Středoškolská technika 2011

Setkání a prezentace prací středoškolských studentů na ČVUT

Technické zabezpečení a řízení provozu v tunelu

Marek Partika

SPŠ ELEKTROTECHNICKÁ

Brno, Kounicova 16

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) uvedené v přiloženém seznamu.

V **Brně** dne **25. 4. 2011** Podpis: ……...

Poděkování:

Tato práce byla vypracována za pomoci Jihomoravského kraje.

Tímto bych chtěl poděkovat Jakubu Streitovi, který mně byl po celou dobu práce na projektu nápomocný a ochotně se mnou řešil danou technickou problematiku, především mikroprocesorů ATMEL.

Dále bych chtěl poděkovat Jaroslavu Faldíkovi z Brněnských komunikací, který mně ochotně seznámil s technickým zabezpečením a řízením provozu tunely. Byl jsem proveden řídicím střediskem, seznámen s reálným řešením krizových situací a bylo mi doporučeno nastudování technických podmínek, které jsem uvedl v použité literatuře.

Mimo výše uvedených bych v neposlední řadě poděkoval také Antonínu Veselému, který po absolvování SPŠ ELEKTROTECHNICKÉ i nadále školu navštěvuje za účelem vedení kroužku programování PLC firmy AMiT. Tento kroužek navštěvuji již od minulého školního roku (2009/2010).

Obsah:

1. Anotace

Mým úkolem bylo vypracovat projekt technické zabezpečení a řízení provozu v tunelu. Tento projekt jsem pojal jako příležitost, kdy se mohu přiučit novým věcem. Při řešení práce jsem ze začátku čerpal spousty informací z různých zdrojů, uvedených v použité literatuře. Projekt nelze srovnávat s realitou, ale při sestavování mé práce byla snaha co nejpřesněji vystihnout nejdůležitější články tunelů, které jsou pro silniční provoz skrze tunel nezbytné. Situace ať už hardwarové, nebo softwarové se blíží reálné stavbě, i když některá řešení se mohou jevit jako odlišná.

2. Úvod

Hned z úvodu bych vás chtěl upozornit, že stejné téma mnou bylo řešeno již v loňském kole soutěže. Ovšem letošní práci jsem začal vypracovávat kompletně od začátku a z loňského kola byly převzaty pouze informace, které jsem nasbíral.

2.1. Model silničního tunelu AMiTsys Expert 2010

Použil jsem řídicí systém AMiNi2D a informace byly čerpány především z internetových stránek. Při vytváření práce jsem se teprve učil základy programování v DetStudiu a práce s PLC. Na modelu (pouze jeden tubus s obousměrným provozem) bylo použito proměnné značení pouze na vjezdu a výjezdu z tubusu. Dále zde byly nainstalovány větráčky pro ventilaci a nakonec pouliční osvětlení. Pro simulaci byly použity pouze potenciometry a pro aktivace čidel DIP přepínače.

2.2. Pohled na letošní projekt

V letošním školním roce mnou byl vyroben nový dvoutubusový model. Při získávání nových informací jsem si domluvil schůzku s Jaroslavem Faldíkem (technickým specialistou) z Brněnských komunikací a.s., od kterého mně bylo doporučeno nastudování norem TP 154 a TP 98, které se týkají staveb, zabezpečení a řízení provozu v tunelu. Ochotně mě provedl dispečinkem a dokonce odsimuloval změny dopravního značení pro řízení provozu na jejich vizualizačním systému. Tyto informace pro mne byly při práci na projektu přínosem.

S projektem jsem se také účastnil SOČ. Pří mých konzultacích jsem se naučil základy programování mikroprocesorů ATMEL, které byly na projektu také použity. Mikroprocesory

ATmega16 byly využity k rozšíření digitálních výstupů. Za pomoci mého konzultanta jsem úspěšně zprovoznil komunikaci mezi AMiNi2D a ATmega16, tudíž jsem již nebyl omezen nedostatkem výstupů, kterých není nikdy dost. Mikroprocesory pouze přijímají data z PLC a jejich výstupy slouží k rozsvěcování proměnného značení, řízení SERV a displejů, na kterých je zobrazena maximální povolená rychlost. Projekt hlavně řeší řízení dopravy pomocí proměnného značení a to tak, že proměnné značení se nerozsvítí stavově v jednu chvíli, ale sekvenčně tak, jak je tomu v realitě. Stavy, které mohou nastat, jsou přesměrování dopravy do prvního nebo do druhého tubusu, tubusy uzavřeny nebo otevřeny a zároveň se jednotlivé pruhy mohou uzavírat, či otevírat. Více o novém pohledu na nynější projekt naleznete dále v dokumentaci a přílohách.

