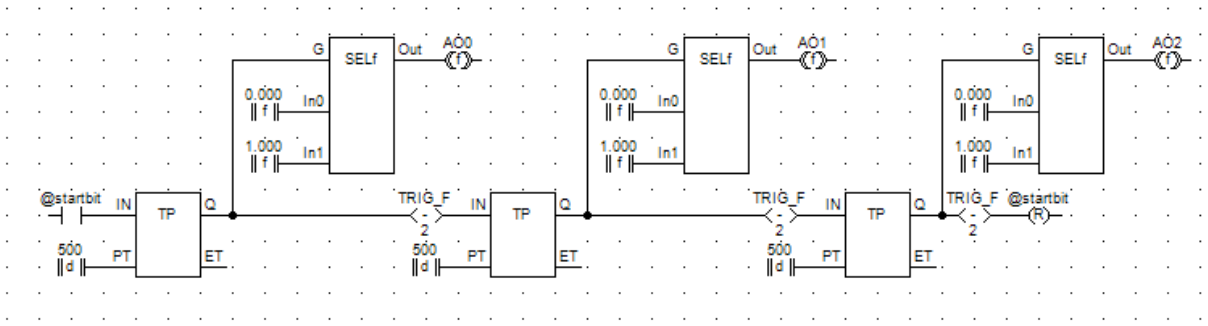
Obrázek 89: Zápis hodnoty do proměnné Float

Následně vytvoříme cyklus, který bude jednotlivé barvy spouštět, tedy bude přepínat hodnoty proměnných pro AnOut pomocí SELf.

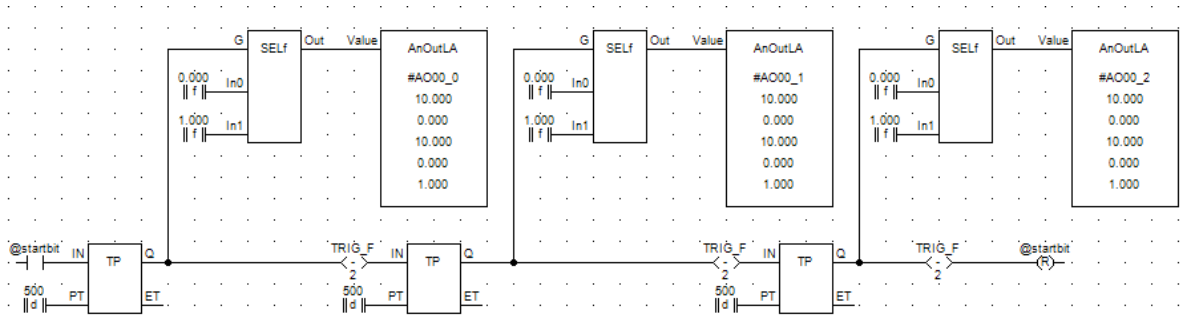


Obrázek 90: Cyklus řízení barev

@startbit zapne časování čítače „TP“, který je nastaven na dobu 500ms. Po tuto dobu bude na výstupu „TP“ log.1 a bude v bitu @bit1 uložena. Po tom, co první časovač „TP“ vypne, tak se změní @bit1 na log.0 a „TRIG_F“ nám vytvoří hranu se seběžnou hranou prvního čítače „TP“. Tento impulz nám nastaví čítání druhého čítače. A takto to pokračuje. Celý cyklus se nesmí spustit znovu, dokud nedoběhne, jelikož může svítit pouze jeden segment majáčku. Pokud je tedy @startbit nastaven do log.1, tak se první časovač nespustí znovu. Až dočasuje třetí čítač, tak s jeho seběžnou hranou je generován impulz, který resetuje bit @startbit a je možné pak spustit cyklus znovu. Pro přehlednost je možné spojit cyklus a zápis do proměnné pro „AnOut“.



Obrázek 91: Program spojený



Obrázek 92: Řízení celého majáku

Řízení celého majáku lze i docílit pomocí *jednoho pomocného bitu*, kde zapisování do proměnné nahradíme přímo zápisem do výstupu.

9.3.5. Úloha 2.2: Tříbarevný zobrazovač

9.3.5.1. Cíl

Obsluhovat tříbarevný zobrazovač pro indikaci výstupu vršků.

9.3.5.2. Příklad

Navrhněte program pro řízení tříbarevného zobrazovače HL23, to jsou výstupy AO.3 a DO.0-1, můžete použít libovolné snímače, nebo tlačítka řídícího systému pro spouštění barev. Zobrazovač je schopen rozsvítit všechny barvy zároveň.

! Pozor: řídíme digitální prvek pomocí analogových výstupů.

! Pozor: výstup je převodem z fyzikální veličiny.

9.3.5.3. Realizace

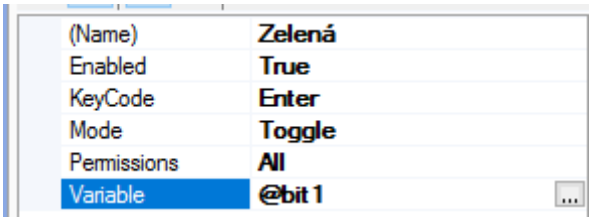
Zelená barva zobrazovače je připojena na AO0.3, modrá na DO0.0 a červená na DO0.1.

9.3.5.4. Program

Jelikož barvy můžeme spínat i všechny, tak využijeme tlačítek na terminálu řídícího systému. Musíme ale tyto hodnoty zapisovat do výstupů pomocí „AnOut“ a „DigOut“.

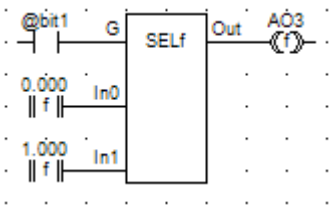
```
AnOut #AO00_3, AO3, 10.000, 0.000, 10.000, 0.000, 1.000
DigOut vystupyD, #0, 0x0000
```

Vytvoříme obrazovku, do které umístíme třikrát „KeyBit“, každý pro ovládání jedné barvy. „KeyBit“ nastavíme na toggle, tedy aby pouze měnil hodnoty bitu při stisku tlačítka. Pro barvy, které jsou připojeny na digitální výstup nastavíme bit proměnné, která je zapisovaná pomocí „DigOut“ na výstupy.



Obrázek 93: Parametrizování tlačítka terminálu

Zelená barva je připojena na AO0.3, tedy na analogový výstup a podle toho musíme uzpůsobit program, jelikož je převod na fyzikální jednotky. SELf nám poslouží opět pro výběr z jedné z proměnné (konstanty). Následně se hodnota AO3 zapíše do analogového výstupu (AnOut).



Obrázek 94: Výběr proměnné (konstanty) pomocí SELf

9.3.6. Úloha 3: Řízení krokového motoru

9.3.6.1. Cíl

Řídit krokový motor M21 pomocí budiče A24 a vyzkoušet si změnu směru otáčení, rychlost a brzdu.

9.3.6.2. Příklad

Navrhněte program pro řízení krokového motoru, to jsou výstupy DO.2-4. Vyzkoušejte si měnit rychlost, směr otáčení, vypnutí brzdy a zobrazte údaje na terminálu. Pokud motor poběží, tak bude svítit modré světlo tříbarevného zobrazovače (DO0.0)

? Nápověda: modul „FreqOut“ tvoří frekvenci na výstupu digitálního výstupu.

! Pozor: motor je řízen pomocí driveru, který je řízen frekvencí.

! Pozor: maximální použitelná frekvence, kterou může AMiNi2D poskytnout, je 1500 Hz.

9.3.6.3. Realizace

Motor je připojen pomocí budiče (Driveru), který řídí rychlost motoru v závislosti na frekvenci. Budič podporuje i mikrokrokování, které je nutné, jinak motor vibruje a jelikož jsou vršky velmi lehké, tak začnou všelijak jezdit po páse. Mikrokrokování a proud motorem se nastavuje přímo na driveru pomocí kódových přepínačů.

9.3.6.4. Program

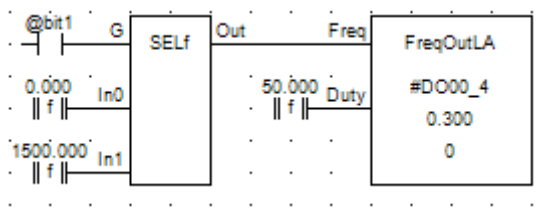
Vytvoříme obrazovku, ve které budou „KeyBit“ pro řízení. Jeden bude řídit směr, druhý brzdu a třetí rychlost. Pro „KeyBit“ nastavíme funkci toggle, abychom mohli pozorovat chování motoru v různých situacích.

(Name)	**Motor**
Enabled	**True**
KeyCode	**Plus**
Mode	**Toggle**
Permissions	**All**
Variable	**@bit 1**

Obrázek 95: Parametrizace "KeyBit"

Řízení směru a brzdy měníme bit přímo v proměnné pro zápis digitálních výstupů.

Pomocí bloku SELf budeme vybírat rychlost motoru. Rychlosti jsou nastaveny jako konstanty (0: motor stojí, 1500: maximální výstupní frekvence řídícího systému). Rychlosti se vybírají pomocí bitu @bit1. „Duty“ je střída výstupu v procentech a nastavíme ji na polovinu, tedy na 50 %.



Obrázek 96: Frekvenční výstup pomocí "FreqOutLA"

9.3.7. Úloha 4: Třídění vršků podle kontrastu

9.3.7.1. Cíl

Zjistit, zda vršek obsahuje těsnění, či nikoli a vytřídit je.

9.3.7.2. Příklad

Navrhněte program pro třídění vršků pomocí kontrastního senzoru B23 (O5K500). Výstup tohoto senzoru je zapojen na vstup DI0.2. Kontrastní senzor nastavte tak, aby spínal jen tehdy, když vršek bude mít těsnění, pokud nebude mít těsnění, tak senzor nebude sepnutý. Budou se vyhazovat vršky bez těsnění solenoidem Y21 (DO0.6). Motor pojede konstantní rychlostí a pozastaví se pouze tehdy, když bude se vršek třídit. Pokud vršek bude dobrý, tak dopravník pojede dál.

! Pozor: pokud vršek není tříděný, senzor vezme i druhou hranu vršku.

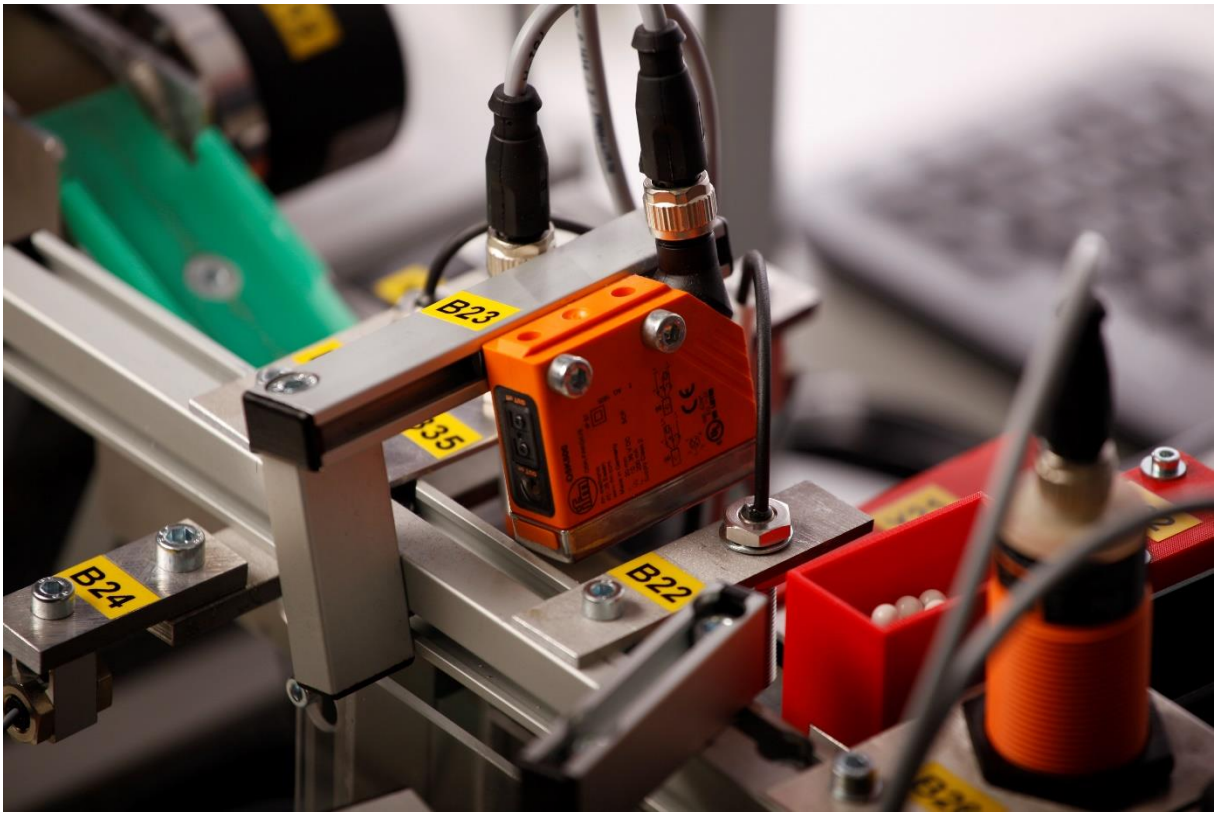
? Nápověda: pokud je vršek netříděný, počítejte hrany.

? Nápověda: Učení senzoru se provádí vložením předmětu, podržením tlačítka ON, dokud nezačne blikat (2 Hz) (senzor bude spínat na tento kontrast), poté se změní pozadí a podrží se tlačítko OFF, dokud signalizační led nezhasne. Tlačítka musí být stisknuta po dobu 2 s.

! Pozor: pokud signalizační led bliká rychle (8 Hz), nastavení senzoru je chybné. Může být zapříčiněno malým rozdílem mezi kontrasty.

9.3.7.3. Realizace

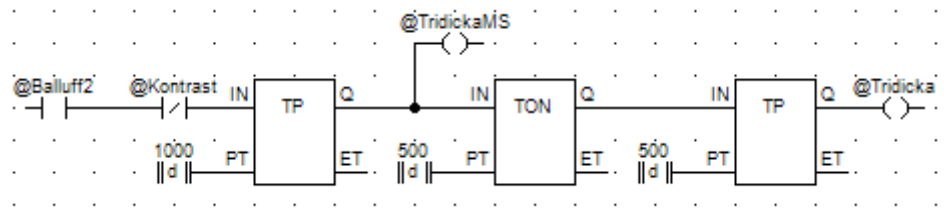
Solenoid pro třídění je spínaný pomocí relé, které je zapojeno na výstup DO0.5. Senzor „BES M08MG-GSC20B-BP03“ pro polohu vršku, zapojený na DI0.1, sepne, když je na místě vršek pro třídění (Vršek má ale dvě hrany). Senzor kontrastu snímá střed vršku, když je vršek v poloze pro třídění.



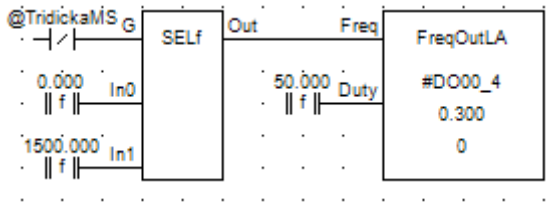
Obrázek 97: Kontrastní senzor

9.3.7.4. Program

Vytvoříme algoritmus (cyklus) pro třídění. Pokud bude ve vršku těsnění (kontrastní senzor sepne), tak vršek pojede dál a pokud bude bez těsnění, dojde k jeho vytřídění. Tedy pokud vršek vyhovuje nic se neděje, pokud vršek nevyhovuje zastaví motor, počká, než se motor zastaví, vršek vytřídí, a poté se zapne motor.

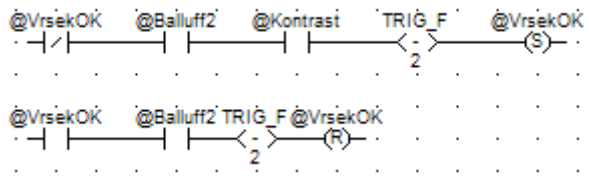


Obrázek 98: Cyklus třídění podle kontrastu



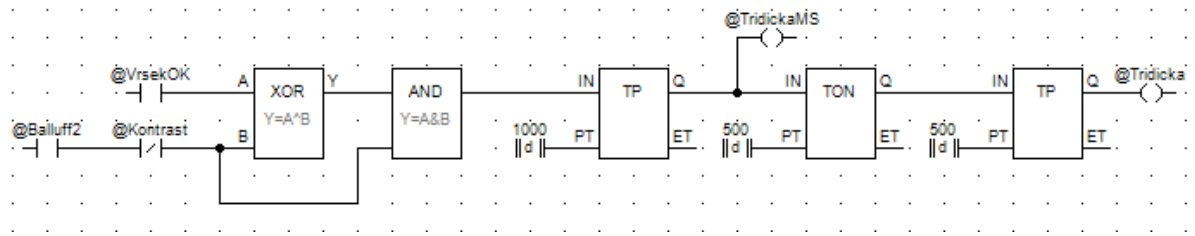
Obrázek 99: Řízení motoru pro třídění

Nyní již třídíme vršky, které nemají těsnění. Problém nastane tehdy, když je vršek s těsněním a pojede dále. Senzor zaznamená i druhou hranu vršku, cyklus se spustí znovu a bude se třídit i s druhou hranou. Musíme tedy zajistit, aby když je vršek dobrý, nereagoval na druhou hranu.



Obrázek 100: Řešení dvou hran

A cyklus třídění bude vypadat následovně:



Obrázek 101: Cyklus třídění s dobrým vrškem

9.3.8. Úloha 5: Třídění vršků podle otočení

9.3.8.1. Cíl

Zjistit otočení vršku a špatné vršky vytřídit.

9.3.8.2. Příklad

Navrhněte program pro třídění vršků. Snímač B21 (DI0.0) snímá natočení vršků a optozávora B24 (DI0.3) udává, zda je vršek v pozici, kdy se může snímat otočení vršku. Otočené vršky se budou třídit solenoidem Y21 (vyhazovat) (DO0.6). Navažte na předchozí úlohu a doplňte i třídění podle kontrastu.

! Pozor: řídící systém si musí zapamatovat pořadí vršků a postupně je počítat. Použijte proměnnou, nebo matici pro ukládání pořadí.

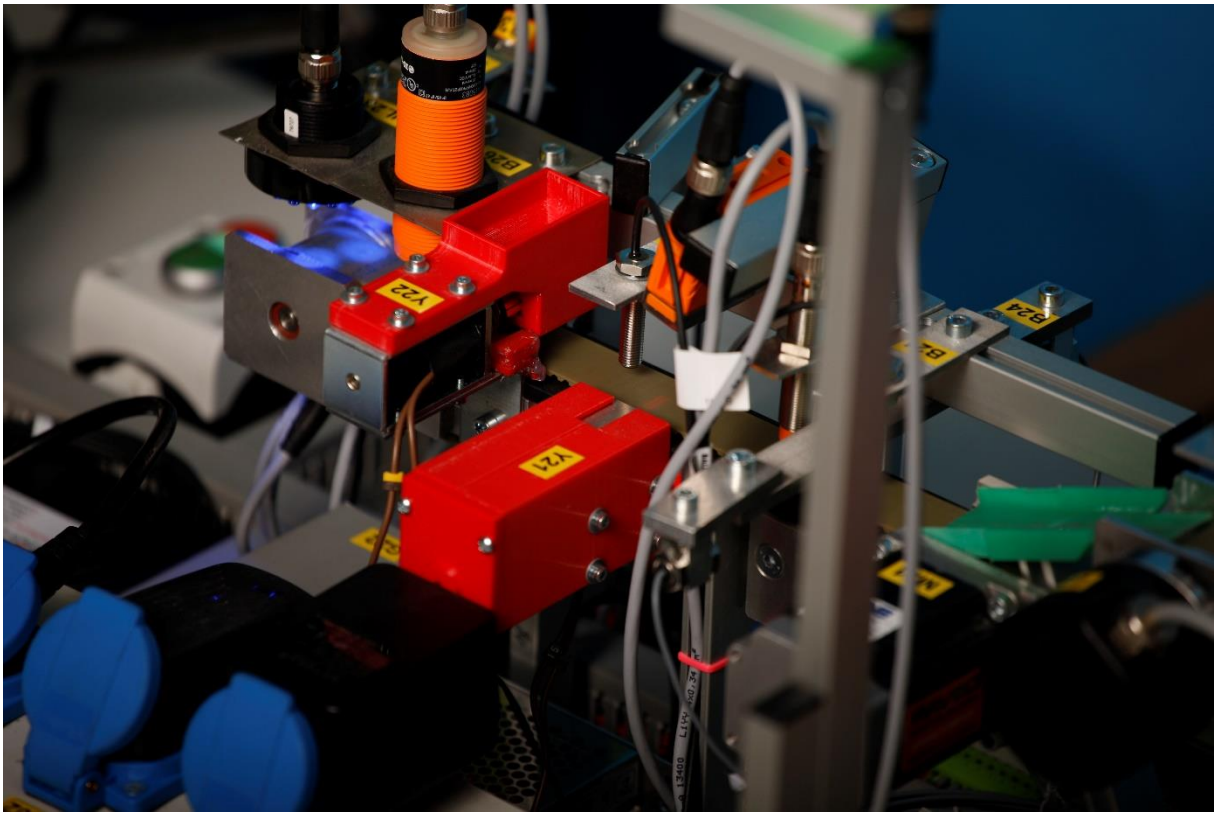
? Nápověda: můžeme se dotázat na jednotlivé bity proměnných (políček matic) pomocí jiné proměnné.

! Pozor: musí být provedení přetypování proměnné (příkaz „bool.()“) pokud ukládáme hodnotu do aliasu z políčka matice.

! Pozor: po zapnutí stanice musí dojít k nulování proměnných pro řízení zápisu a čtení z proměnné.

9.3.8.3. Realizace

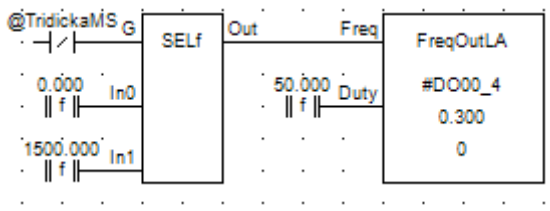
Snímač natočení vršku je nastaven tak, že pokud se přeruší paprsek optozávory, tak je senzor B21, „BES M08MG-GSC20B-BP03“ pro snímání otočení vršku uprostřed vršku. Vršek pak musí ale urazit nějakou dráhu, aby se dostal ke třídičce a mezitím může být snímán i jiný vršek.