2.3. Popis projektu

V poslední době mě začala zajímat teorie řízení silničního provozu a převážně moderních inteligentních technických staveb. Z tohoto důvodu mě toto téma natolik zaujalo. Nejenom, že jsem se při práci na projektu seznámil se spoustou nových informací této problematiky, ba dokonce jsem se ve studentském životě přiučil práci s mikroprocesory ATMEL a dále jsem si rozšířil logické myšlení při návrhu různých algoritmů.

Projekt se neskládá pouze ze softwarové části, ale i z hardwarové - praktického modelu. Tento model představuje dvoutubusový komplex o dvou jízdních pruzích. Bezpečný provoz zajišťuje hlavní řídící jednotka AMINi2D od firmy AMIT, kterými naše škola disponuje. Tato jednotka obsahuje nejdůležitější část algoritmu a její funkce je pro zajištění bezpečného provozu nezbytná. Jako podřadné řídicí systémy jsem použil mikroprocesory ATmega16, které jsem se mimo jiné učil programovat. Komunikace mezi PLC a mikroprocesory je zajištěna pomocí RS485, kdy jsou určená data z PLC posílána po sériové lince mikroprocesorům. Pomocí nadefinovaného protokolu každý mikroprocesor přijme data jemu určená a dále s nimi pracuje. Mikroprocesory jsou zde použity hlavně z důvodů rozšíření výstupů, kterých je na moderních technologických zařízeních potřeba čím dál více. Tolika výstupů je zapotřebí z toho důvodu, protože tunel o dvou tubusech a dvou jízdních pruzích vyžaduje vetší množství regulované osvětlovací techniky, displejů, proměnných značení apod. Vraťme se tedy ještě k tomu, proč jsem jako další řídicí jednotky použil mikroprocesory. Je to z toho důvodu, že modul AMINi2D disponuje 8DI, 8DO, 8AI, 4AO. Z tohoto už je patrné, že bych byl při programování omezen nedostatkem vstupů a výstupů. Naše škola disponuje pouze šesti řídicími systémy AMiNI2D a jedním rozšiřujícím modulem

I/O. Z tohoto důvodu jsem se rozhodl využít mikroprocesorů, které jsou pro můj studentský projekt dostačujícím řešením.

PLC nebude pouze řídit provoz, ale zároveň to bude i hlavní řídící jednotka pro dispečink, který má dohled nad tunelem a pomocí nadefinovaných obrazovek má možnost ovládat provoz a bezpečnost tunelových trub. Programovatelné automaty jsou zde pomocnými zařízeními, která pomáhají dispečinku dohlížet nad bezpečným provozem. Ovšem při určitých krizových situacích přebírají kontrolu nad tubusy řídící automaty a to například při požáru apod. Při méně závažných situacích je pouze upozorněn dispečink, který danou situací vyhodnotí a dále zpracuje.

2.4. Tunely obecně

Ve skutečnosti se v tunelu nachází mnohem více bezpečnostních prvků, mimo ty které se nacházejí na mém projektu. Pokud se přihodí v některém z tunelů dopravní nehoda, ve většině případů mají pracovníci dispečinku možnost sledovat dění přímým přenosem pomocí kamer, které pokrývají prostor tunelů, mohou zareagovat buď okamžitým uzavřením tubusu, přestavěním variabilních dopravních značek, svedou provoz do volného pruhu a současně zalarmují potřebné složky záchranného systému – Zdravotní záchrannou službu, Hasičský záchranný sbor, Policii ČR a pomocí svých nástrojů jim zjednoduší přístupovou cestu k nehodě. Současně s informací dispečinku tunelů je upozorněna Hlavní dopravní řídící ústředna, která koordinuje dopravu, objízdné trasy, příjezd záchranných složek a další opatření. Ne všechna rozhodnutí jsou jenom na uvážení obsluhy. Tunely bývají vybaveny velkým množstvím různých čidel, které dodávají řídícímu počítači informace a počítač reaguje tak, jak je na různé situace naprogramován. Modelových situací, které si umí počítač představit, je nepřeberné množství. Rovněž funkčnost tohoto systému se automaticky prověřuje. Motoristé jsou čas od času zaskočeni technologickým uzavřením tunelů, ale právě v té době, většinou v nočních hodinách, pracovníci servisních firem kontrolují stav technologie a provádějí potřebné opravy.