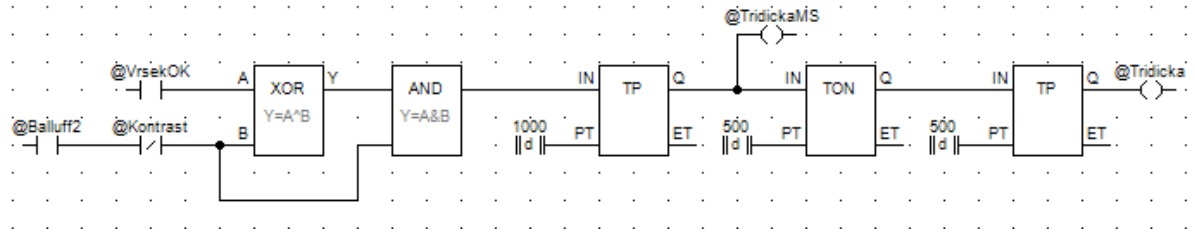
Obrázek 102: Třídička s optozávorou

9.3.8.4. Program

Vytvoříme cyklus stejně jak v předchozí úloze, tedy:

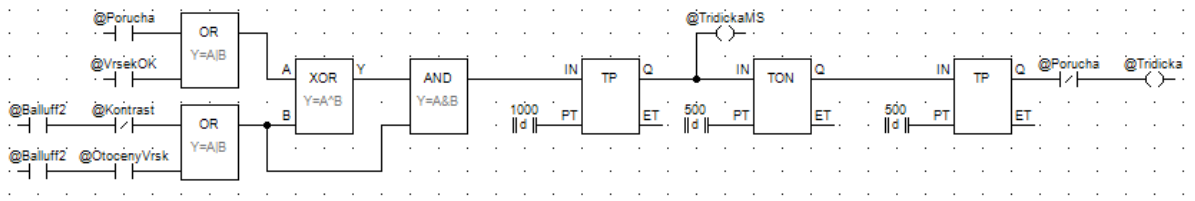


Obrázek 103: Zastavení motoru při třídění

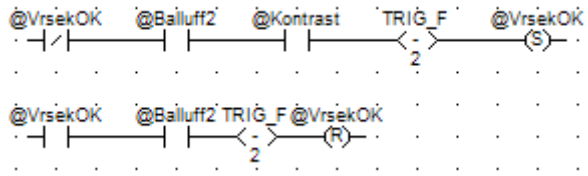


Obrázek 104: třídění vršku podle kontrastu

Jelikož je mezi třídičkou a rozpoznáváním natočení vršku mezera, tak musíme zabezpečit, aby byly informace o dodání vršku správné. Tuto mezeru můžeme vyplnit pomocí matice (lepší představa, jak to funguje), nebo proměnná typu Intiger. V matici se můžeme pohybovat i pomocí proměnných. Bude tedy matice o velikosti 1*16. Do každého políčka jsme schopni uložit proměnnou intiger. Pokud se budeme v matici pohybovat pomocí proměnné intiger, tak nesmíme zapomenout, že intiger, má mnohem více kombinací nežli 16 a proto bychom měli zabezpečit přetečení, tedy abychom neudávali políčko 20, když jich je jen 16. Nebo bychom mohli zvolit místo matice proměnnou intiger a udávat, který bit přečíst, a do kterého bitu zapsat pomocí další proměnné. Princip je stejný, jen se změní názvy.



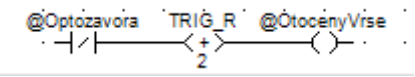
Obrázek 105: Cyklus třídění



Obrázek 106: řešení dvou hran s otočeným vrškem

Třídění tedy záleží nyní na kontrastu a otočeném vršku. Nyní je ještě třídění nepoužitelné pro otočené vršky, jelikož chybí paměť o natočení vršků. Ale pokud je vršek natočen, tak se počítá s tím, že se nečeká na druhou hranu vršku, to stejné i pokud je vytříděn v závislosti na přítomnosti těsnění.

Aby nedocházelo k opakovanému zápisu, po kterou bude vršek snímán, tak vytvoříme algoritmus, který vytvoří impulz pouze při příjezdu pod senzor natočení (optozávora je přerušena).



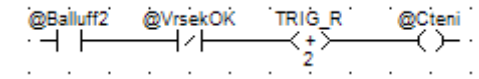
Obrázek 107: Snímání otočení vršku

Nyní již víme, kdy se má zapsat otočení vršku do pole matice a potřebujeme v RS skript pro zápis v ST. Budeme v něm i inkrementovat proměnnou pro paměť, která nám určuje pořadí vršků. Musíme ohlídat i přetečení 15 (matice má souřadnice pro sloupce 0, 1, 2, …, 15). Řádek matice je pouze jeden

```
If  @OtocenyVrse
|  Let  MaticeOtVrse[0,MaOtVrWr] = @Balluff1
|  Let  MaOtVrWr = MaOtVrWr + 1
|  If  (MaOtVrWr == 16)
|  |  Let  MaOtVrWr = 0
|  EndIf
EndIf
```

Obrázek 108: Zápis natočení vršku do matice

Nyní již jsme si uložili informace o natočení vršku a musíme ji přečíst a zpracovat.



Obrázek 109: Impulz pro čtení informace z matice

Vytvořili jsme impulz, který nám udává, kdy se bude číst z matice, tedy je vršek pod třídičkou a zamezili jsme, aby vršek to načetlo i při druhé hraně, pokud je vršek dobrý.

```
If  @Cteni
|  Let  @OtocenyVrsk = bool(MaticeOtVrse[0,MaOtVrRd])
|  Let  MaOtVrRd = MaOtVrRd + 1
|  If  (MaOtVrRd == 16)
|  |  Let  MaOtVrRd = MaOtVrRd + 1
|  EndIf
EndIf
```

Obrázek 110: Čtení informace z matice

Čtení z matice opět proběhne pouze jednou, jelikož je vytvořen volací impulz. Pokud nastane k vypnutí řídícího systému, kdy je vršek rozpoznán a není vytříděn, tak po zapnutí, kdy tam vršek již nebude, dojde k rozhození a nebude fungovat správně! Proto vytvoříme proces INIT, kde se při zapnutí řídícího systému dojde k vynulování proměnných.

```
Let MaOtVrRd = 0
Let MaOtVrWr = 0
```

! Pozor: proces INIT není název procesu, ale typ procesu. Tento proces se spustí pouze jednou, a to při zapnutí řídící stanice. Převážně v něm jsou uloženy konfigurace.

9.3.9. Úloha 6: Plnění vršků kuličkami

9.3.9.1. Cíl

Naplnit vršek požadovaným počtem kuliček.

9.3.9.2. Příklad

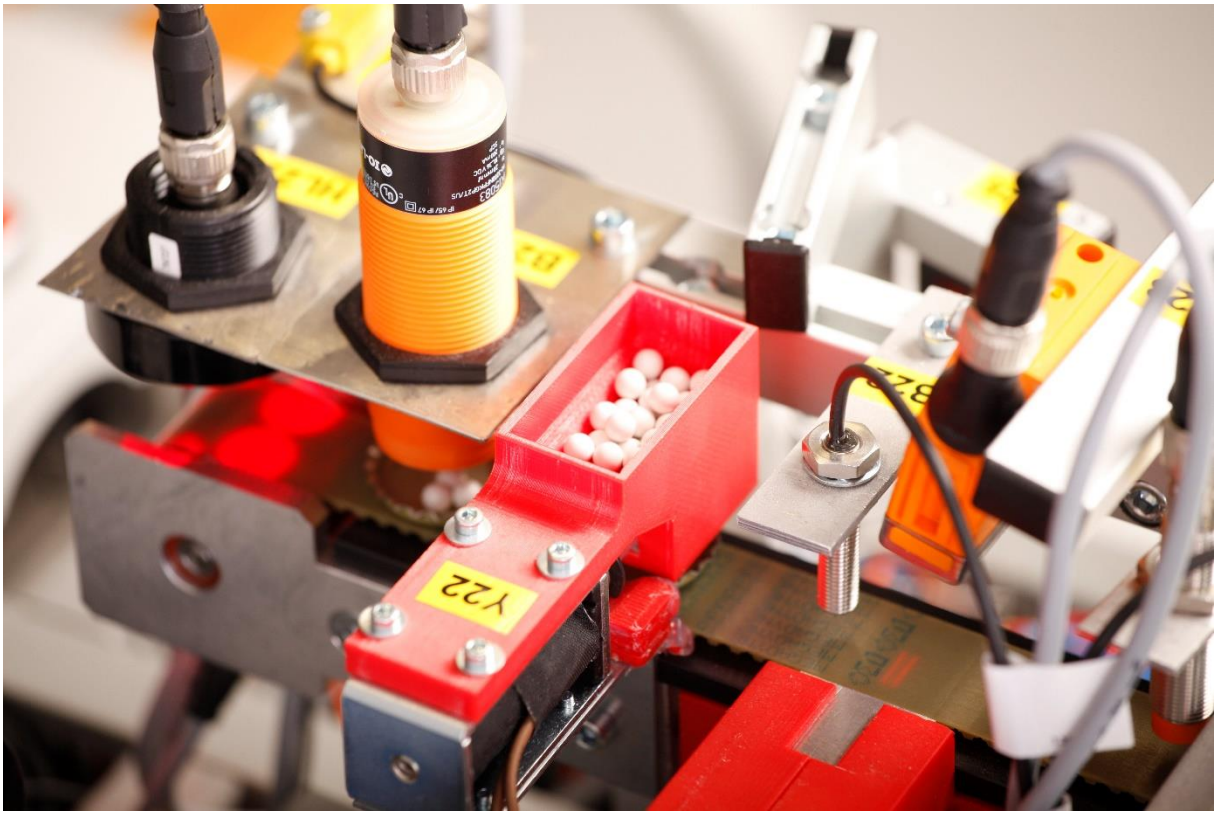
Navrhněte program pro plnění vršků kuličkami. Vršek se bude plnit třemi kuličkami. Vzdálenostní snímač B25, „QS18VP6LAF“ (DI0.4), snímá vzdálenost vršku a sepne tehdy, když je pod plničkou vršek. Dávkování kuliček se provádí solenoidem Y22 (DO0.5).

? Nápověda: dávkování je možno uskutečnit modulem „PulsOut“, který vytvoří definované pulzy.

! Pozor: počítejte s dobou potřebnou pro přemístění kuličky.

9.3.9.3. Realizace

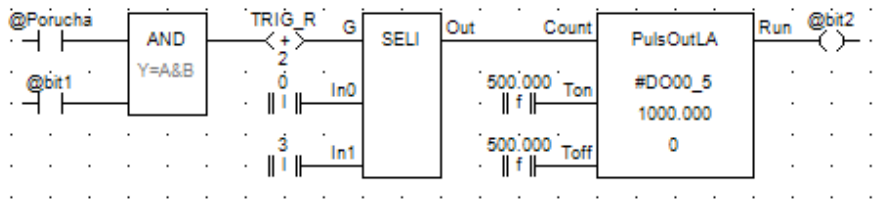
Vršek je snímán vzdálenostním senzorem „QS18VP6LAF“ (DI0.4), který určuje pozici vršku pod plničkou. Senzor je nastavitelný. Plnička má dávkovač na jednu kuličku, takže jde vršek naplnit přesným počtem kuliček.



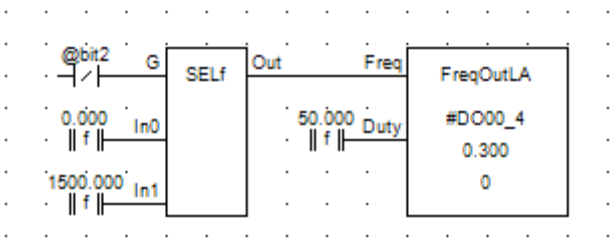
Obrázek 111: Plnička na lince

9.3.9.4. Program

Budeme plnit vršek požadovaným počtem kuliček, v tomto příkladu třemi kuličkami. Použijeme blok „PulsOutLA“, který vygeneruje na digitální výstup požadovaný počet impulsů. Nastavíme délky jednotlivých impulzů a maximální délku obou impulsů. Modul by měl číst vstup každý cyklus, takže vytvoříme pouze krátký impuls na jeden cyklus. Pro výběr kolika kuličkami budeme plnit je výběr proveden pomocí SELi. Modul má i výstup, který lze použít pro zastavení motoru, když budeme plnit.



Obrázek 112: Program pro plnění



Obrázek 113: Program pro řízení motoru

9.3.10. Úloha 7: Kontrola vršků, zda jsou naplněny

9.3.10.1. Cíl

Zpětná vazba, zda byly vršky naplněny, což může být zapříčiněno prázdným zásobníkem plničky.

9.3.10.2. Příklad

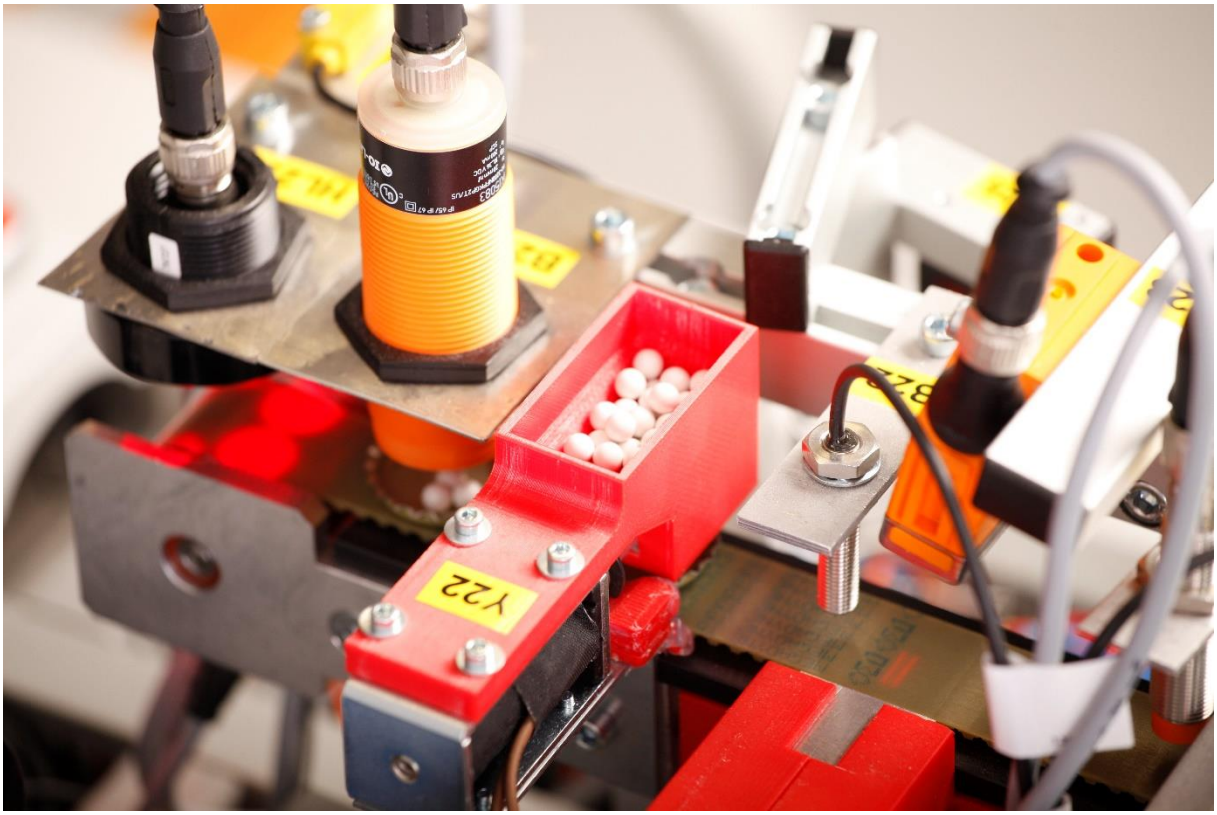
Navrhněte program pro kontrolu, zda byl vršek naplněn kuličkami. Můžete využít program z předchozího úkolu pro plnění vršků kuličkami. Indukční senzor B27 (DI0.6) určuje polohu vršku pod kapacitním senzorem B25 (DI0.5). Nastavte kapacitní senzor tak, aby sepnul, když bude vršek naplněn správným počtem kuliček. Pokud je vršek naplněn, rozsvítí se na chvíli zelené světlo tříbarevného zobrazovače HL23 (AO0.3) a jestli byl vršek nenaplněn, rozsvítí se na chvíli červené světlo tříbarevného zobrazovače HL23 (DO0.1).

? Nápověda: můžete si zvolit, zda kapacitní senzor bude spínat nebo rozpínat při správné kapacitě.

! Pozor: nastavení kapacity je velmi citlivé a pokud je rozdíl kapacit malý, dojde k chybě při nastavování.

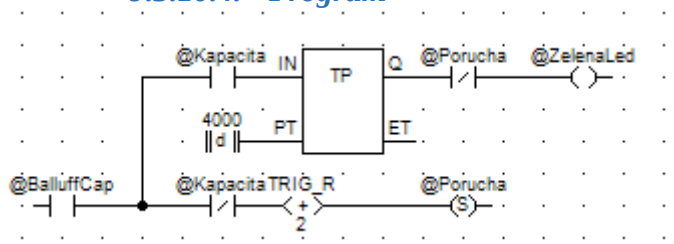
9.3.10.3. Realizace

Senzor „BES M08MG-GSC20B-BP03“ snímá pozici vršku pod kapacitním senzorem „KI5083“, který porovnává kapacitu vršku.



Obrázek 114: Uspořádání senzorů na lince

9.3.10.4. Program



Obrázek 115: Kontrola naplnění

Indukční senzor nám udává pozici, kdy je vršek pod kapacitním senzorem. Poté musíme zjistit stav kapacitního senzoru. Pokud bude vršek dobrý, tak se spustí časovač a na 4 sekundy se rozsvítí zelená led na konci dopravníku (viz. Tabulka str.79). Pokud vršek není naplněn, tak se nastaví porucha a linka se zastaví (proces pro řízení motoru). Poruchu bude muset vynulovat obsluho pomocí terminálu řídícího systému.

Nastavíme tlačítko „KeyBit“ na „clear“.

(Name)	**Porucha**
Enabled	**True**
KeyCode	**Enter**
Mode	**Clear**
Permissions	**All**
Variable	**@Porucha**

Obrázek 116: Parametrizace tlačítka pro resetování poruchy

9.3.11. Úloha 8: Simulace poruchy

9.3.11.1. Cíl

Chování linky při poruše vyvolané vnějším podmětem, anebo nesprávností úlohy.

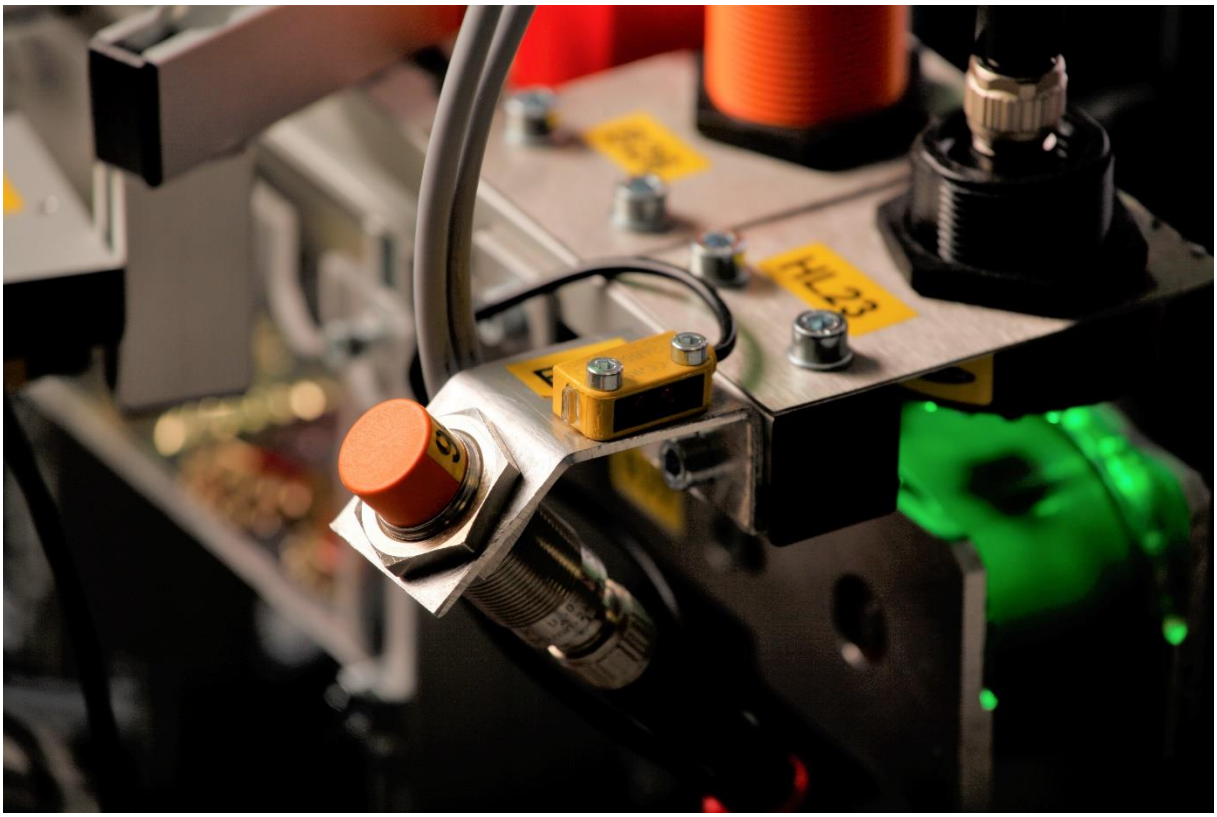
9.3.11.2. Příklad

Navrhněte program, kde použijete snímač B36 (AI0.7) pro vytvoření poruchy a senzor B28 (DI0.7) pro vytvoření odstávky. Při poruše se vypne brzda motoru, dokončí všechny úkony, zneaktivní se všechny ostatní funkce a bude svítit červeně tříbarevný majáček. Při odstávce se dokončí všechny úkony, brzda motoru bude aktivní a bude svítit žlutě tříbarevný majáček. Pokud linka bude v provozu, bude svítit zeleně tříbarevný majáček. Porucha nebo odstávka se zruší pomocí tlačítek na terminálu AMiNi2D.

! Pozor: hodnota senzoru je uložená do proměnné, se kterou musíme následně zpracovat pro další použití.

9.3.11.3. Realizace

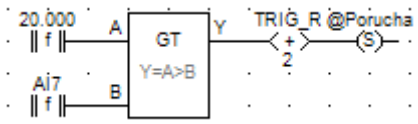
Analogový indukční senzor „IG6084“ a optický vzdálenostní „Q12AB6FF30Q“.



Obrázek 117: Umístění senzorů

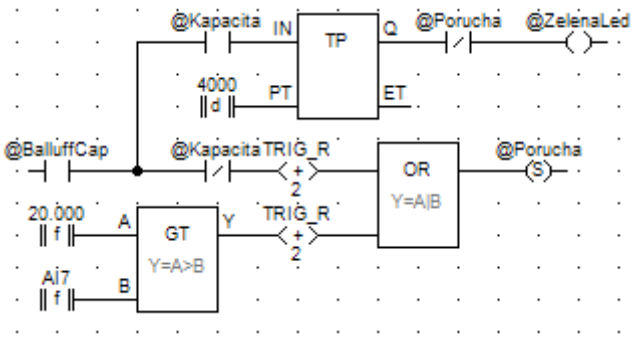
Poruchu budeme simulovat pomocí analogového senzoru, je tedy číst požadovaný vstup a porovnávat jej s konstantou. V příkladu je konstanta 20 a rozsah senzoru je 0-100% ve fyzikálním převodu, tedy bude porucha se nastaví, pokud bude výstup senzoru na 20%.

9.3.11.4. Program



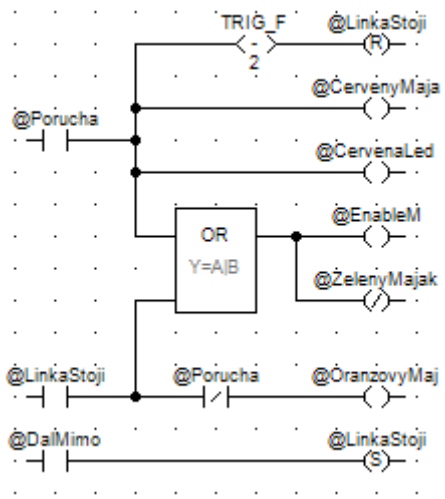
Obrázek 118: Nastavení poruchy podle analogového senzoru

Můžeme dodělat i nastavení poruchy podle kontroly plnění vršků.



Obrázek 119: Porucha v závislosti na senzoru a kapacitní kontrole naplnění

Pokud jsme již naprogramovali, kdy nastane porucha, tak ji musíme začlenit i tak, aby se něco dělo. Využijeme zobrazovače pro znázornění chodu. Pokud linka pojede, tak svítí zeleně majáček. Jestli linka má poruchu, tak svítí červeně a jestli bude mít odstávku, tak oranžově. Porucha má přednost před odstávkou, tedy pokud je odstávka, tak může vzniknout porucha, ale opačně nikoli.



Obrázek 120: Řízení zobrazovačů a odstávka pomocí vzdálenostního senzoru

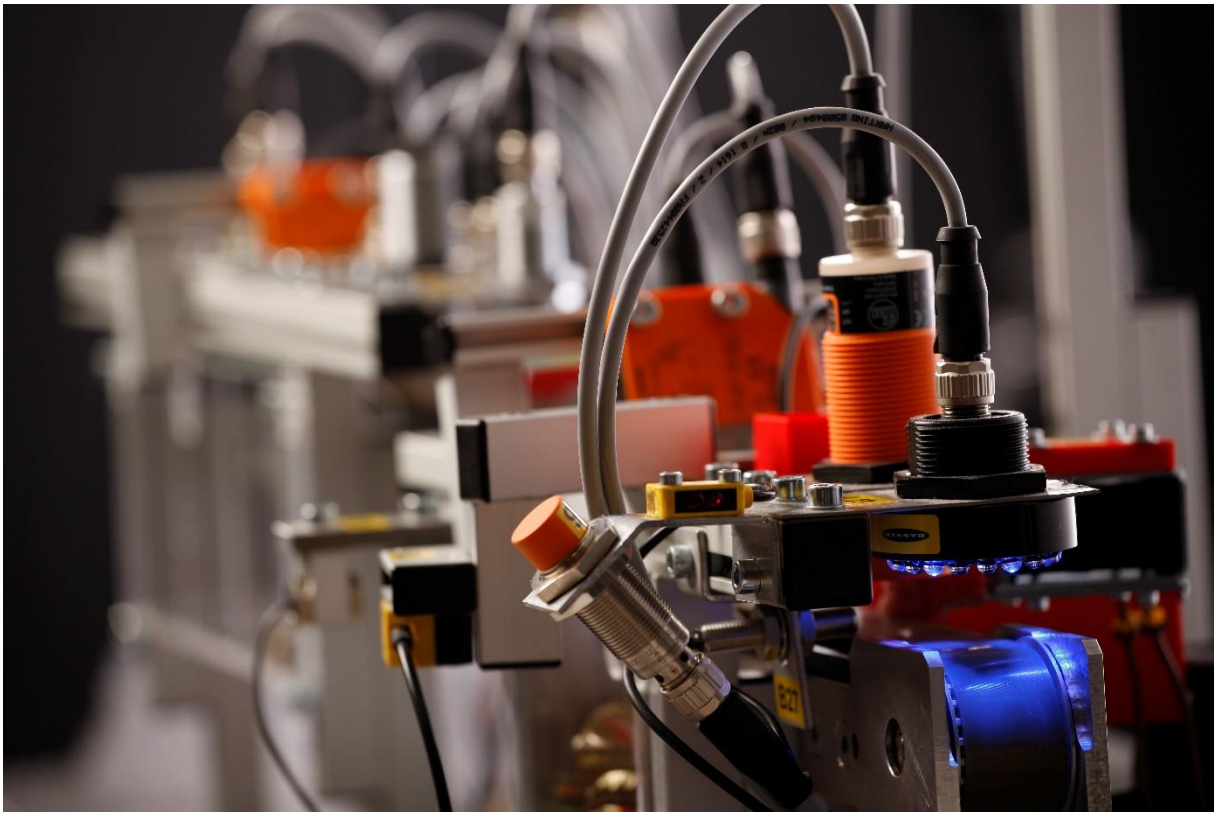
9.3.12. Úloha 9: Řízení celé linky

9.3.12.1. Cíl

Zkombinovat všechny možné úlohy a synchronizovat je.

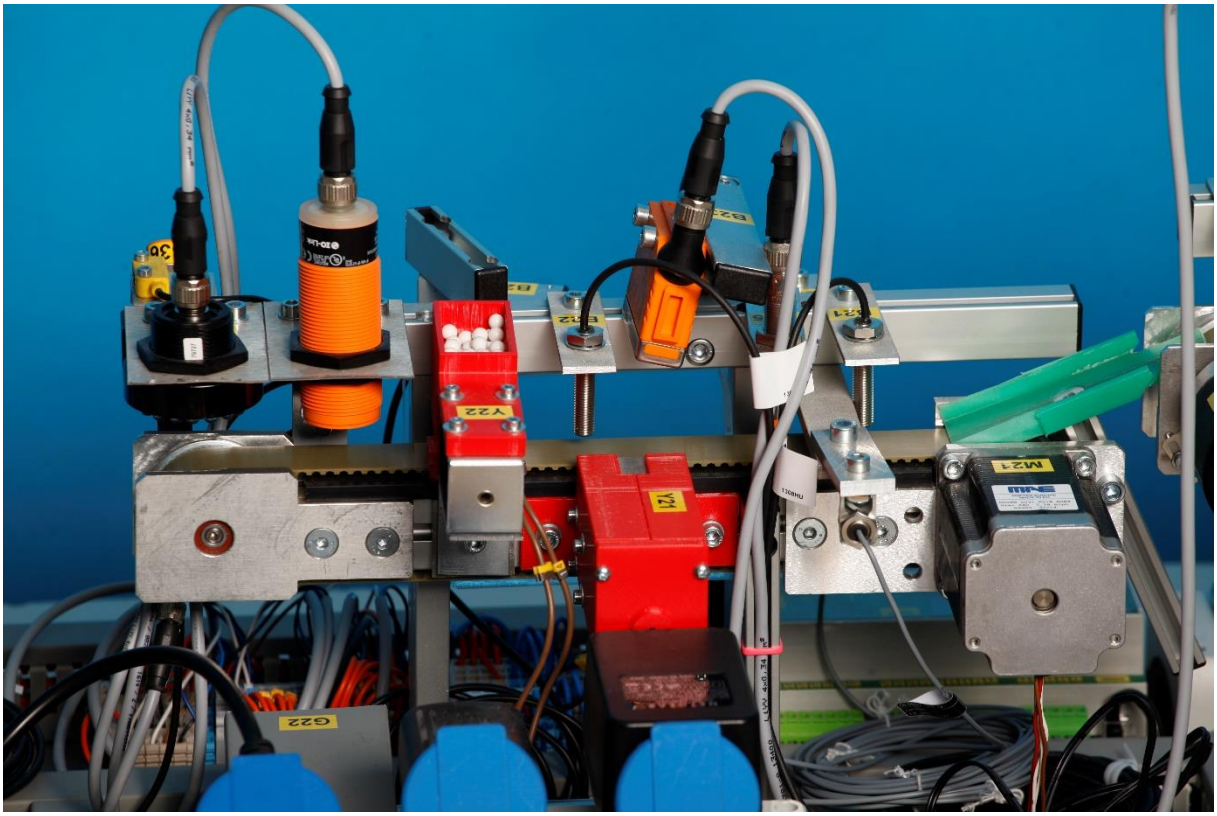
9.3.12.2. Příklad

Navrhněte program pro řízení celé linky. Využijte programů a znalostí z předchozích úloh. Linka bude třídit vršky podle natočení, a zda bude mít těsnění, plnit vršky a provádět kontrolu plnosti. Pokud nebude vršek naplněn, bude se hlásit porucha. Bude možno zapnout poruchu i odstávku stejně jako v úloze 8 (str. 98) a budou platit i stejná pravidla. Vršky se budou plnit třemi kuličkami.

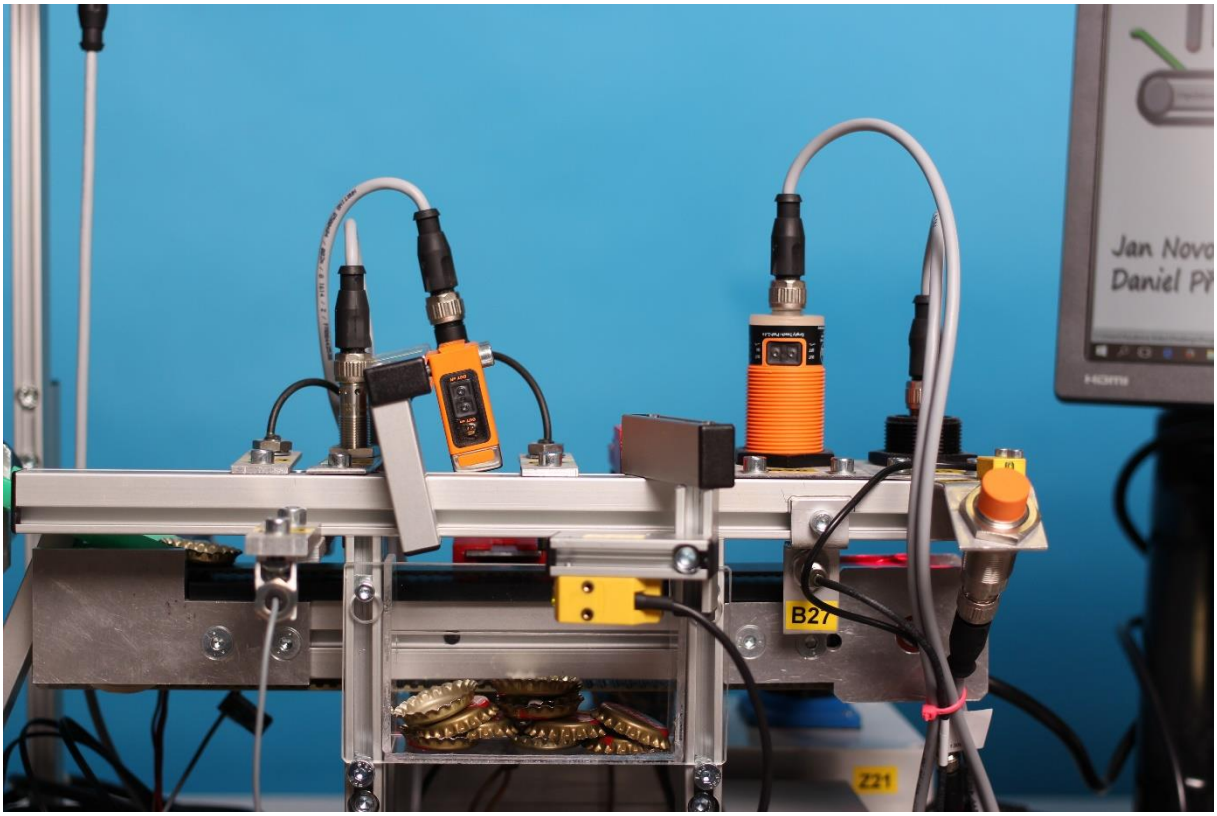


Obrázek 121: Pohled na dopravník linky 2

9.3.12.3. Realizace



Obrázek 122: Pohled na dopravník zezadu



Obrázek 123: Pohled na dopravník zepředu

9.3.12.4. Program

----------Proces INIT----------

```
Let MaOtVrRd = 0
Let MaOtVrWr = 0
Let MotorSpeed = 1500
let @LinkaStoji = false
Let @porucha = false
```

----------Konec----------

Nastaví se zde při zapnutí řídícího systému potřebné proměnné, tedy vyresetují poruchy, nastaví rychlost motoru na „základní“ a nastaví se proměnné pro třídění do nuly, aby nedošlo ke špatnému počítání, když je linka zastavena a vršky z ní odebrány.

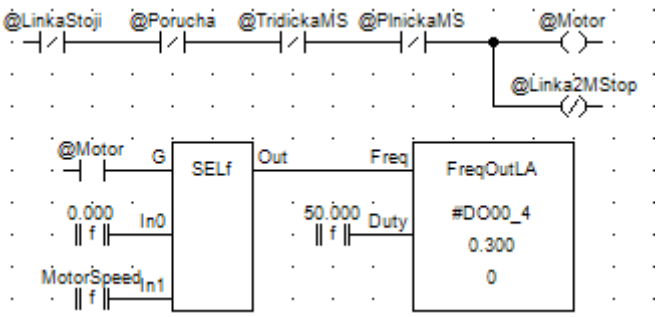
----------Proces čtení vstupů a výstupů----------

```
DigIn #0, vstupyD, 0x0000

DigOut vystupyD, #0, 0x0000

AnIn #AI00_6, AI6, 10.000, 0.000, 10.000, 0.000, 100.000
AnIn #AI00_7, AI7, 10.000, 0.000, 10.000, 0.000, 100.000

AnOut #AO00_0, AO0, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_1, AO1, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_2, AO2, 10.000, 0.000, 10.000, 0.000, 1.000
AnOut #AO00_3, AO3, 10.000, 0.000, 10.000, 0.000, 1.000
```

----------Konec----------

Dojde k přečtení vstupů a k zápisu výstupů.

----------Proces pro řízení motoru----------

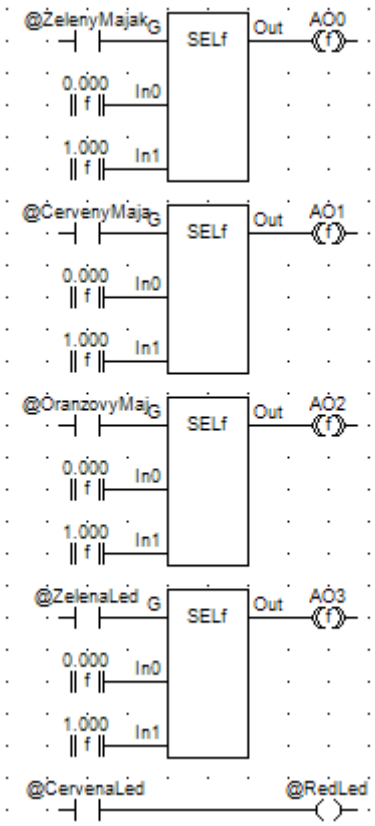


Obrázek 124: RS-Řízení motoru

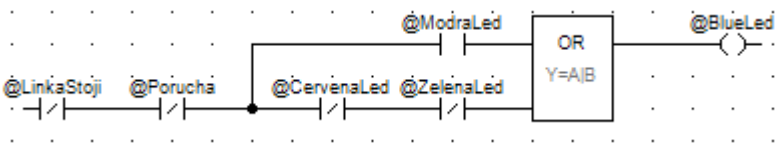
----------Konec----------

Řízení motoru pomocí modulu „FreqOutLA“

----------**Proces pro zápis do proměnných pro analogové výstupy**----------



Obrázek 125: RS-Analogové výstupy

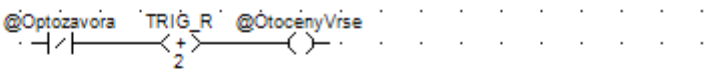


Obrázek 126: RS-Algoritmus pro modrou ledku

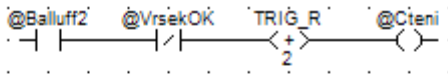
----------**Konec**----------

Pro zápis do analogového výstupu je zapotřebí vytvořit z bitu hodnotu proměnné, tak použijeme modul „SELf“, který vybere podle vstupního bitu konstantu, která je zapsaná do proměnné, která udává hodnotu analogového výstupu.

----------**Proces pro třídění**----------

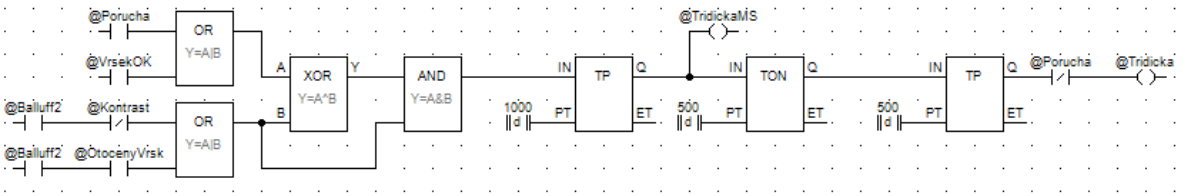



```
If  @OtocenyVrse
|  Let  MaticeOtVrse[0,MaOtVrWr] = @Balluff1
|  Let  MaOtVrWr = MaOtVrWr + 1
|  If  (MaOtVrWr == 16)
|  |  Let  MaOtVrWr = 0
|  EndIf
EndIf
```

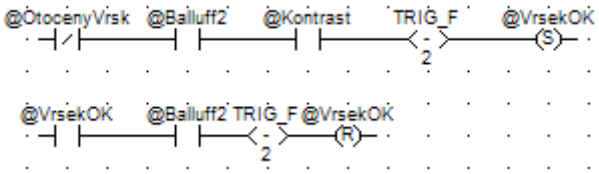



```
If  @Cteni
|  Let  @OtocenyVrsk = bool(MaticeOtVrse[0,MaOtVrRd])
|  Let  MaOtVrRd = MaOtVrRd + 1
|  If  (MaOtVrRd == 16)
|  |  Let  MaOtVrRd = MaOtVrRd + 1
|  EndIf
EndIf
```

Obrázek 127: ST a RS-algoritmus pro pamatování vršků



Obrázek 128: RS-Třídění

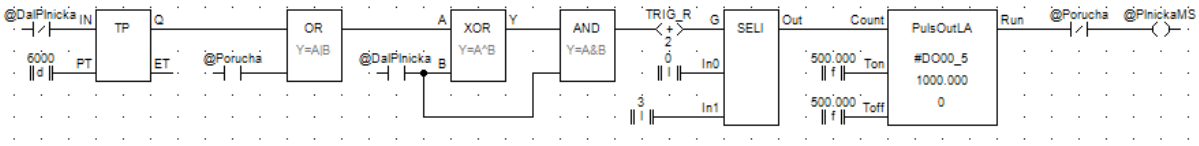


Obrázek 129: RS-Reakce na druhou hranu vršku

----------**Konec**----------

Třídění vršků podle natočení a podle kontrastu je dohromady, jelikož se jedná o třídění stejnou třídičkou. Bude se tedy třídit na základě proměnné, zda je vršek natočen, či zda je vršek bez těsnění. Pro třídění podle natočení je nutno si pamatovat kolikátý vršek je na třídění a využijeme na to matic. Musíme si dávat i pozor na to, že vršek má buď jednu hranu (dno), anebo dvě hrany. To nám určí natočení vršku. A pak je nutný algoritmus, který bude naměřená data vyhodnocovat a řídit třídění.

----------Proces pro plnění vršků----------

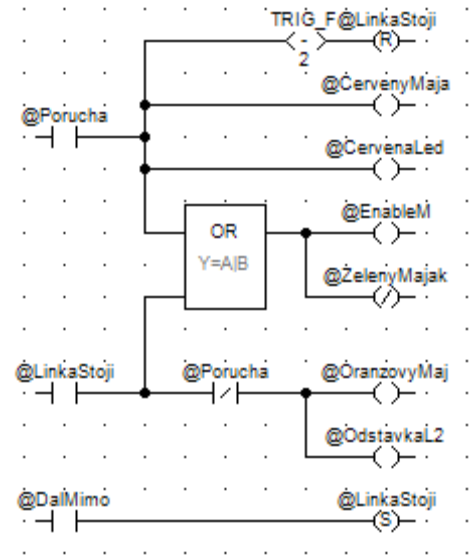


Obrázek 130: RS-Plnění vršků

----------Konec----------

Vršek pod plničkou, snímá senzor B25. Senzor snímá bok vršku, takže nemusíme ošetřit druhou hranu. Po ošetření vstupů zavoláme modul „PulsOutLA“, který nám vytvoří požadovaný počet pulzů, který jsme do něj přesunuli. Modul má výstup „RUN“, který použijeme k zastavení motoru, když se vršek plní.

----------Proces kontroly plnosti vršků a pro poruchy----------



Obrázek 131: RS-Kontrola vršků

----------Konec----------

Kontrola vršků probíhá okamžitě a není potřeba zastavovat linku, pokud je vršek dobrý. Jestli vršek nebyl naplněn, vytvoříme poruchu, která zastaví linku a rozsvítí červená světla. Doplníme i senzory B28 pro simulaci odstávky a senzor B30 pro simulaci poruchy.

9.3.13. Úloha 10: Uživatelské rozhraní

9.3.13.1. Cíl

Naučit se pracovat s terminálem a ovládání obsluhou.

9.3.13.2. Příklad

Navrhněte program s uživatelským rozhraním. Použijte program z úlohy 9. Vytvořte harmonogram mezi obrazovkami, aby se dalo mezi nimi přepínat. Pro plnění vytvořte obrazovku, kde bude zobrazeno plnění a editace, kolika kuličkami se bude plnit. Pro motor vytvořte to stejné a bude se nastavovat rychlost motoru. Další obrazovky budou pro potvrzení poruchy a odstávky. Jedna obrazovka bude zahrnovat informace.

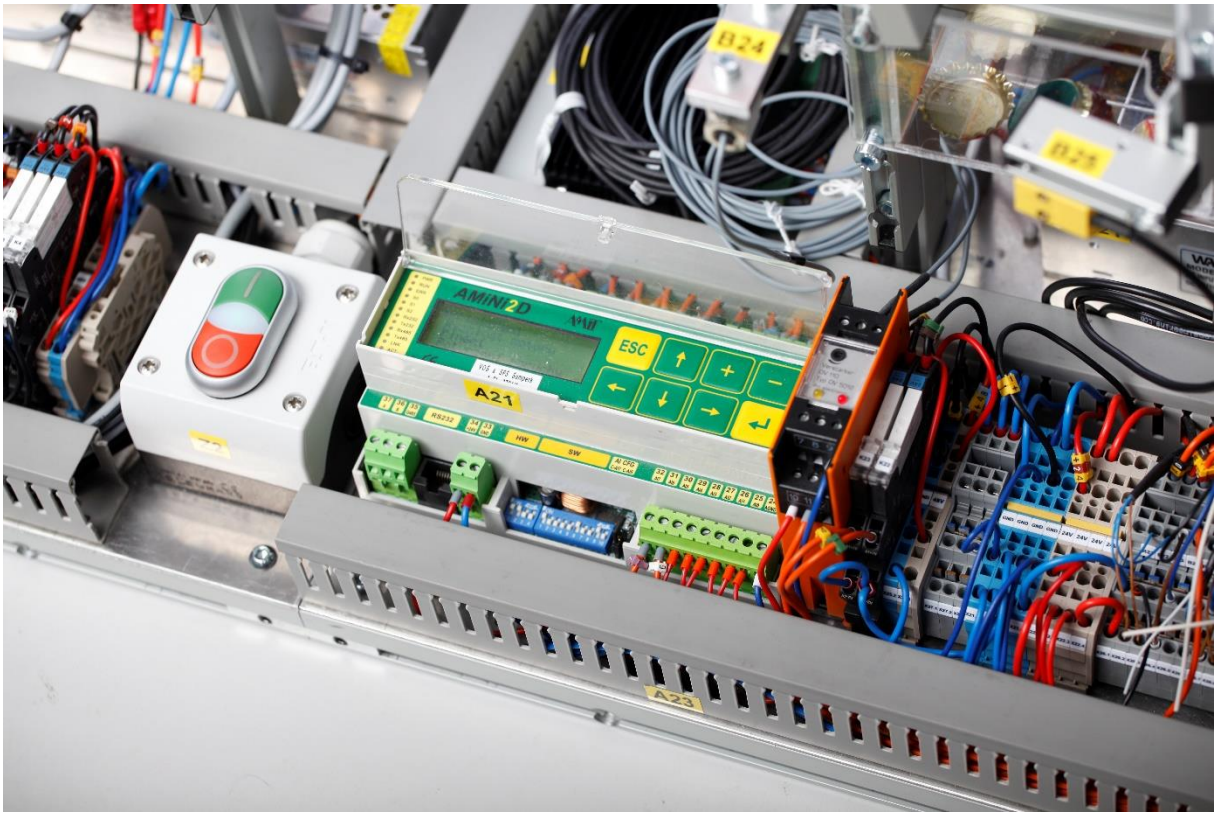
? Nápověda: vytvořte si hlavní obrazovku, do které umístíte „MenuScreen“, odkud se budete pohybovat mezi obrazovkami.