Proč speciální bezpečnost v tunelu? Jízda v tunelu může pro řidiče a ostatní osoby v něm se pohybující znamenat riziko ohrožení života a zdraví. Toto riziko vyplývá z toho, že se v uzavřeném prostoru provozuje silniční doprava. Nejzávažnějšími ohroženími pak je požár nebo výbuch. Avšak běžná dopravní nehoda, která zapříčiní např. kolonu, je z hlediska ohrožení zdraví stejně nebezpečná, zvláště v případě, kdy nejsou dostatečně zajištěna všechna bezpečnostní a technologická zařízení mezi která patří únikové cesty, ventilace apod.

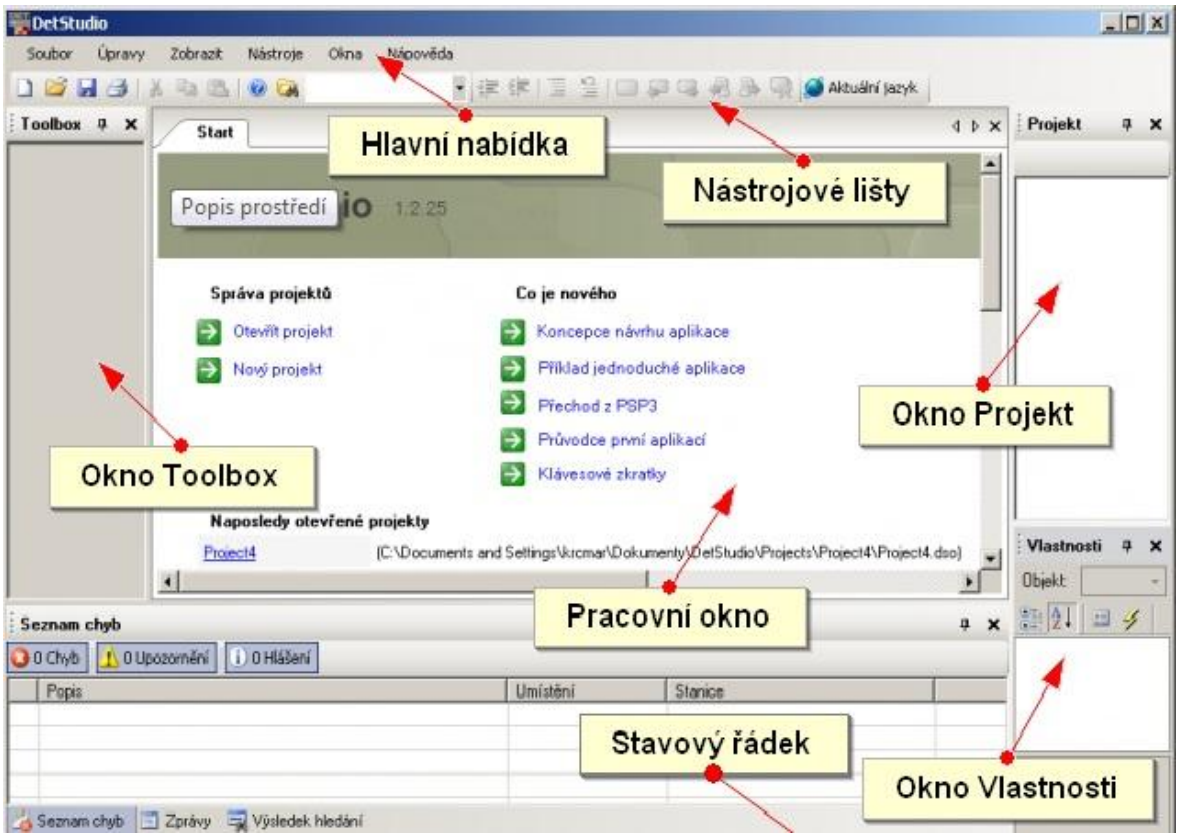
2.5. Účel staveb tunelů

Tunely jsou dnes velice rozšířené a jejich počet se stále zvyšuje. Tunel je dopravní stavba, která vede pod zemí skrz krajinou vyvýšeninu, pod mořem, říčním tokem či městem. Obvykle slouží pro silniční nebo kolejovou dopravu. Výstavba tunelů slouží pro zprovoznění jízdních tras, které se nachází např. v městech pod obydlenými oblastmi, v různých údolích, kde dopravní provoz křižují hory či nějaká vyvýšenina. Dnes už tunely můžeme najít i pod mořem, které např. spojují pevninu s ostrovem. Jeden takový známí tunel je nazýván Eurotunel a spojuje anglický Folkestone s francouzským Calais. Tunely bývají v nynější době dlouhé až několik kilometrů a uvnitř se nachází spousta automatizační techniky, která zajišťuje nejenom ideální podmínky provozu, ale i spuštění daných opatření pro případ nehody. Z hlediska bezpečnosti jsou tunely velice bezpečným místem. Jediným nebezpečím uvnitř tunelu bývají sami řidiči.

3. Vývojová prostředí

Stručný přehled vývojových prostředí použitých během zpracování projektu.

3.1.DetStudio



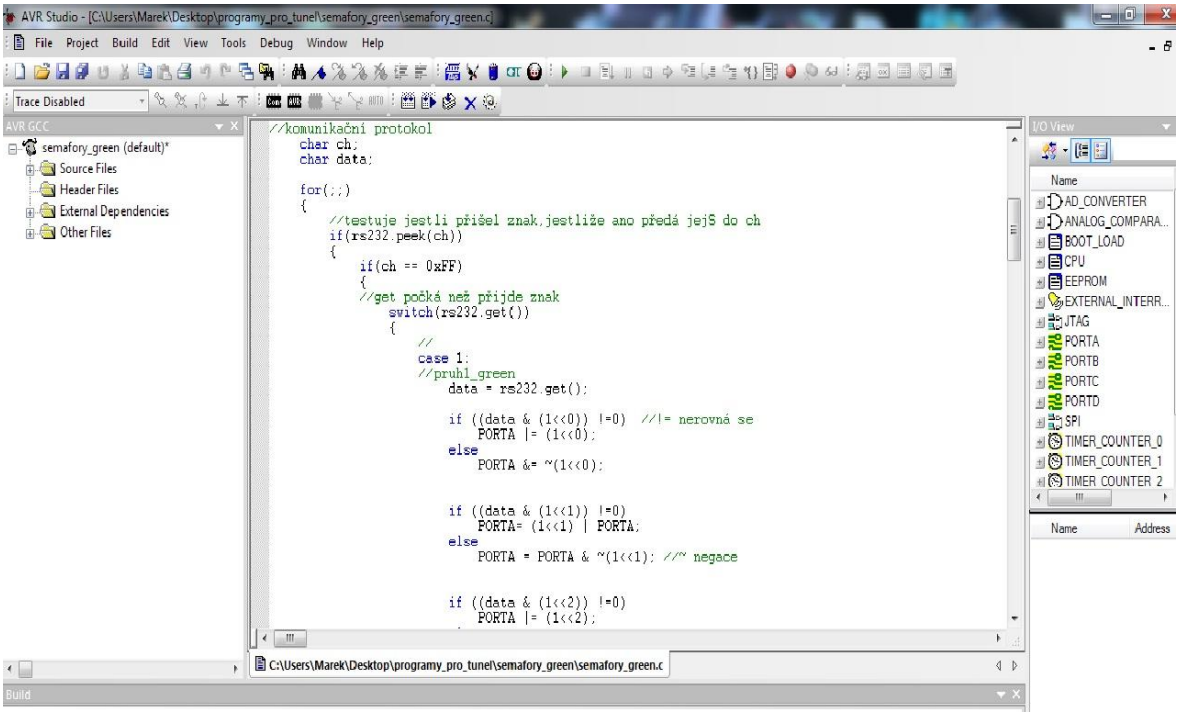
obr. 1 Vývojové prostředí.

Návrhové prostředí **DetStudio,** jak můžete vidět na *obr. 1* je určeno pro tvorbu uživatelských aplikací pro všechny standardní řídicí systémy firmy AMiT. V jediném vývojovém prostředí lze vytvořit vlastní aplikaci, navrhnout vzhled obrazovek zobrazovačů řídicích systémů, definovat chybová hlášení, on-line ladit běžící aplikaci, vytvořit dokumentaci vytvořeného programu. Způsob programování a algoritmizace vychází ze staršího osvědčeného parametrizačního prostředí PSP3 a na úrovni vstupních zdrojových kódů je s ním DetStudio kompatibilní.