! Pozor: vytvořte na každé obrazovce, na kterou se odkazujete i zpáteční odkaz.

 # Extra: můžete použít přihlášení pro editaci proměnných.

! Pozor: je nutné nastavit u prvků, kdo může editovat a kdo může pouze vidět.

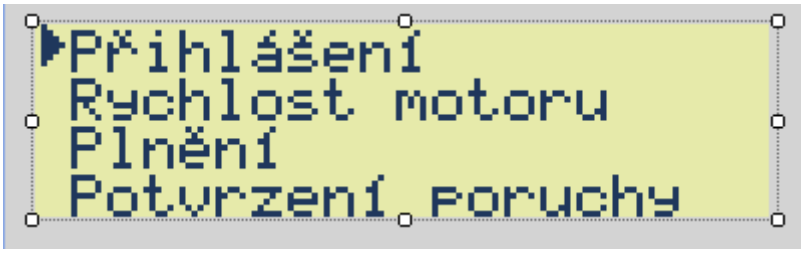
! Pozor: je nutné vytvořit pro „login“ skript, který zajistí výběr v modulu.



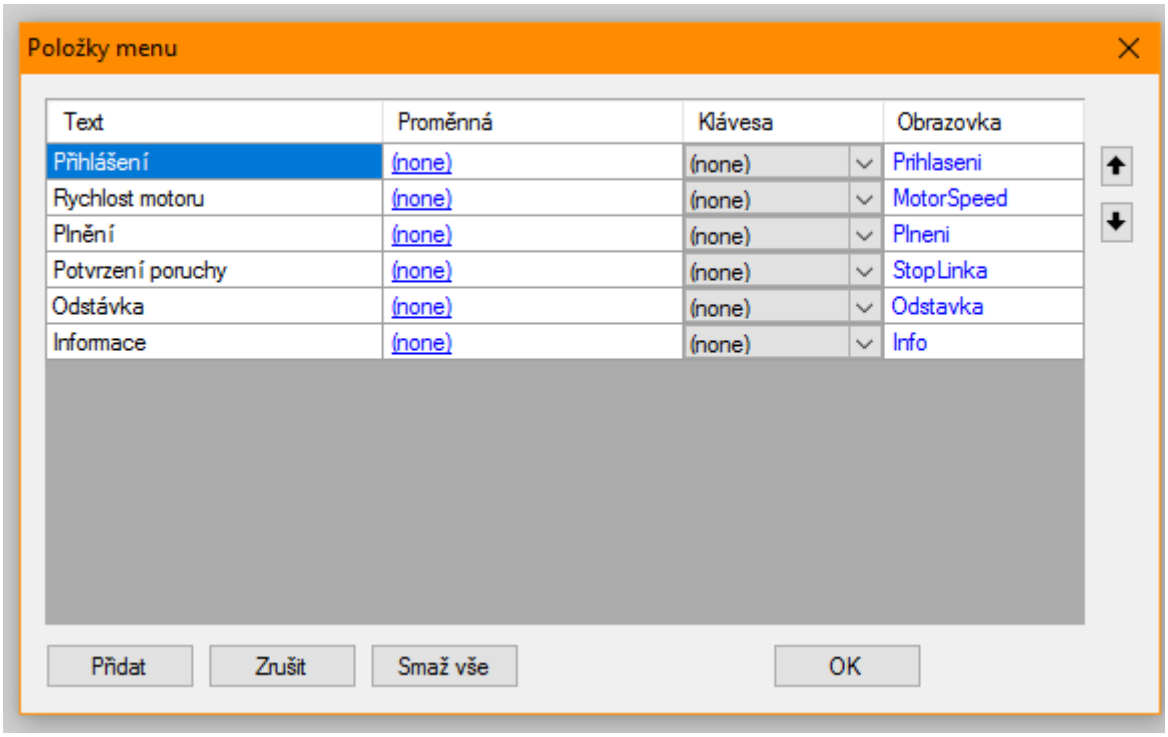
Obrázek 132: PLC s terminálem

9.3.13.3. Program

Pro přehlednost použijeme „MenuScreen“. Jedná se o menu, ve kterém vyberu obrazovku, na kterou se chci odkázat.

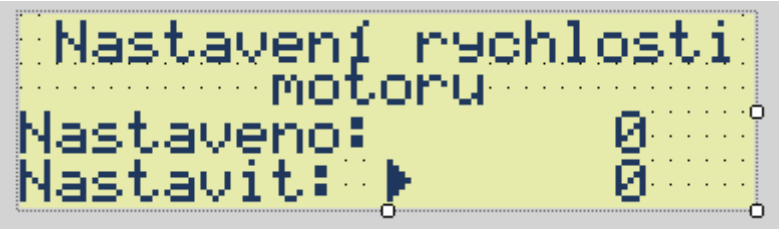


Obrázek 133: Obrazovka s "MenuScreen"



Obrázek 134: Nastavení "MenuScreen"

Poté vytvoříme obrazovky, kterým přiřadíme náležité funkce a vzhled. V každé obrazovce musí být zpáteční odkaz na rozcestník, jinak se nevrátíte do menu.

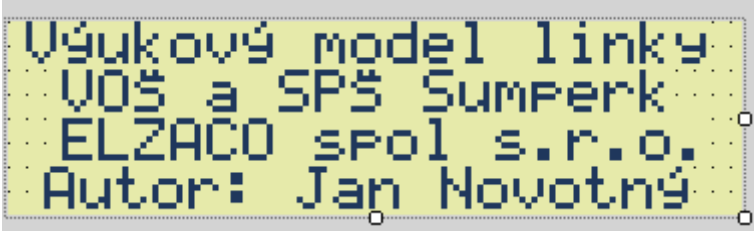


Obrázek 135: Obrazovka nastavení rychlosti motoru

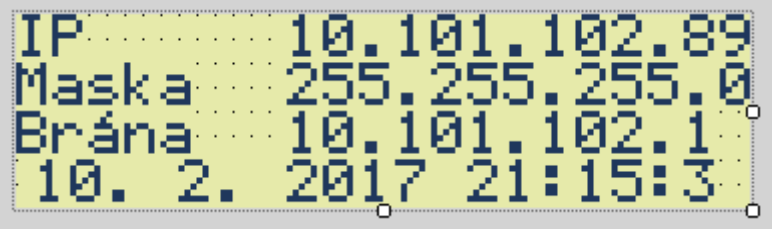
Obrázek 136: Obrazovka nastavení plnění

Obrázek 137: Obrazovka potvrzení poruchy

Obrázek 138: Obrazovka pro konec odstávky



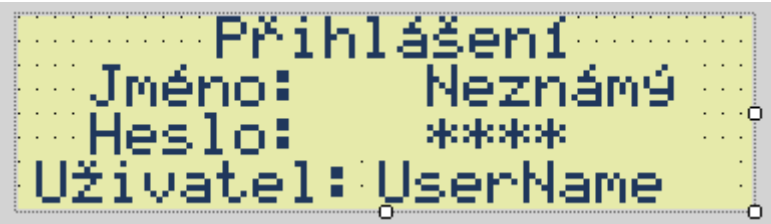
Obrázek 139: Obrazovka s informacemi



Obrázek 140: Obrazovka s informacemi

Extra: Přihlašovací obrazovka

Jako první je zapotřebí vytvořit uživatele (Projekt/Správa uživatelů): viz. strana 123.

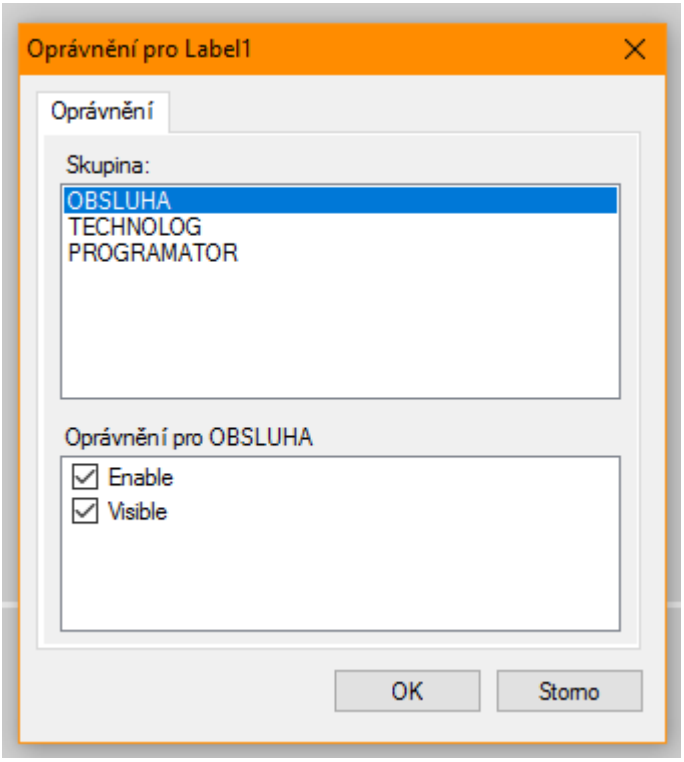


Obrázek 141: Přihlašovací obrazovka

Pro pohyb v modulu „LogIn“ je zapotřebí napsat skript. Skript modulu „LogIn“ bude vypadat následovně:

```
event Login1_OnAccessGrant()
      Login1.Focus();
      Application.SendKey(keys.up);
      Login1.Focus();
      Application.SendKey(keys.down);
end;
```

Přihlášení by bylo k ničemu, když by se pak nevyužívala práva uživatelů. Je tedy zapotřebí, nastavit u každého bloku (najdete názvem „Permission“ a někdy je ještě pod záložkou „Advanced“) a pro každý typ uživatele jde nastavit viditelnost a zda ho sní používat.



Obrázek 142: Nastavení oprávnění

10. Sestava obou linek

10.1. Cíl

Naučit se komunikovat mezi linkami a předávat si mezi linkami data.

10.2. Příklad

Navrhněte program, kde bude řídící systém linky 2 aktivní a bude komunikovat s řídícím systémem na lince 1, do které bude ukládat proměnnou a bude číst proměnnou. Pokud linka 2 bude stát, tak bude stát i linka 1 a budou se předávat informace o provozu.

#Extra: navrhněte programy pro linky tak, aby mezi sebou komunikovali a byly schopné spolupracovat.

10.3. Rozbor zapojení

Linky spolu komunikují přes rozhraní Ethernet. Je využit aktivní prvek, switch, pro zapojení

10.4. Program

Budeme používat pouze jeden řídicí systém jako aktivní, který bude dávat žádosti té druhé. Program bude důležitý pouze tedy v jednom řídícím systému a v druhém řídicím systému bude pouze program pro zobrazování proměnné na displeji a dvě proměnné (do jedné zapisovat a druhou číst). Řídicí systémy mají rozdílná pořadová čísla, a budou rozdílné widy (1001 nebo 2001, atd.), a proto je zapotřebí změnit i číslo widu v aktivním řídícím systému na ty, která obsahuje pasivní řídicí systém. Aby bylo vidět, že linky mezi sebou komunikují můžete naprogramovat i inkrementaci jednotlivých proměnných, nebo na základě tlačítek či senzorů.

Proces INIT proběhne pouze při zapnutí řídicího systému a nastaví se komunikace. Nastavení komunikace má návěstidlo, na které se pak budeme odkazovat při komunikaci.

----------Proces INIT----------

```
:01000 EthNetSeg 0x0A656658, 59, 12345678, 5000, 0x00000000, NONE
```

----------Konec----------

Pro čtení a zápis vytvoříme samostatný proces, kde bude odvozeno od cyklu, jak často se budou data posílat a číst.

----------Proces čtení vstupů a výstupů----------

```
//čtení
If Cti_Stv.0
      //Pokud čtení probíhá, tak komunikaci nevoláme
Else
      EthReqDb :01000, 0x0000, 2, ToLinka2, 1, 1, 2017, Cti_Vlz, Cti_Stv
EndIf

//zápis
If Zap_Stv.0
      //Pokud zápis probíhá, tak komunikaci nevoláme
Else
      EthReqDb :01000, 0x0001, 2, ToLinka1, 1, 1, 2009, Zap_Vlz, Zap_Stv
EndIf
```

----------Konec----------

11. Vizualizace

11.1. ViewDet

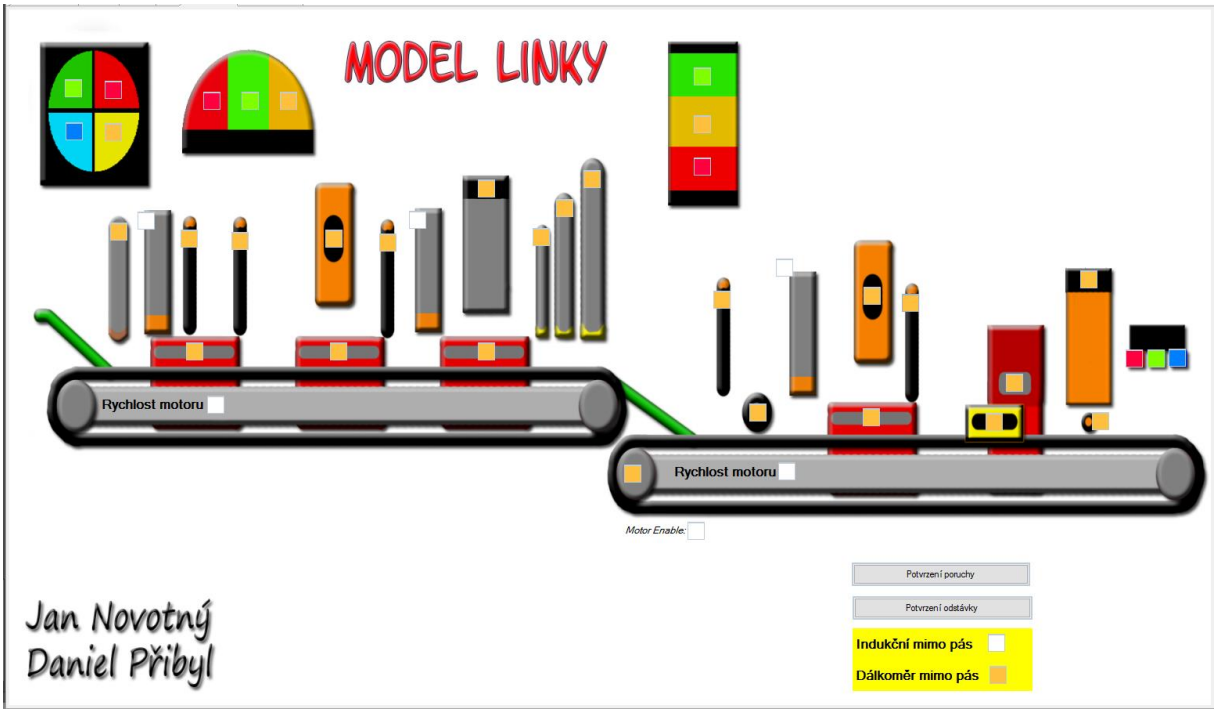
11.1.1. Cíl

Vytvoření vizualizace, pomocí které jde ovládat linka a dohlíží se na chod linky.

11.1.2. Zadání úlohy

Navrhněte vizualizaci ve ViewDetu, kde se bude sledovat průběh linky (aktivnost senzorů) a bude se moci zasahovat do řízení linek (rychlost motoru, potvrzení poruchy, ...).

! Pozor: pokud nastavíte obnovovací čas na 100ms, tak je to zbytečné a řídicí systém to zbytečně zatěžuje a může to vést ke kolizi.

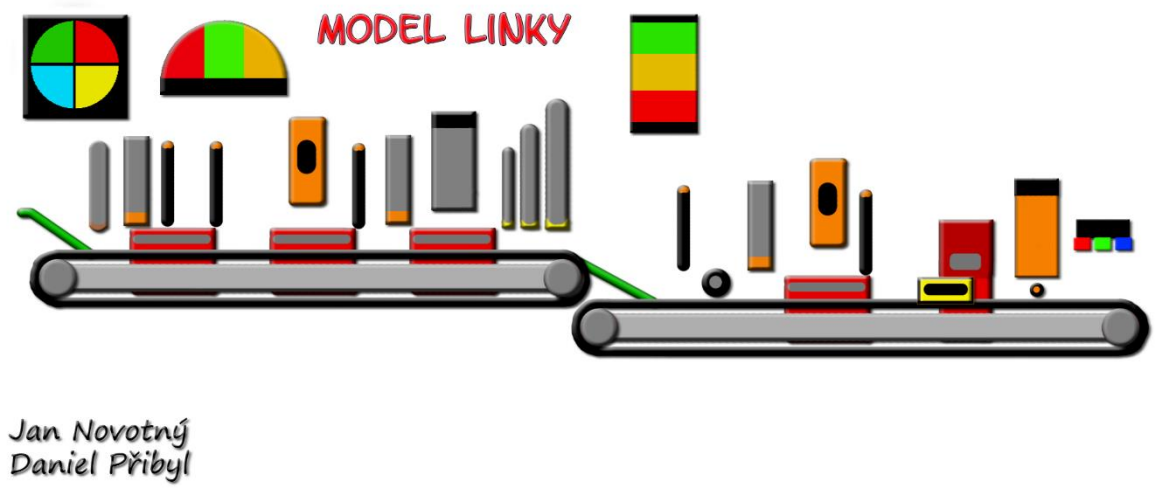


Obrázek 143: Předloha, jak by mohla vypadat závěrečná vizualizace

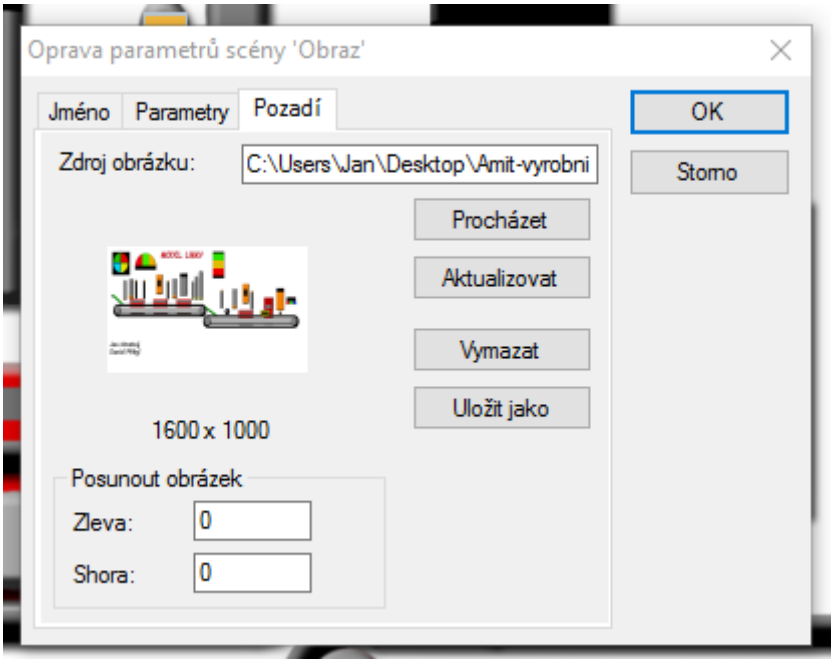
11.1.3. Vytvoření kompozice

Základem vytvoření vizualizace je kompozice neboli pozadí. Nejedná se o funkční, ale pouze o vizuální část a program bude fungovat i bez toho. Obrázek se může vytvořit v jakémkoli grafickém editoru.

! Pozor: velikost obrázku je důležitá, jelikož se s ní pak již nedá nadále pracovat.



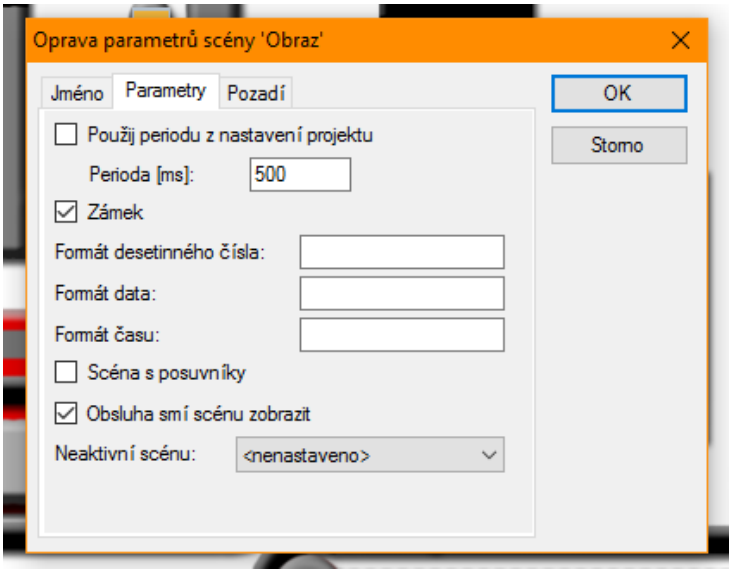
Obrázek 144: Kompozice vizualizace



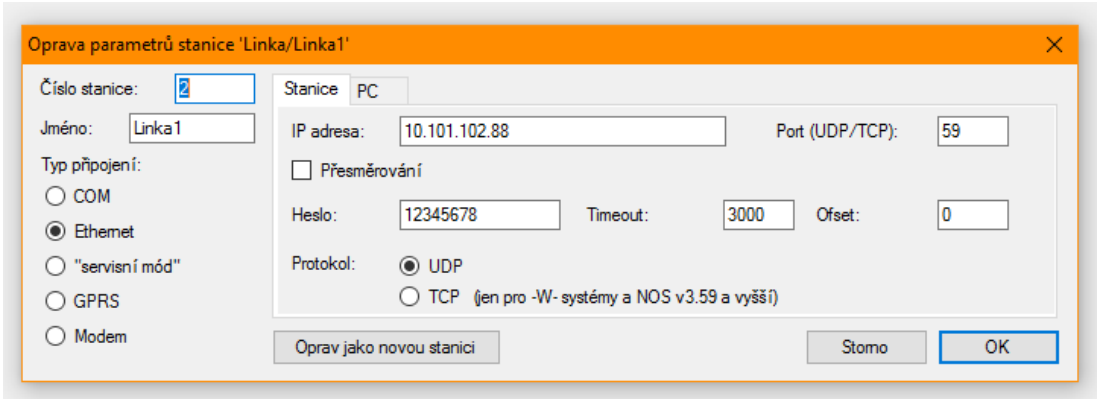
Obrázek 145: Parametrizace pozadí (vytvoření kompozice)

11.1.4. Nastavení komunikace

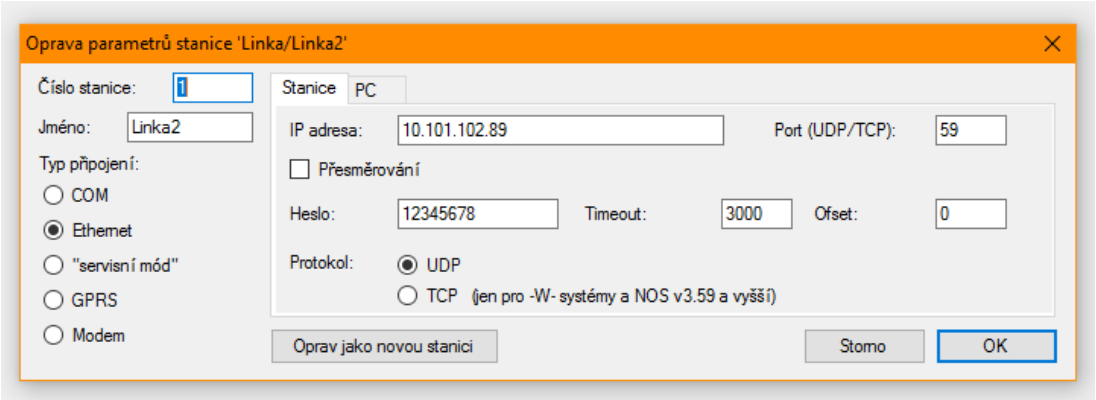
Důležité je nastavit správně komunikaci a periodu. Každé PLC bude mít svoje nastavení. Periodu je důležité nastavit na začátku. Pokud bude nastavena perioda na 100ms, tak to bude zbytečně zatěžovat PLC a komunikace nemusí fungovat podle představ. Pokud by se jednalo o čtení teploty, jde o pomalý proces, tak by stačilo například číst teplotu z PLC jednou za 5s. My budeme číst senzory, kde jsou děje rychlejší, takže periodu nastavíme na 500ms. Pokud budeme používat více scén, je zapotřebí pro každou scénu nastavit zvláště periodu.



Obrázek 146: Nastavení periody obrazovky



Obrázek 147: Nastavení komunikace pro linku 1

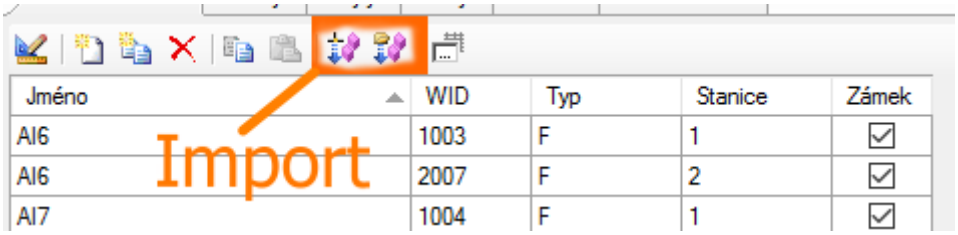


Obrázek 148: Nastavení komunikace pro linku 2

11.1.5. Import dat

Abychom mohli dále pracovat s daty a zobrazovat je, musíme naimportovat proměnné ze stanice či z programu.

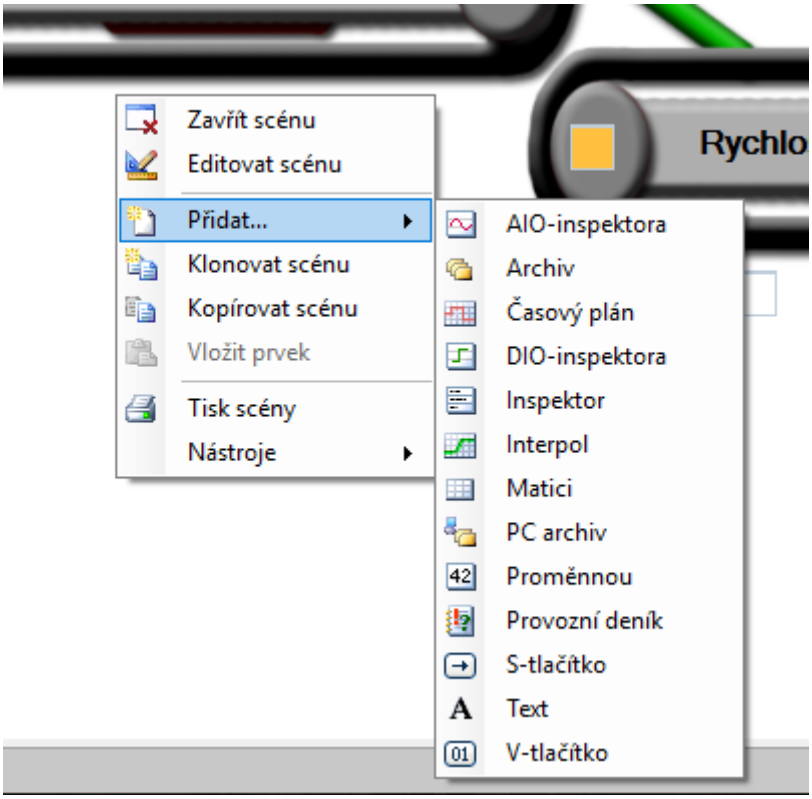
? Nápověda: pokud budeme importovat proměnné z programu a budou shodné proměnné, tak nedojde k jejich duplikaci ale k aktualizaci a proměnné z jiných stanic zůstanou. Aliasy by se měly nahrát s proměnnými.



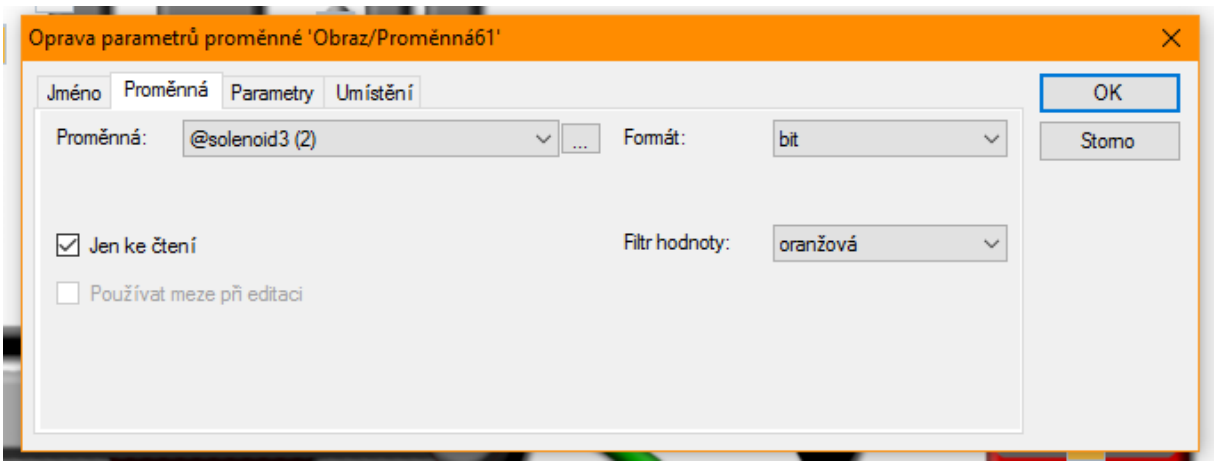
Obrázek 149: Import proměnných z projektu v programu

11.1.6. Vkládání objektů a jejich parametrizace

Funkční věcí jsou pak objekty. Vložíme je pravým kliknutím do scény, přidat a vybereme objekt, co chceme přidat. Pro základní věci nám vystačí objekt proměnná, tlačítko, text.



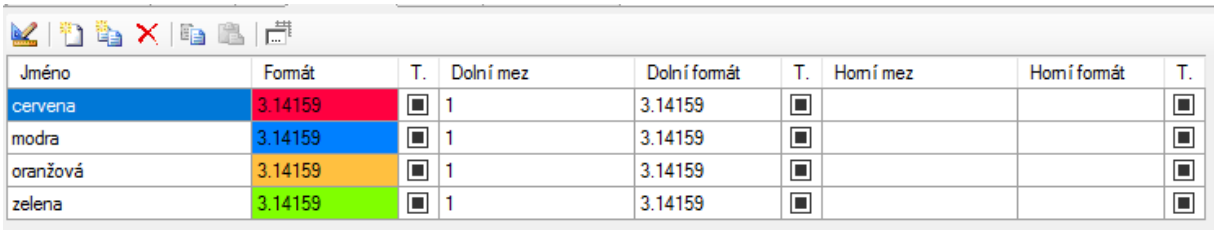
Obrázek 150: Vkládání jednotlivých objektů do kompozice



Obrázek 151: Ukázka parametrizace jednoho objektu

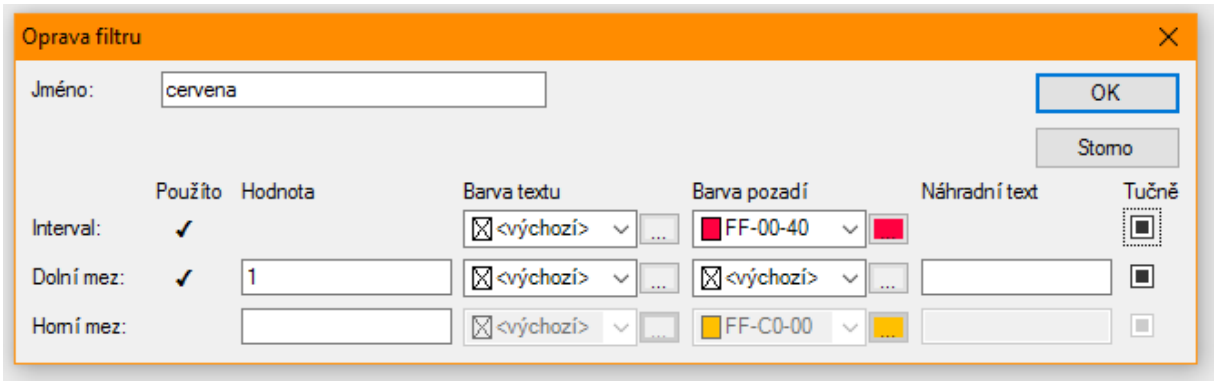
11.1.7. Filtry a styly

Filtry se dají použít v proměnných, jakožto změnění barev při změně hodnoty (log.0 bude bílá a pro log.1 bude pozadí barevné), nebo při kritických či neobvyklých hodnotách (pokud je teplota vyšší, než by měla být, tak pole bude červené, jinak bude zelené). Styly se používají na text.



Jméno	Formát	T.	Dolní mez	Dolní formát	T.	Horní mez	Horní formát	T.
cervena	3.14159	☐	1	3.14159	☐			☐
modra	3.14159	☐	1	3.14159	☐			☐
oranžová	3.14159	☐	1	3.14159	☐			☐
zelena	3.14159	☐	1	3.14159	☐			☐

Obrázek 152: Seznam filtrů



Obrázek 153: Parametrizace jednoho filtru

12. Závěr

Podařilo se nám navrhnout a vytvořit funkční model automatizační linky. Linka se využívá při výuce programování řídicích systémů.

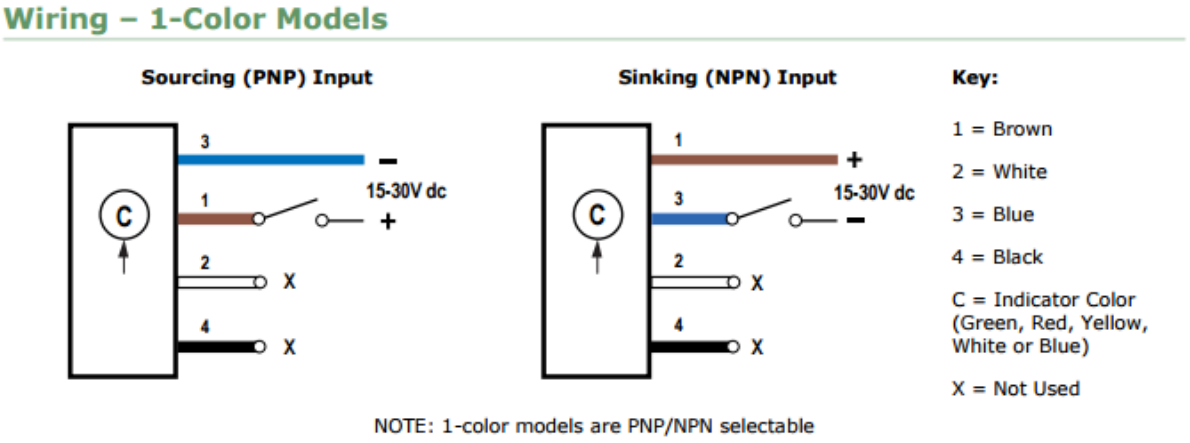
Linka je řízena pomocí řídicího systému AMiNi2D.

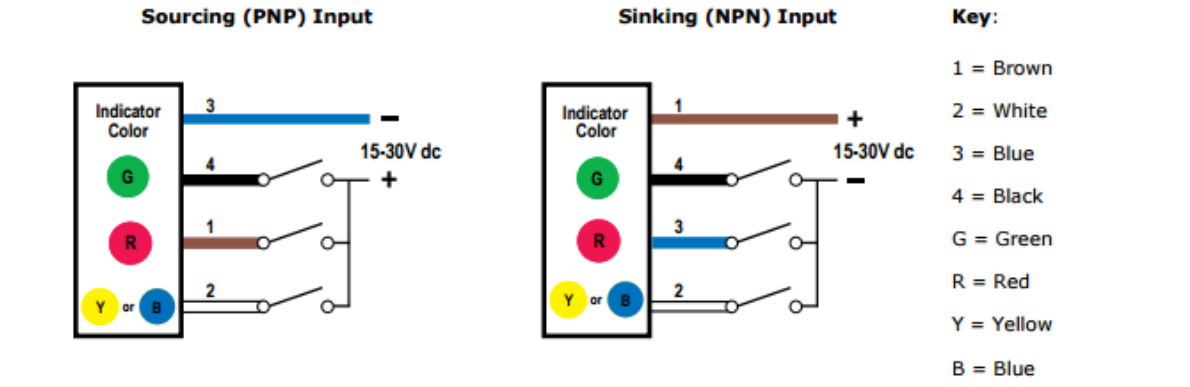
Automatizovaná linka se skládá ze dvou částí, které lze podle potřeby spojit. Úkolem linky je třídit kovové uzávěry pivních lahví. Uzávěry jsou vyhodnocovány jak digitálními, digitálními programovatelnými, ale také analogovými senzory. Vyhodnocuje se natočení, barva, přítomnost těsnění a kapacita uzávěrů. Pokročilejší funkcí linky je plnění uzávěrů plastovými kuličkami o průměru 6 milimetrů a kontrola tohoto procesu. Pro vizualizaci je na jedné lince umístěn osobní počítač, přes který se lze nahrávat program. Celý projekt je propojen pomocí Ethernetu do sítě LAN.

13. Zdroje

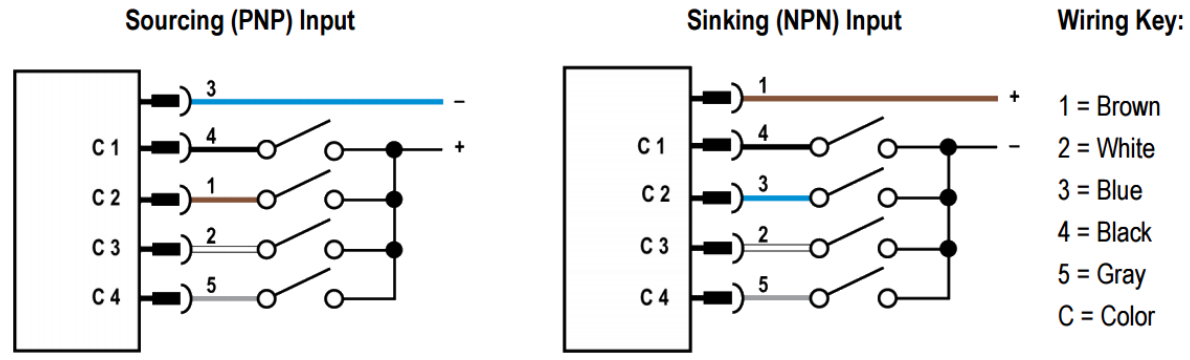
[1] [online]. [cit. 2017-03-24]. Dostupné z: http://www.amit.cz/docs/cz/obsolete/amini2dm.pdf;
[2] [online]. [cit. 2017-03-24]. Dostupné z: http://www.raveo.cz/sites/default/files/download/2014/11/16-hs2002231.pdf;
[3] [online]. [cit. 2017-03-24]. Dostupné z: http://www.cncshop.cz/m542-driver-pro-krokove-motory-50v-4-2a;
[4] [online]. [cit. 2017-03-24]. Dostupné z: http://stolni-pocitace.heureka.cz/rikomagic-mk36s/specifikace/#http://stolni-pocitace.heureka.cz/rikomagic-mk36s/specifikace/;
[5] [online]. [cit. 2017-03-24]. Dostupné z: https://www.mall.cz/lcd-monitory/benq-gw2270h-9hle6latbe?utm_source=heureka.cz&utm_medium=cse&utm_campaign=EG&utm_content=lcd-monitory&utm_term=821763;
[6] [online]. [cit. 2017-03-24]. Dostupné z: http://www.hotair.cz/detail/prumyslove-zdroje/modulove-pro-vestavbu/prumyslovy-zdroj-wxd-240w-24v-10a-240w.html;
[7] [online]. [cit. 2017-03-24]. Dostupné z: [7] http://www.tme.eu/cz/details/trz-24vdc-1co/elektromagneticka-rele-sady/weidmuller/1122880000/;
[8] [online]. [cit. 2017-03-24]. Dostupné z: http://www.ebay.com/itm/100W-120W-150W-DC-DC-Boost-Converter-10V-32V-to-12V-60V-Step-Up-Power-Supply-/191949418653?var=&hash=item2cb113b09d:m:mVCt85esokpA6emv_ngeBmA;
[9] [online]. [cit. 2017-03-24]. Dostupné z: http://amit.cz/cz/support/appnotes.htm;
[10] [online]. [cit. 2017-03-24]. Dostupné z: http://www.ifm.com/ifmcz/web/home.htm;
[11] [online]. [cit. 2017-03-24]. Dostupné z: http://www.turck.cz/cs/;
[12] [online]. [cit. 2017-03-24]. Dostupné z: http://www.balluff.com/local/cz/home/;
[13] [online]. [cit. 2017-03-24]. Dostupné z: http://www.aluteckk.cz/;

14. Obrázky

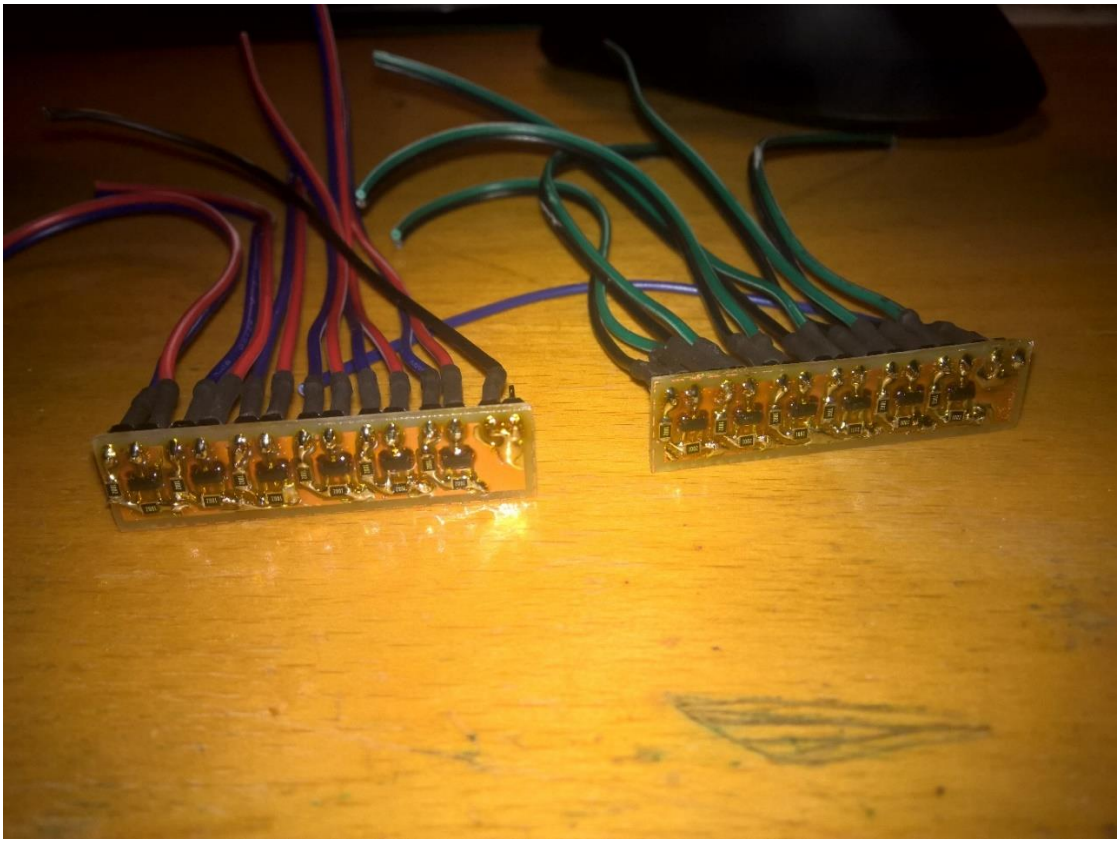




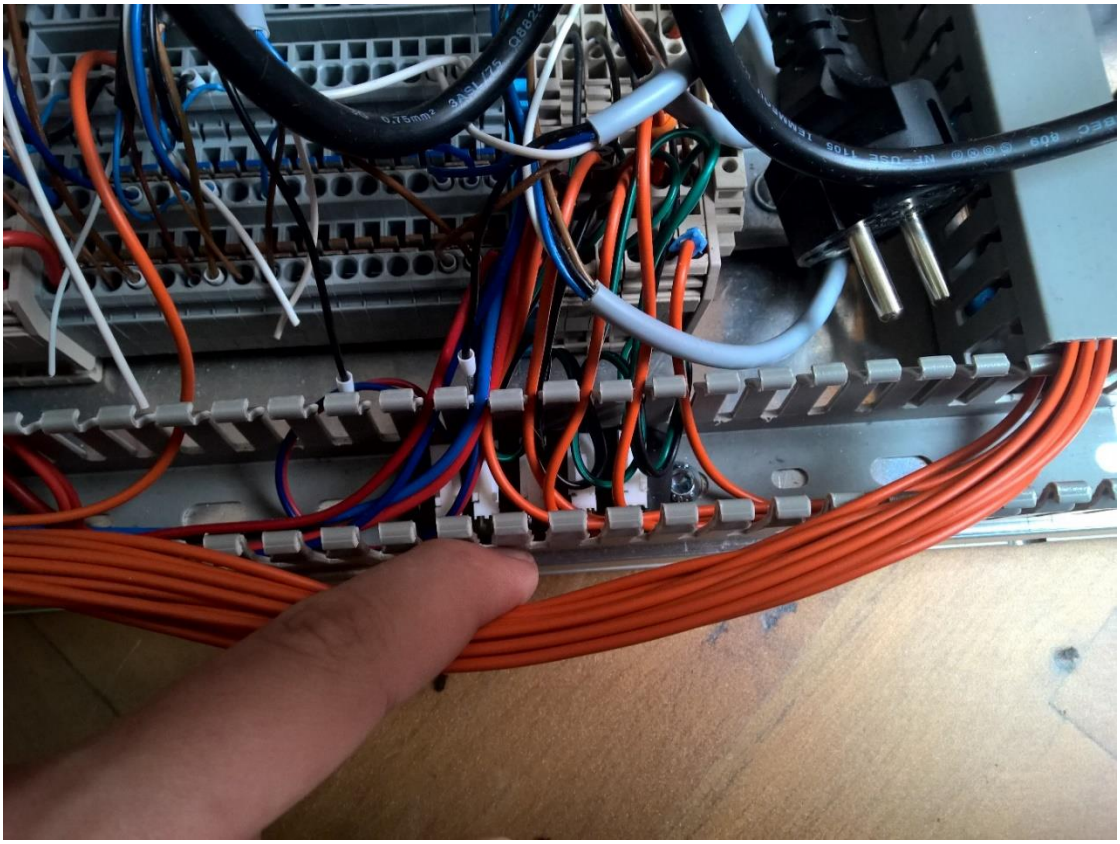
Obrázek 154: Zapojení tříbarevného světla