Řídicí systémy se dají pomocí DetStudia programovat třemi typy jazyků, kterými jsou:

- Strukturovaný text
- Releová schémata
- Pomocí jazyka LA – jazyk podobný assembleru

3.2.AVR Studio



obr. 2 Vývojové prostředí AVR Studia

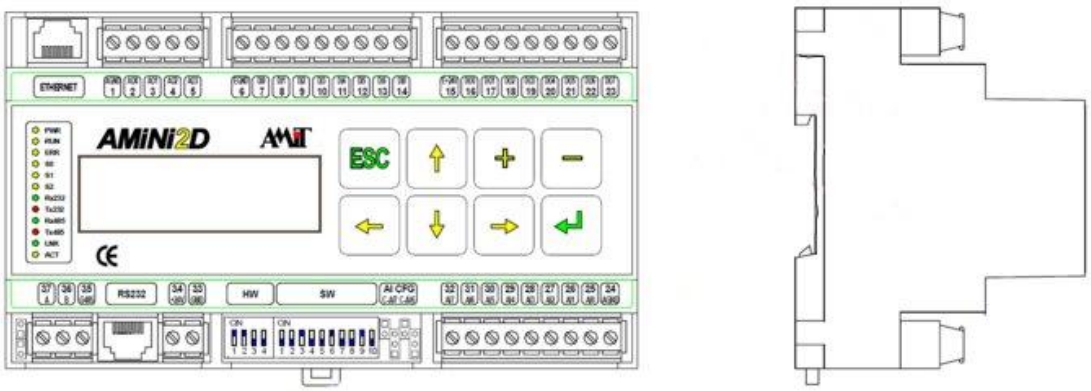
K naprogramování AVR mikroprocesoru je nutný AVR programátor a příslušný software pro kompilaci programového kódu do strojového kódu. Nejčastěji se používá nástroj vyvinutý firmou Atmel a to program *„AVR studio“*, jak můžete vidět na *obr. 2*. V tomto programu lze pak tvořit za pomocí GNU Assembleru, C/C++ program, který je po kompilaci skrze programátor nahrán do programové paměti mikroprocesoru.

4. Metodika

4.1. Řídicí systémy

Použité řídicí jednotky, které jsou na projektu využity.

4.1.1. AMiNi2D



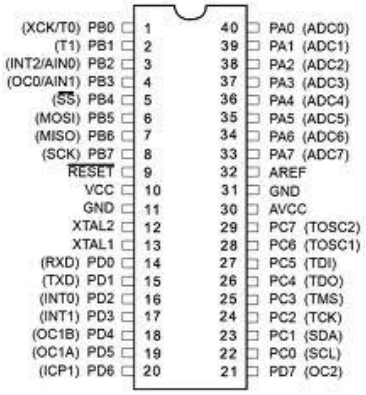
obr. 3 Řídicí systém AMiNi2D

Řídicí jednotka firmy AMiT, kterou jsem v mé práci použil jako hlavní řídicí systém. Jak je patrné z obrázku, systém disponuje grafickým displejem a osmi tlačítky. Tento řídicí systém dále obsahuje 8 DI, 8 DO, 8 AI, 4 AO. Komunikace mezi PC a řídicí jednotkou je zastoupena průmyslovým ethernetem nebo RS 232. V poslední řadě systém obsahuje komunikační linku RS 485, kterou jsem využil pro komunikaci s mikroprocesory, a dále se dá využít pro komunikaci s dalšími řídicími jednotkami, kterých je možno připojit až 32, což je dostačují pro mnoho aplikací.

4.1.2. ATmega16

Mikrokontroléry dnes najdeme skoro v každém elektronickém zařízení. Umožňují dálkové ovládání z PC, ukládání naměřených dat či jednoduchou realizaci ovládání a řízení, kterou by dříve bylo nutné realizovat mnoha logickými obvody.