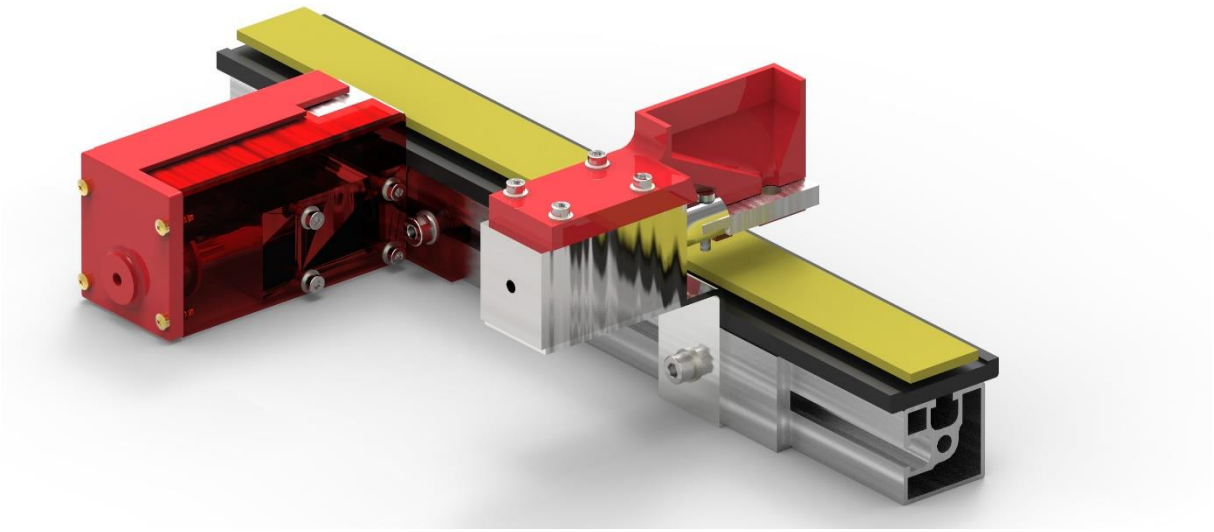
Obrázek 155: Zapojení čtyřbarevného indikátoru



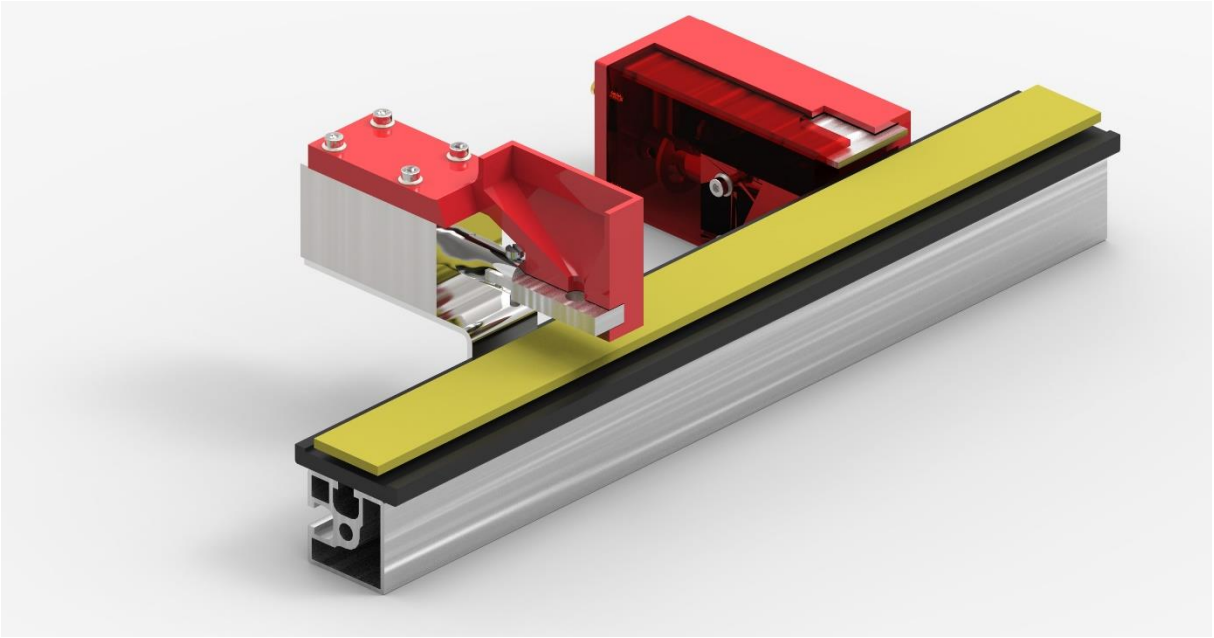
Obrázek 156: AI konventory



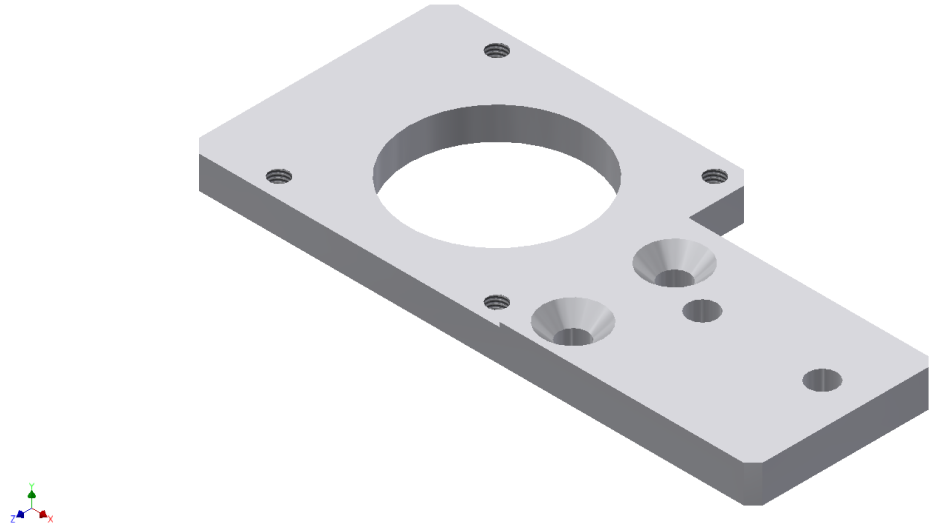
Obrázek 157: AO amplifier v korýtku



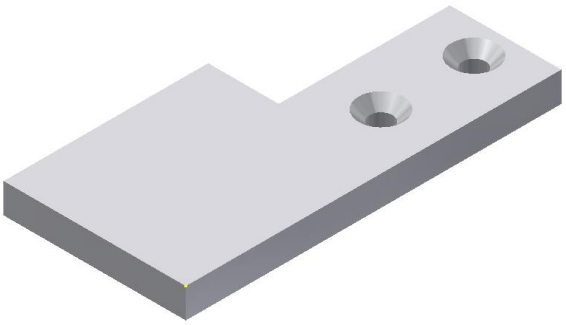
Obrázek 158: Renderovaný obrázek z cadu linky s návrhem plničky a třídičky



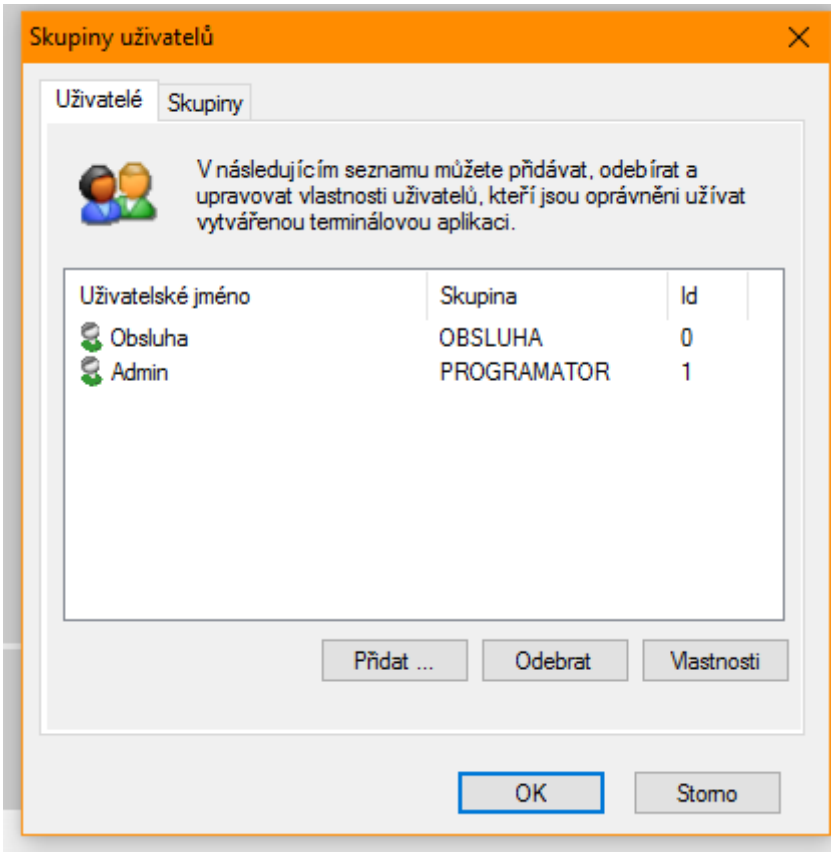
Obrázek 159: Renderovaný obrázek z cadu linky s návrhem plničky a třídičky



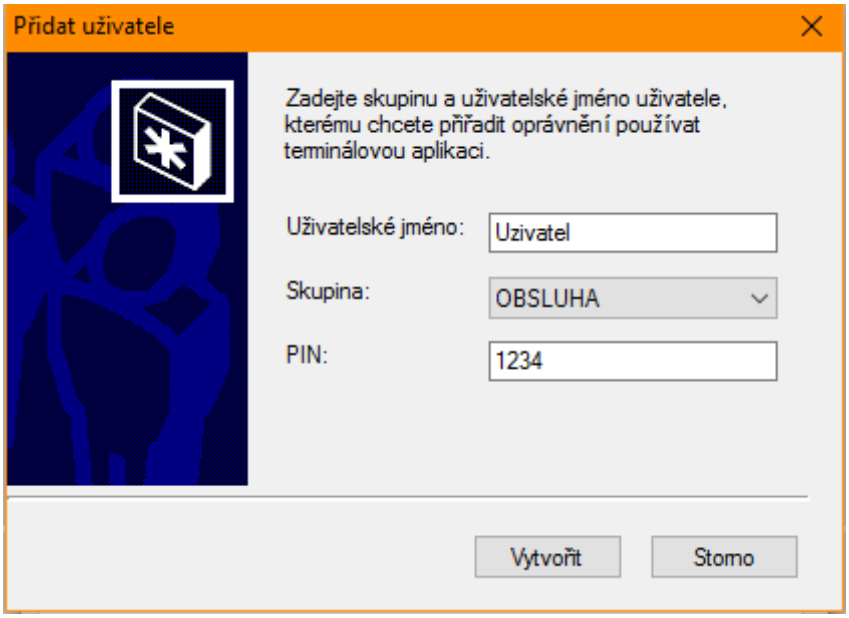
Obrázek 160: Návrh příruby pro uchycení motoru na dopravník



Obrázek 161: Návrh pro kryt řemenice u motoru na dopravníku

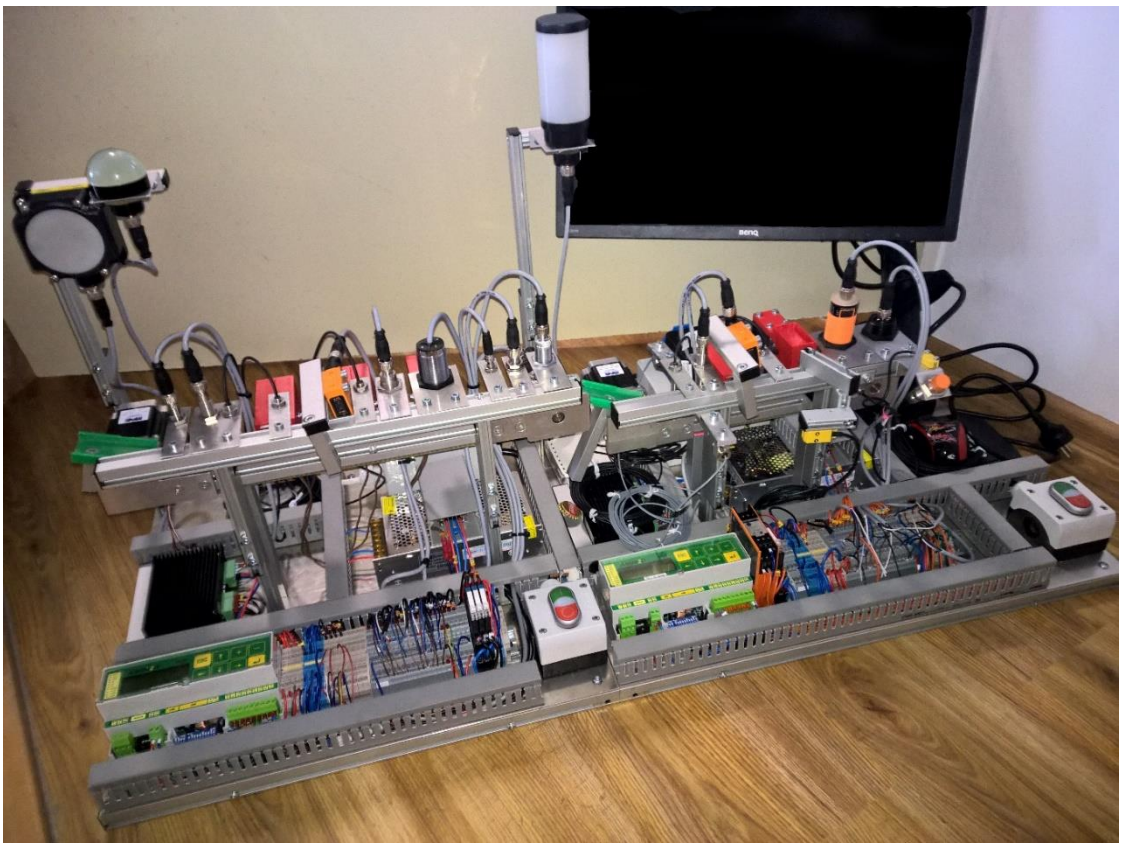


Obrázek 162: Správa uživatelů

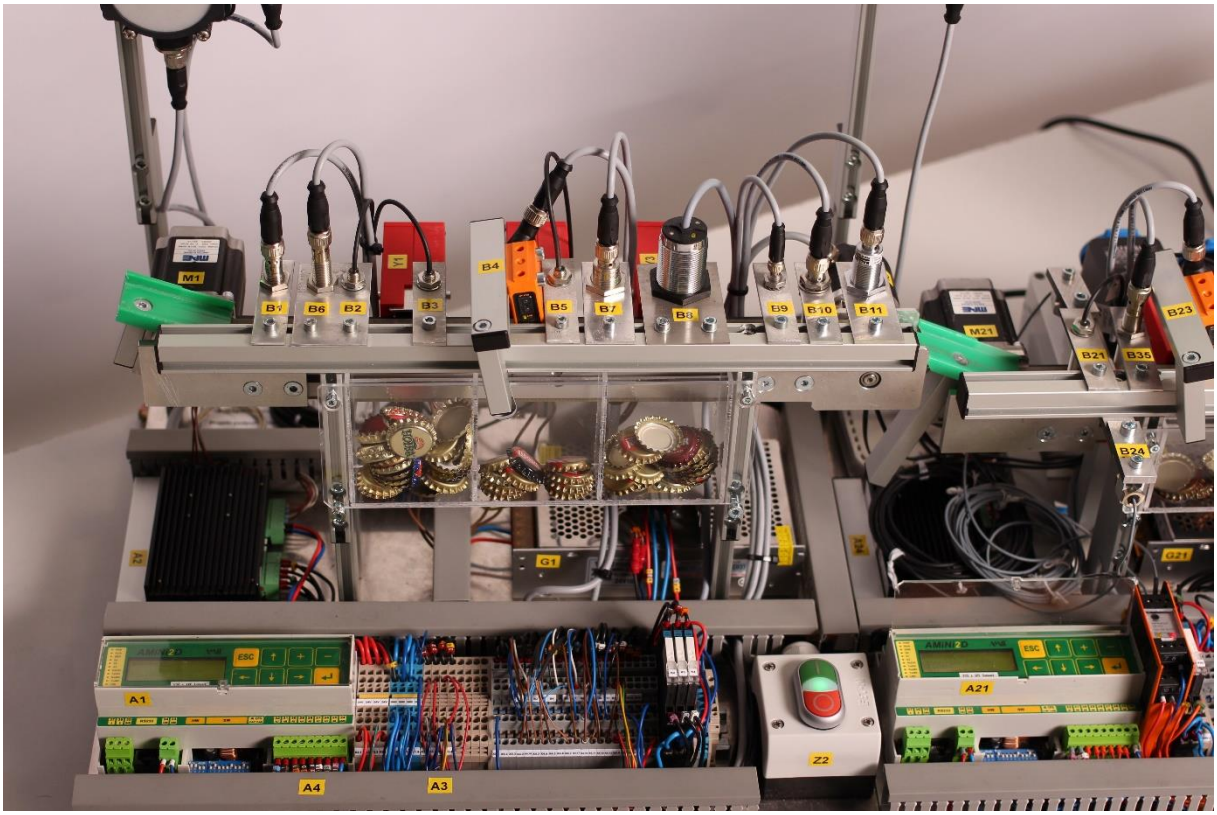


Obrázek 163: Přidání nového uživatele

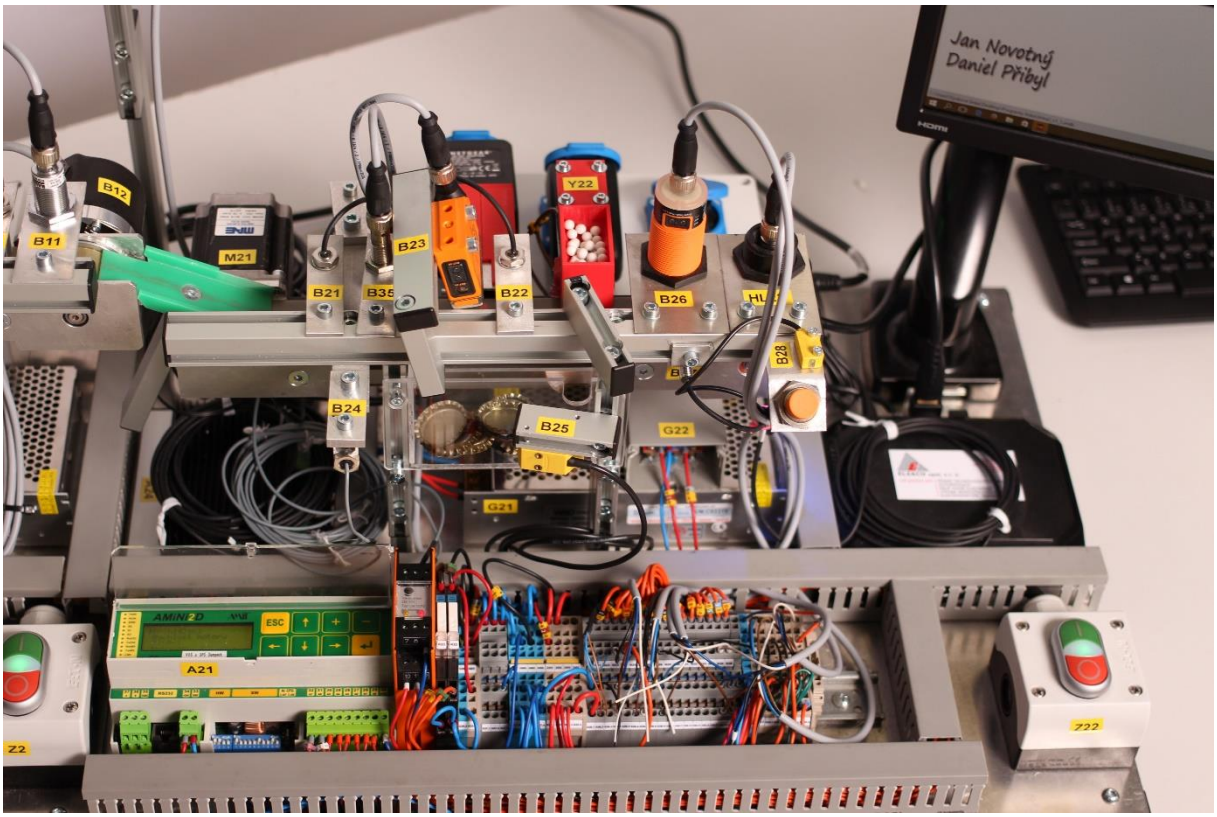
Obrázek 164: Foto autorů z výroby



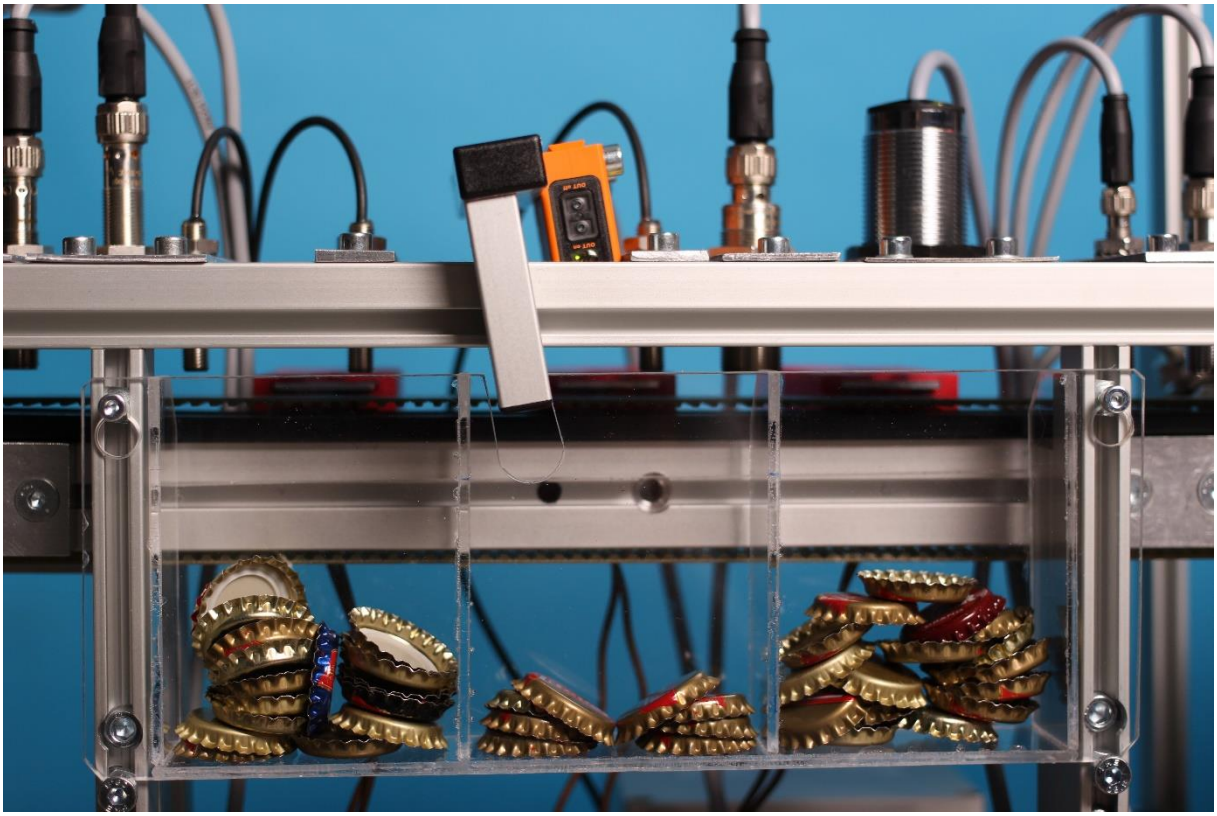
Obrázek 165: Celá linka



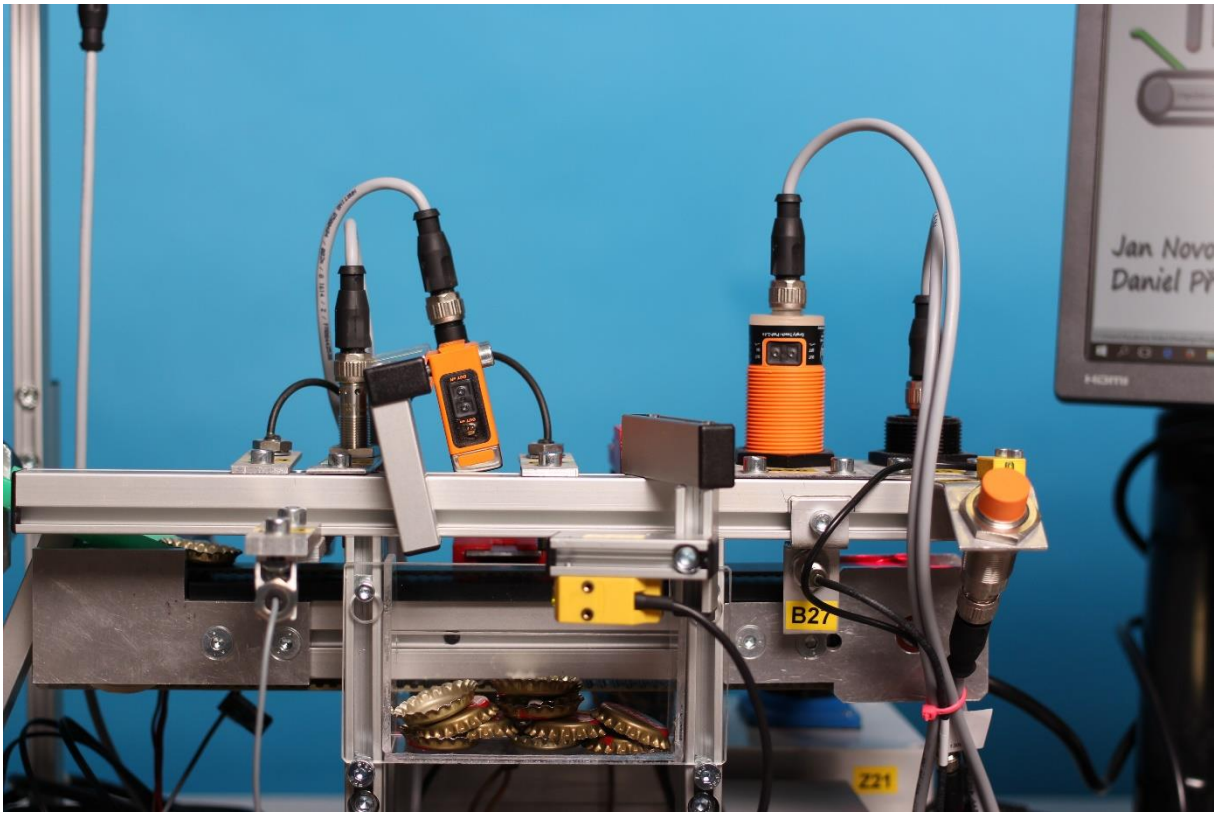
Obrázek 166: Linka 1



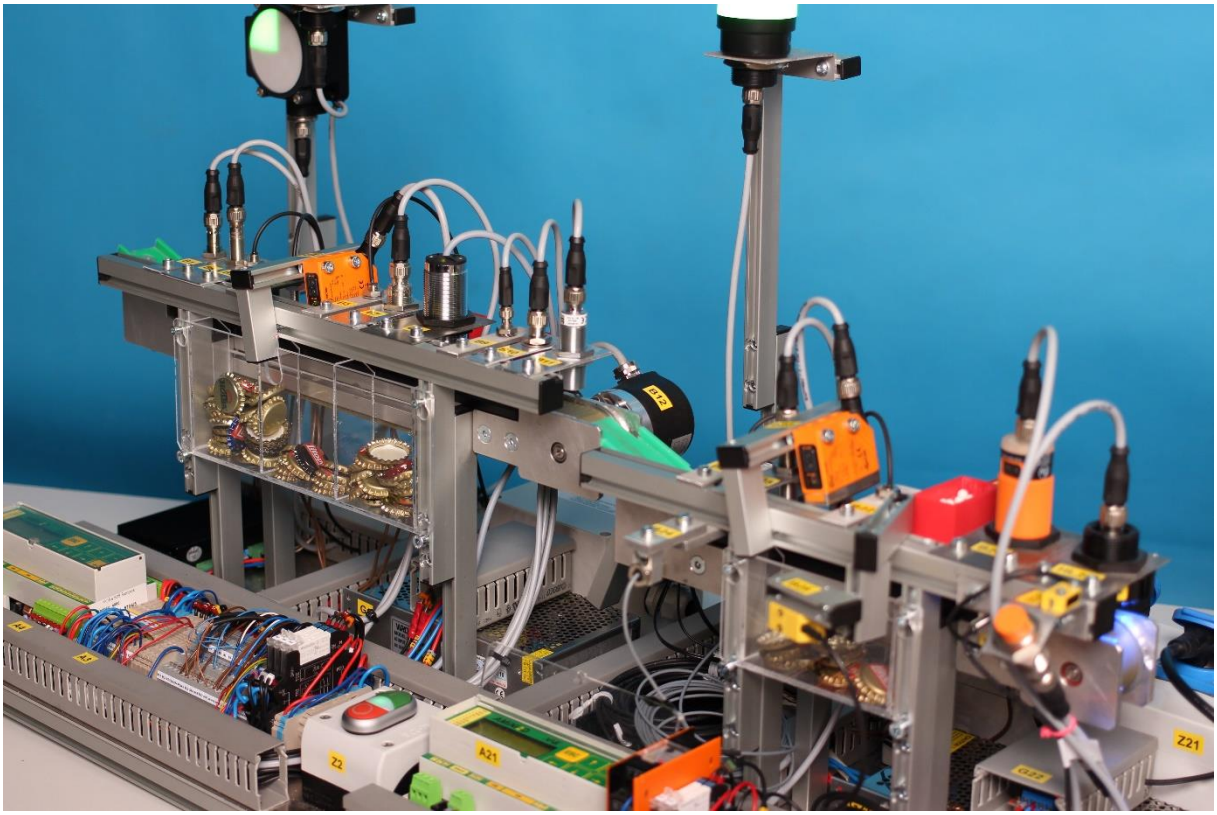
Obrázek 167: Linka 2



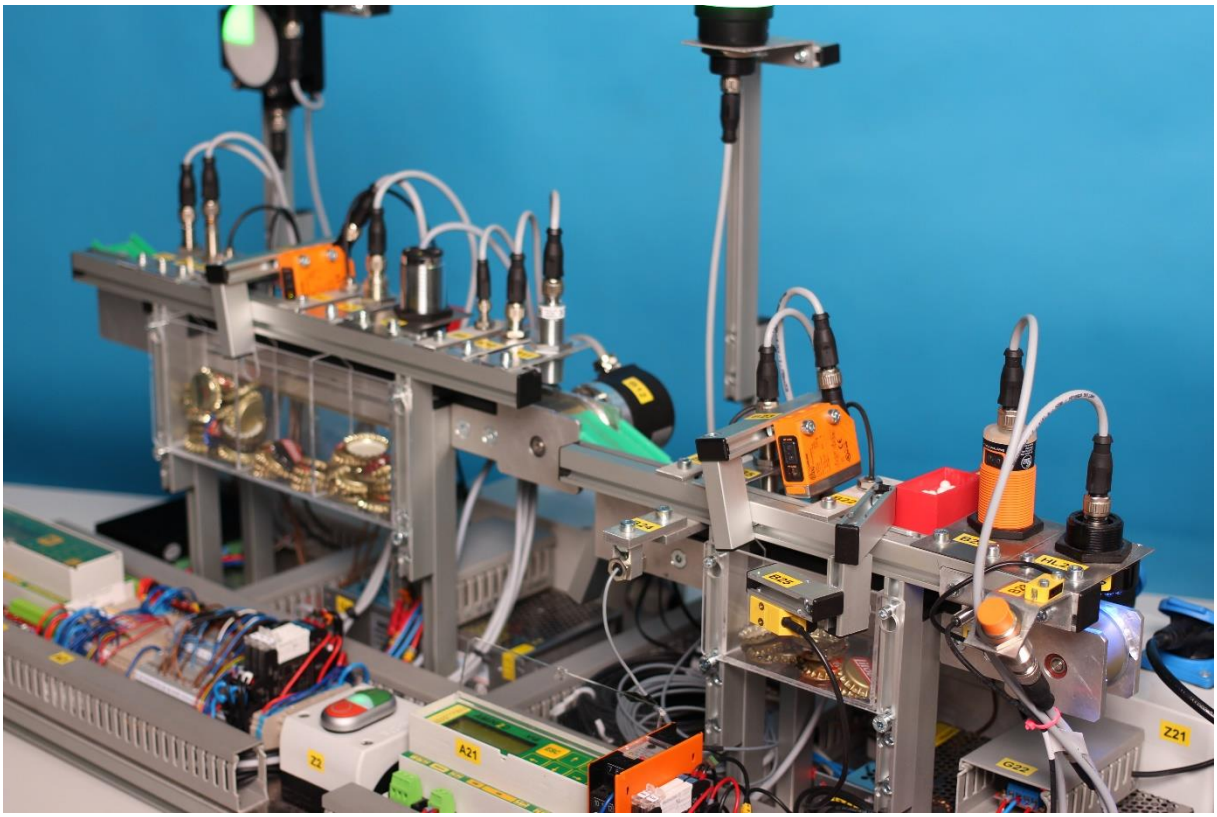
Obrázek 168: Zásobník linky 1



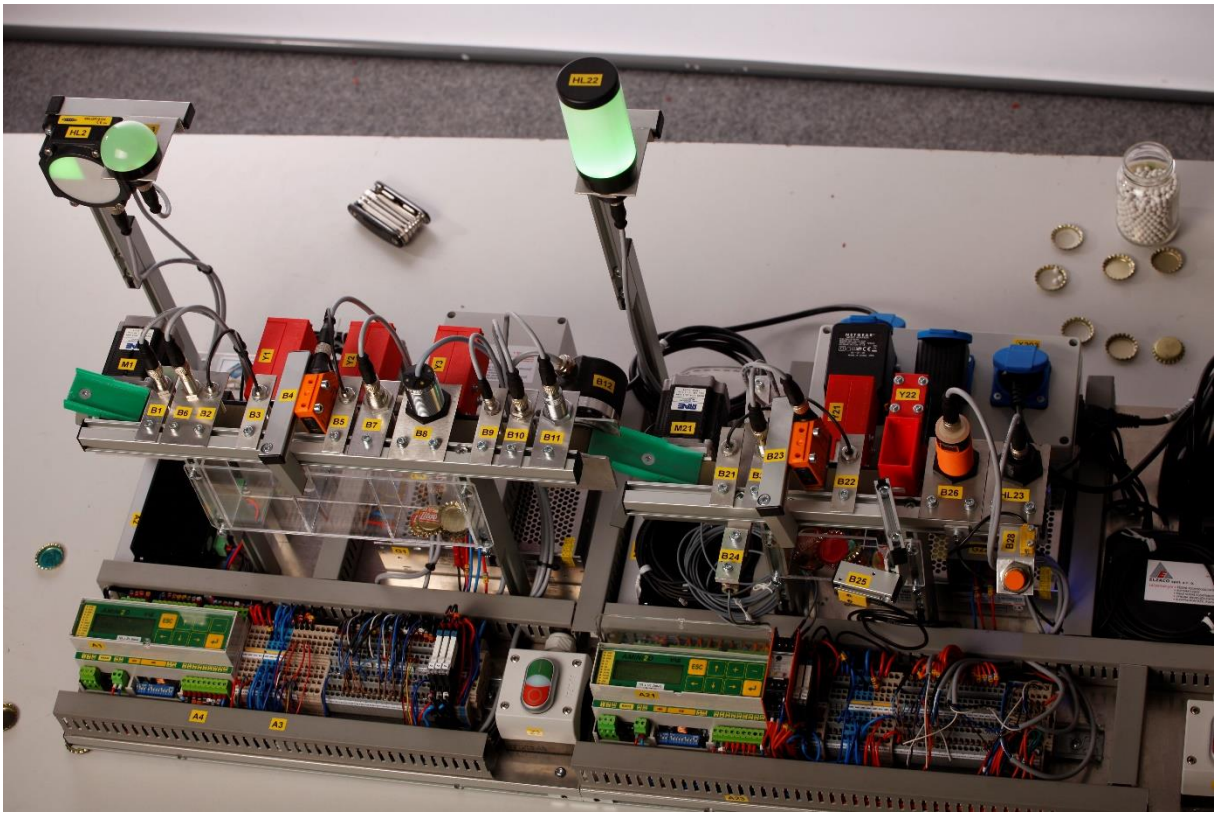
Obrázek 169: Zásobník linky 2



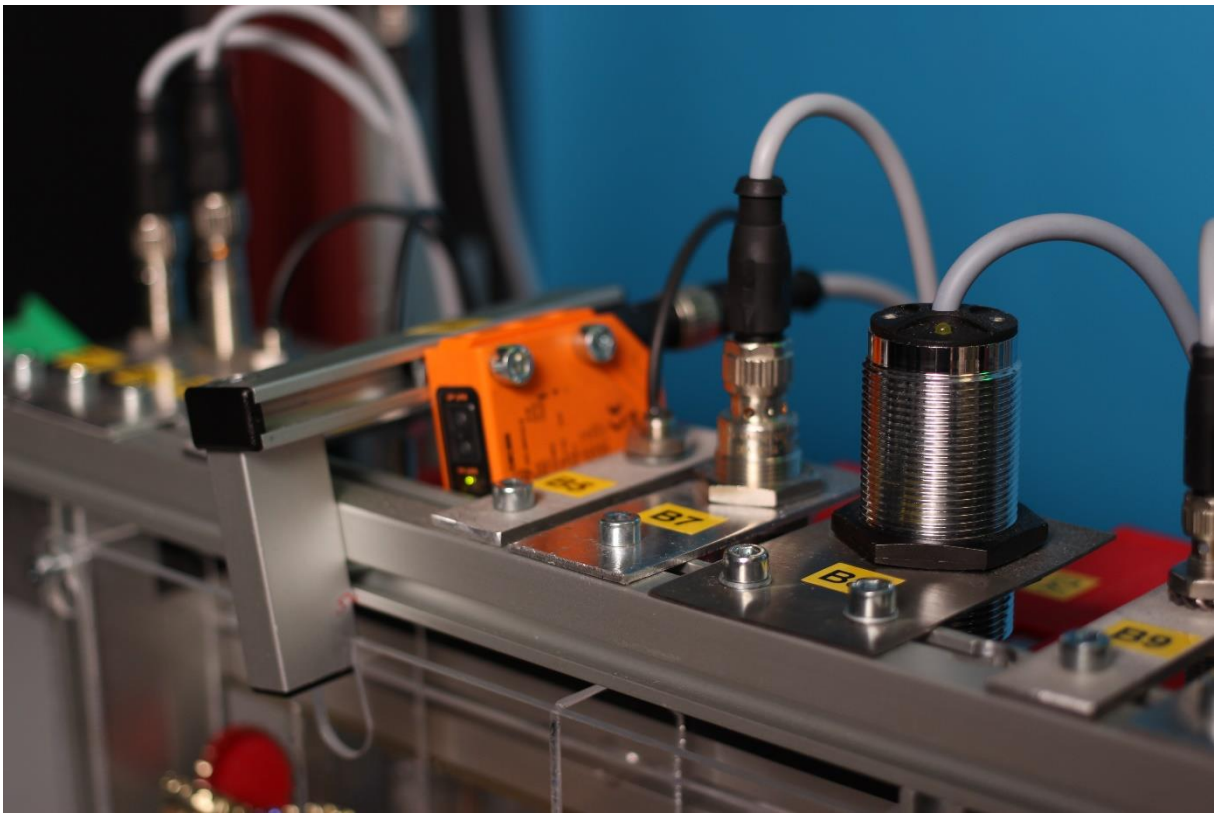
Obrázek 170: Zaostřeno na linku 1



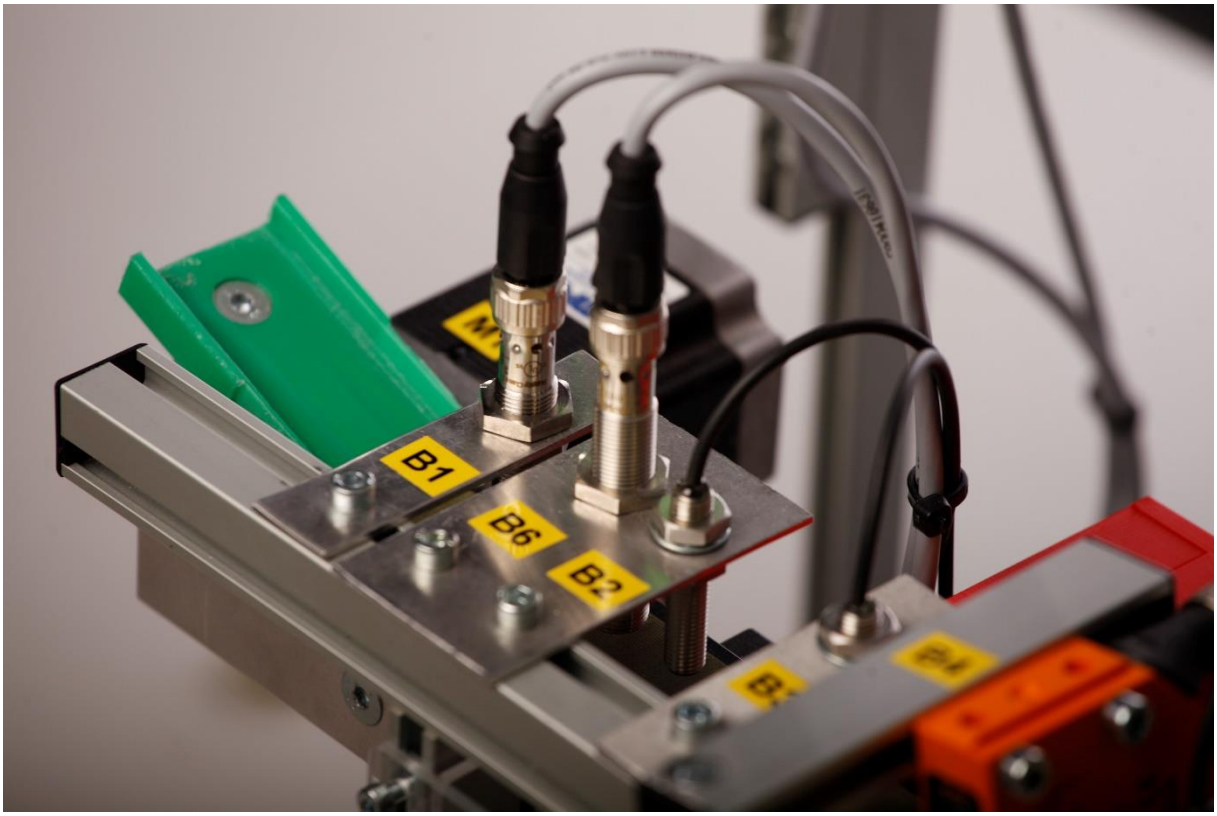
Obrázek 171Zaostřeno na linku 2



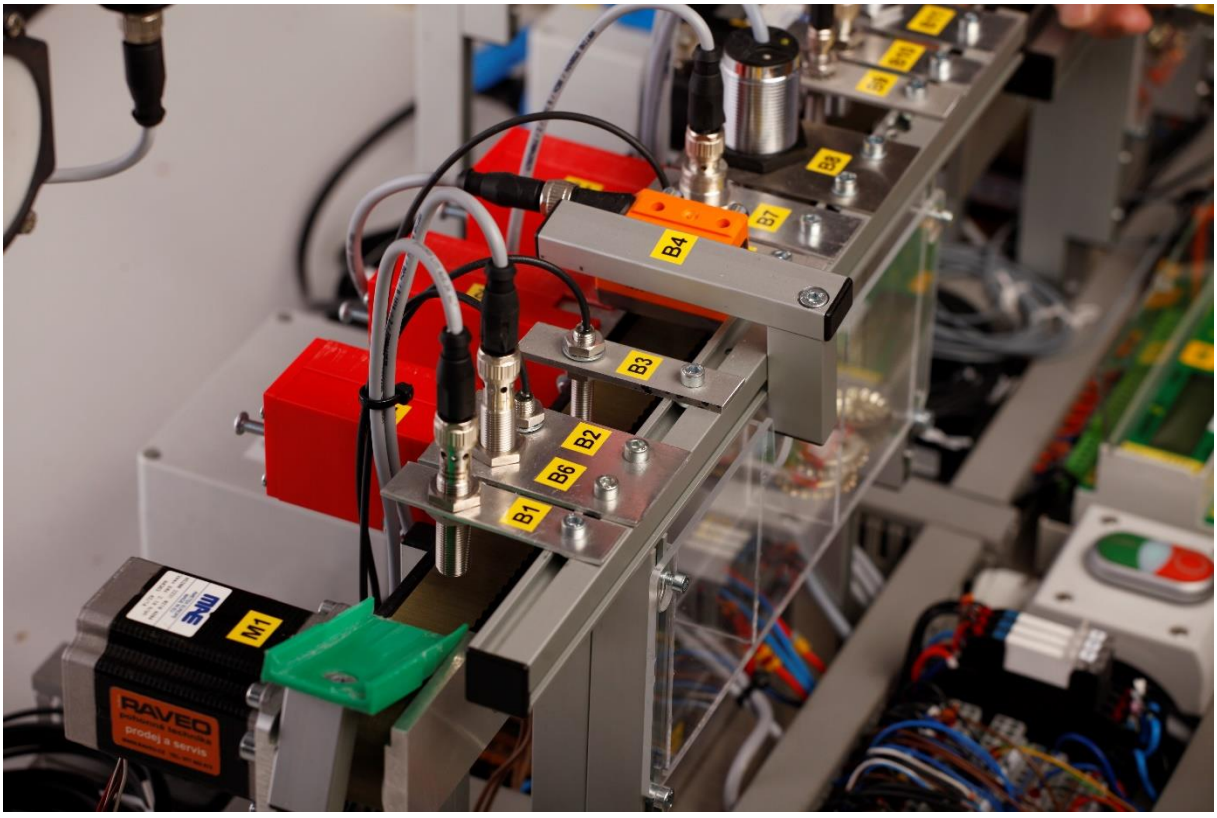
Obrázek 172: Pohled na linky ze shora



Obrázek 173: Senzory

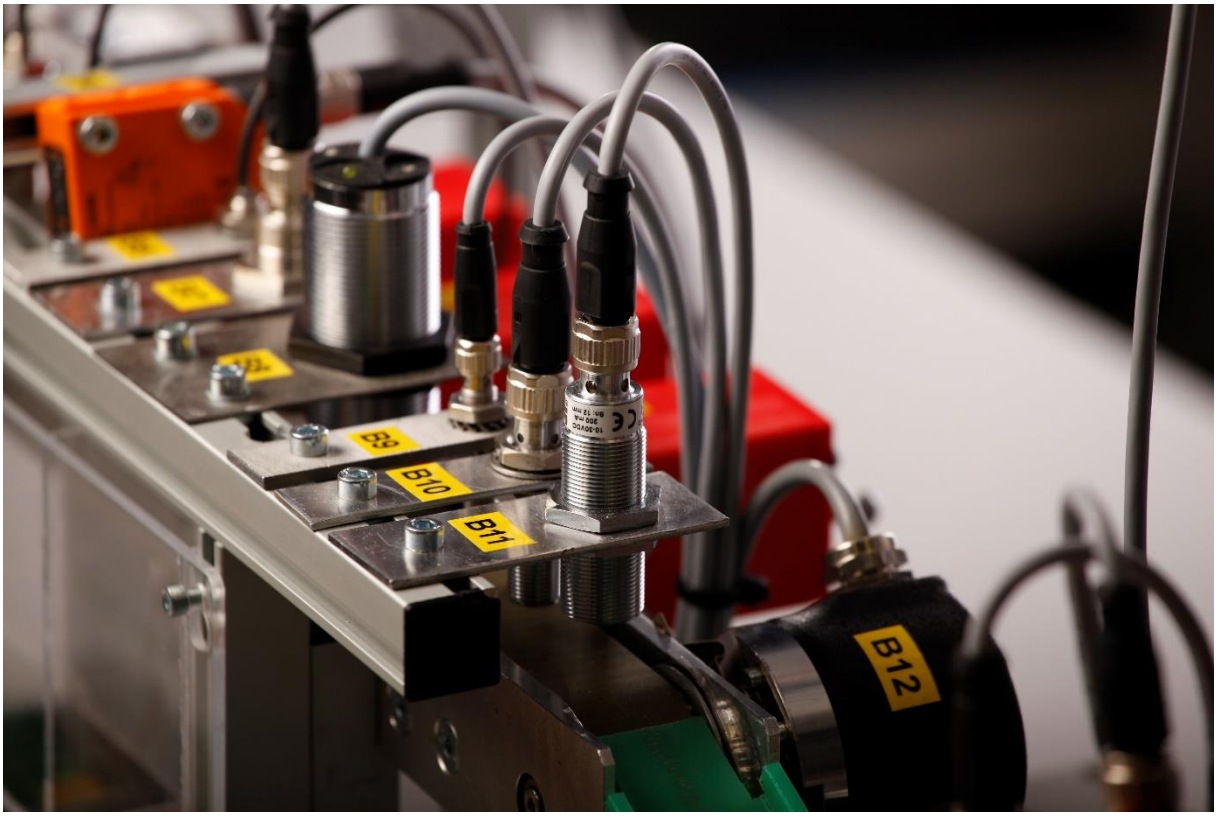


Obrázek 174: Skluz na lince 1 s prvními senzory

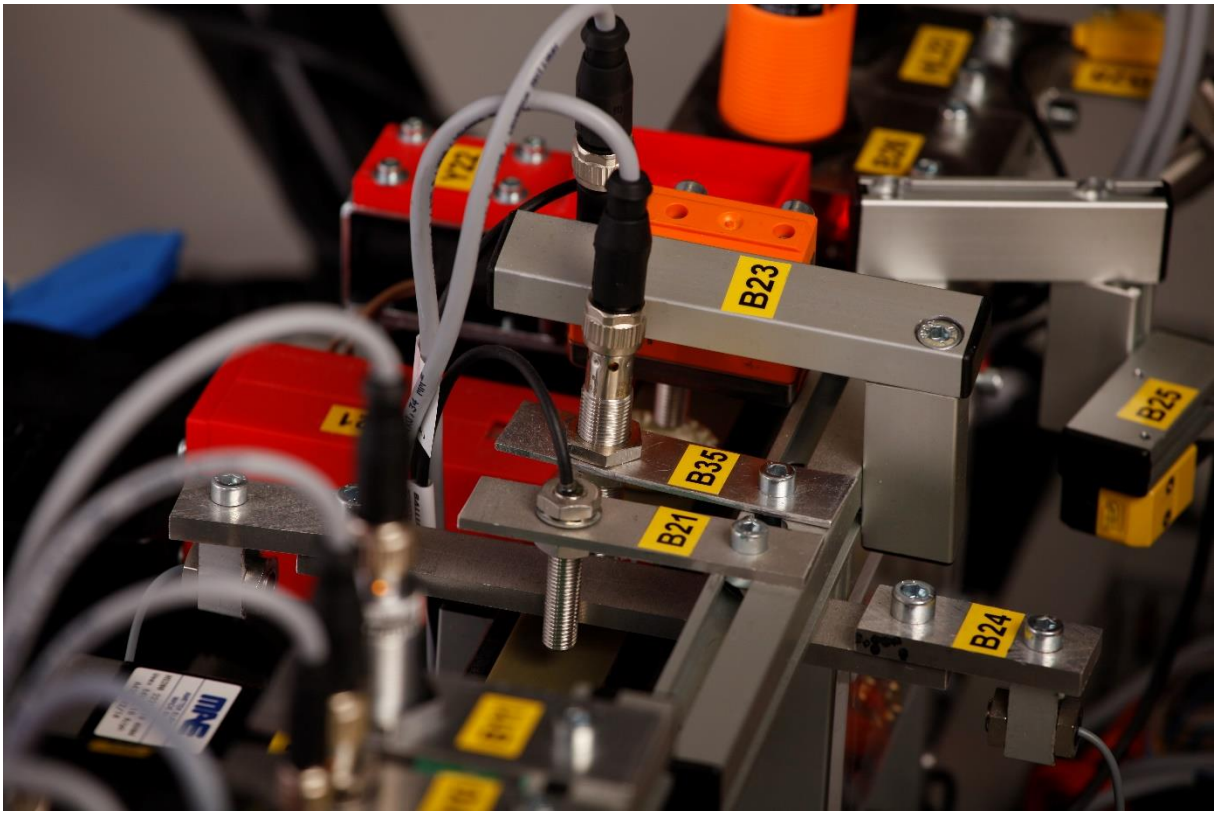


Obrázek 175: Senzory linky 1

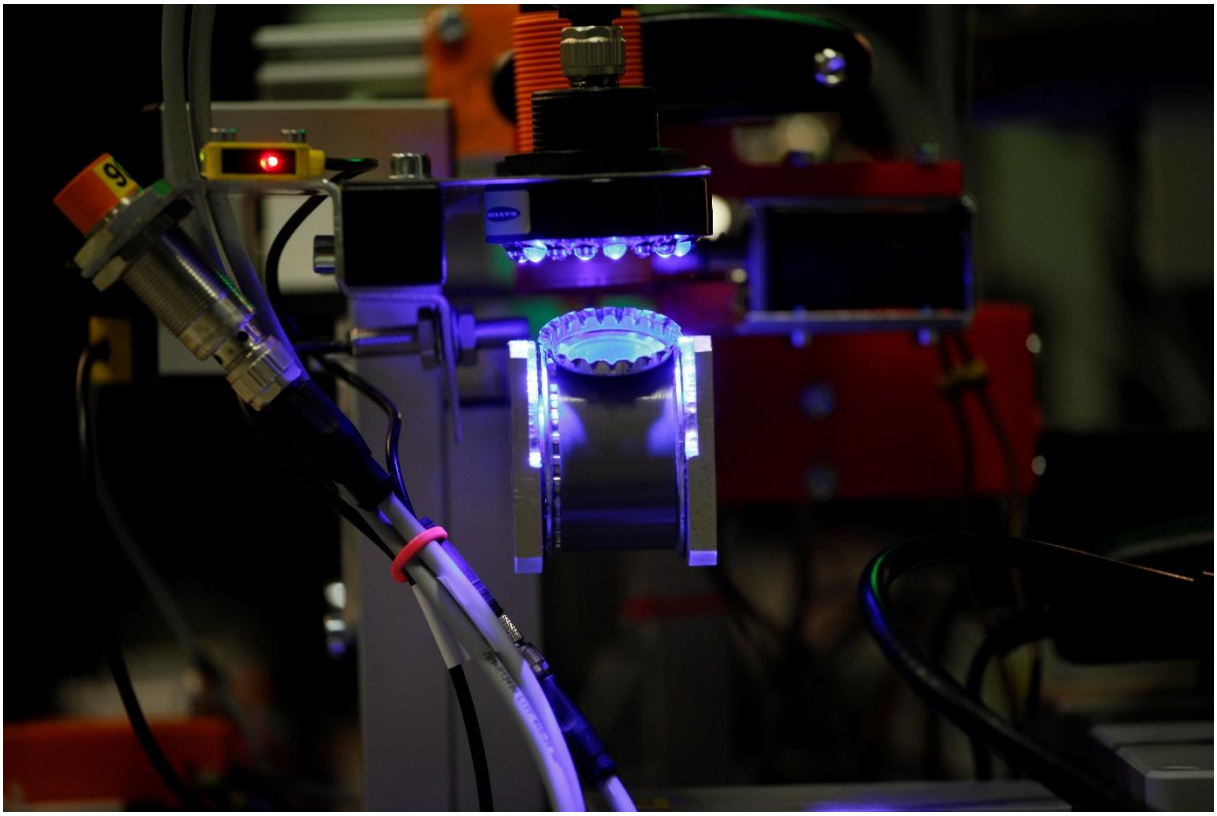
Obrázek 176: Senzory na lince 1



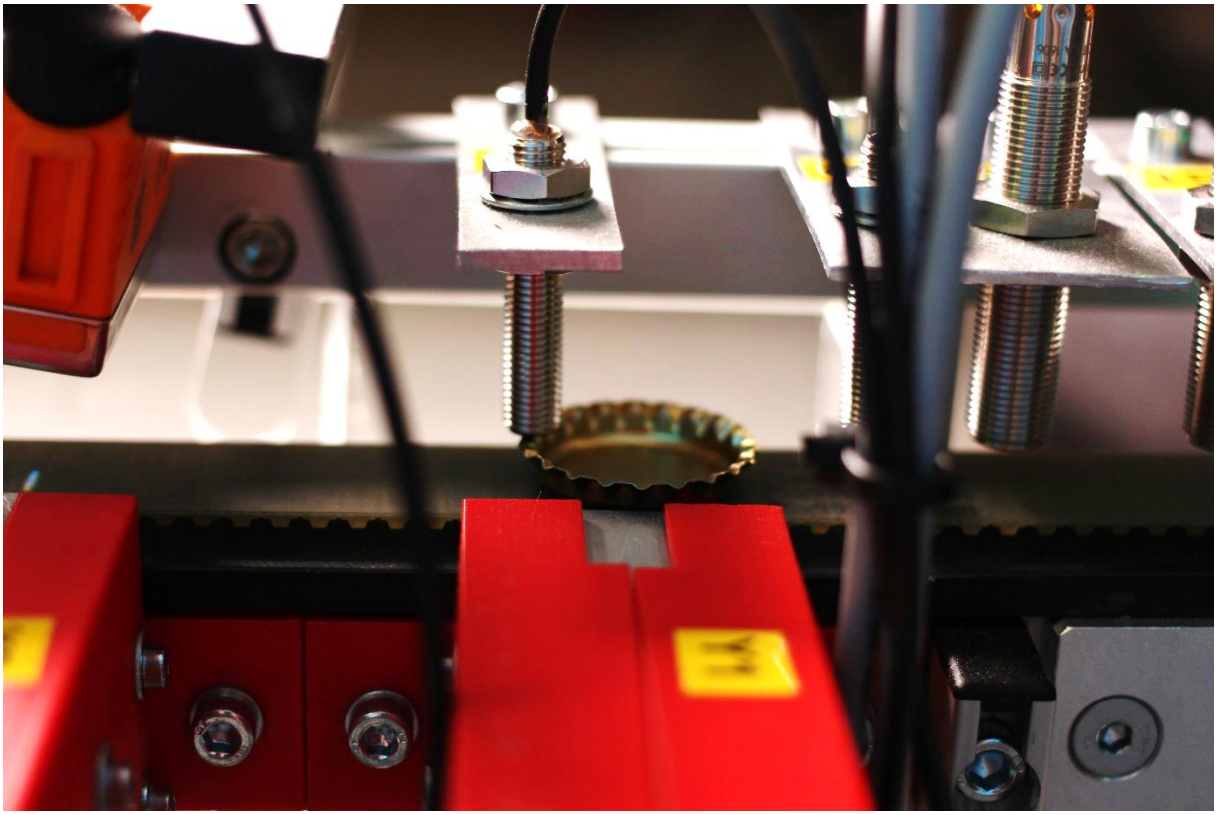
Obrázek 177: Trojice různých senzorů



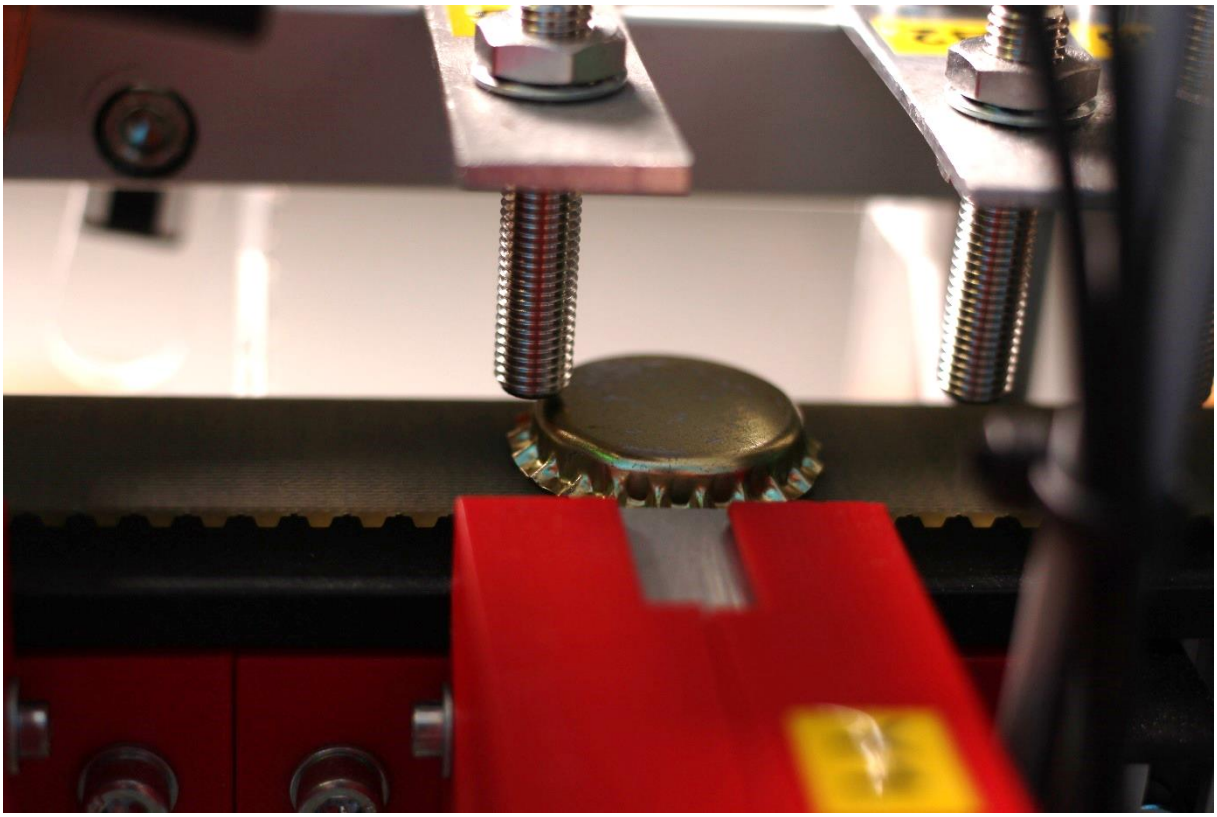
Obrázek 178: Senzory linky 2



Obrázek 179: Zakončení linky 2

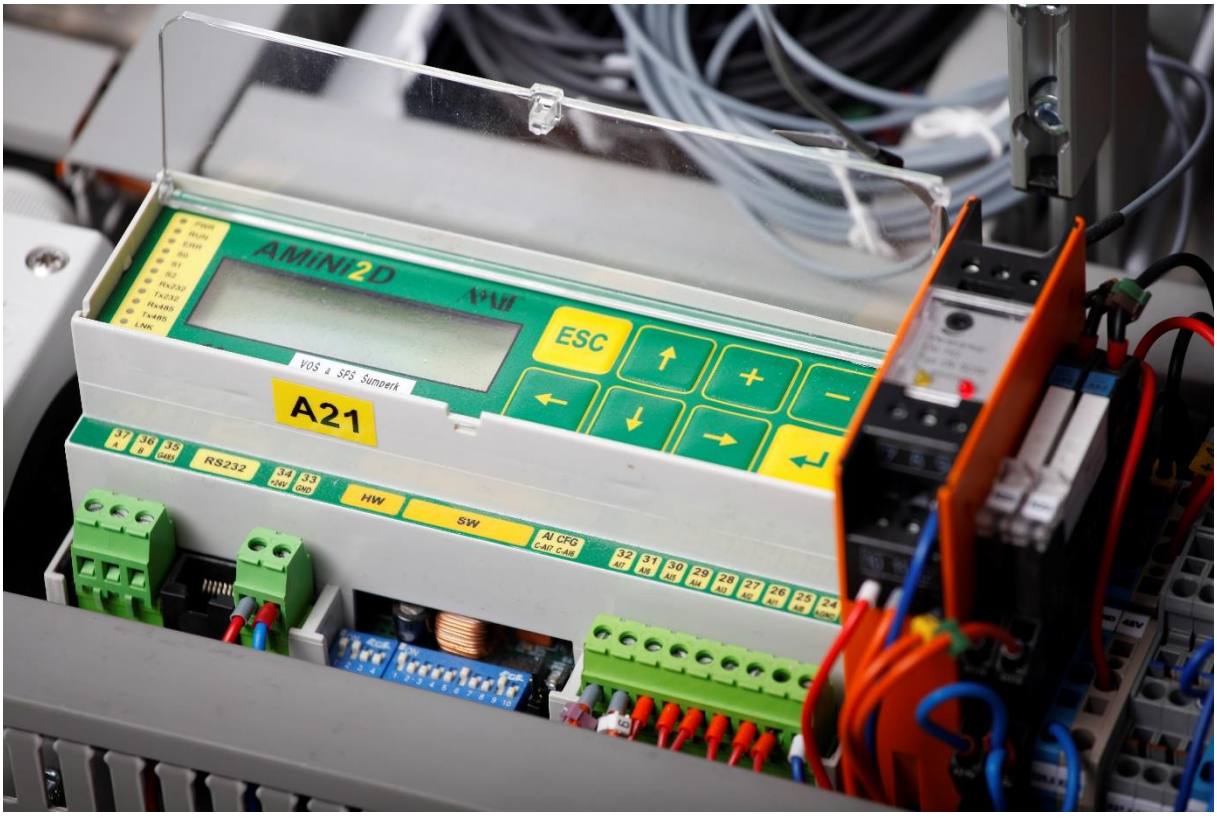


Obrázek 180: Vršek jedoucí na pásu



Obrázek 181: Otočený vršek jedoucí na pásu

Obrázek 182: Zapínací tlačítka silové části

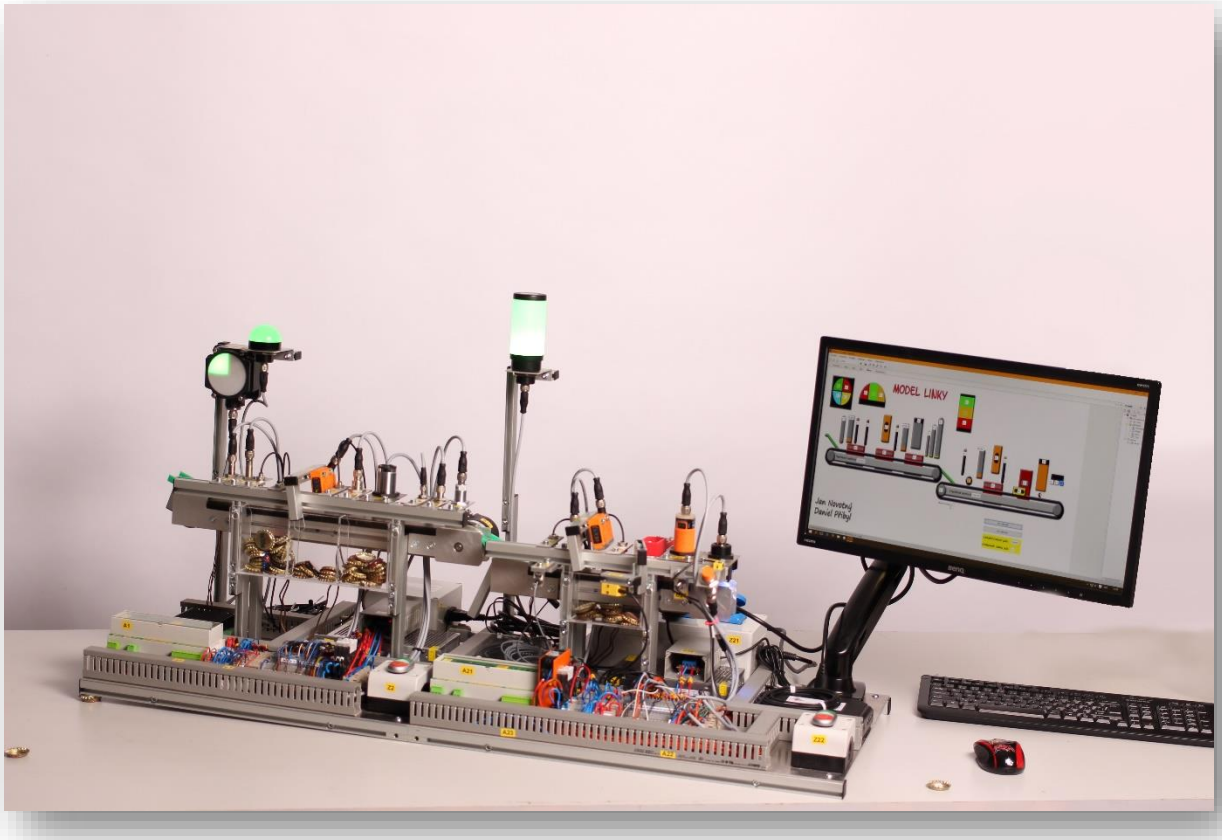


Obrázek 183: PLC AMiNi2D od firmy AMiT

15. Přílohy

Obrázky, datasheety a programy na přiloženém CD

Metodika výuky



Výukový model automatizační linky s PLC

Metodika výuky

Jan Novotný, Daniel Přibyl

VOŠ a SPŠ Šumperk

V rámci výroby linky je i součást výukový modul, kde jsou rozebrány jednotlivé úlohy.

Každá úloha se skládá z cíle, příkladu, realizace a programu.

V cíli je napsáno, co je cílem úlohy.

V příkladu je zadání úlohy, co má žák zpracovat a je doplněno o poznámky typu **!Pozor**, **?Nápověda**, **#Extra**. Příklady by měli žáci zpracovat i bez použití nápovědy. Úlohy jsou sestaveny tak, aby je každý dělal postupně, jelikož některé úlohy na sebe navazují. Úlohy nabývají i na náročnosti.

V programu je popsán vzorový program, jak by mohl program vypadat. Každý stejnou úlohu však může naprogramovat jinak. Příklady jsou především v RS (reléovém schématu) ale jsou kombinované se ST (strukturovaným textem). Vzorové řešení obsahuje podrobný popis úlohy i jak se úlohy rozvětvují, což by šlo zadávat jako jednotlivé úlohy, což by vedlo k programování jednoduchých částí a pak přidávání různých funkcí, jako je třeba u úloh s majáky, kde jde zapisovat a řídit výstupy více možnostmi. Jde to jak přes RS, tak i přes ST.

Dokumentace je doplněna o fotografie linky.

Studenti mohou pracovat jak samostatně, tak i ve skupinách. Linka jde rozdělat na dvě části, takže každá skupina by ji mohla mít před sebou, anebo je možnost spojení linek, kam studenti budou nahrávat programy přes ethernet, což je rychlejší a každý by si mohl vyzkoušet svůj vlastní program.

Každá skupinka by měla dostat výukový modul, ve kterém najde parametry senzorů, zapojení a příklady.

Modul obsahuje základy programování PLC. Úlohy jsou seřazeny podle obtížností a navazují na sebe, takže studenti by je měli řešit postupně.