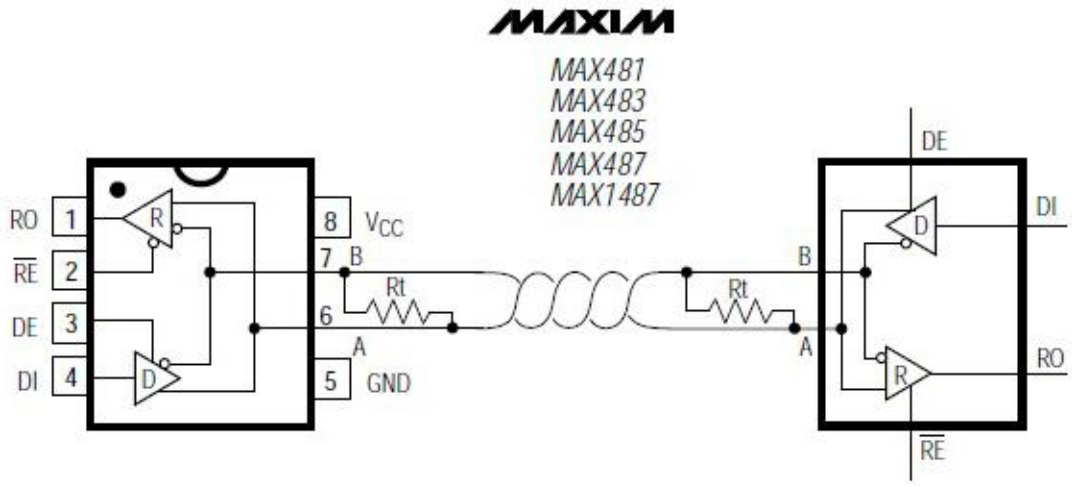
Vývoj se postupně ubíral od jednodeskových mikropočítačů, kdy byly veškeré potřebné obvody na jedné desce plošných spojů k mikropočítačům, které mají na jednom čipu mikroprocesor, paměť i obvody I/O. Integrace dále pokračuje tak, aby byla minimalizována potřeba vnějších obvodů. Na čipu jsou dále integrovány obvody čítačů, A/D převodníků, sériové linky a další obvody.



obr. 4 ATmega16

4.2.Komunikace

Komunikace mezi AMiNi2D a mikroprocesory ATmega16 je zprostředkována pomocí RS485 (sériová linka). Ke komunikaci je zapotřebí součástka MAX 485, která nám



obr. 5 MAX 485

komunikaci s mikroprocesory umožní. Tato součástka minimalizuje a redukuje přeslechy způsobené nesprávně ukončenými kabely, a tak dovoluje bezchybný přenos dat. MAX 485 neomezeně dovoluje přenášet rychlostí až do 2.5Mbps.

Další věcí ke komunikaci je knihovna pro AVR Studio, která mně umožní jednoduše nadefinovat komunikační parametry. Stačí zapsat komunikační rychlost, kde jsem si zvolil

(38000 Baud). Mikroprocesory mají napájení 5V, řídicí jednotka AMiNi2D 24V. Ke komunikaci jsou použity dvě přiložené knihovny #include "queue.h"; #include "rs232new-basic.h", které jsem získal od mého konzultanta, který mě seznamoval s navázáním komunikace. Komunikace by šla zprovoznit i bez použití těchto knihoven a to nastavením jednotlivých registrů ATmega16, které se dají nastavit podle datasheet.

4.2.1. Nastavení komunikace na AMiNi2D

Komunikační kód je v procesu INIT, který se vykoná právě jednou, a to po startu procesní stanice a ještě před spuštěním jakéhokoliv jiného procesu. Do tohoto procesu je také výhodné umisťovat veškeré inicializační sekvence, kterými se zajišťují počáteční podmínky běhu aplikace. Komunikace je nastavena pomocí funkčního bloku **ComInit.** Baudová rychlost je 38400 a rozsah jedno znaku je nastaven pouze na 8 bitů. Čímž se zmenší možnost chyby přenosu dat. Vliv na kvalitu přenosu má délka znaku, komunikační rychlost, propojení kabelem.

```
//návěstí/00003->RS485/1-kom.kanál/rychlost komunikace/délka znaku (8
bitů)/žádná parita (kontrola jestli není chyba)/stop bity žádné
:01000 ComInit 0x0003, 1, 38400, 8, 0, 1, :NONE, :NONE, :NONE, :NONE, NONE,
ComInit_Tx
```

4.2.2. Přenos dat po sériové lince

Data jsou přenášena pouze jedním směrem, a to z řídicí jednotky AMiNi2D do mikroprocesorů ATmega16. Přenos dat je naprogramován v procesu s názvem prenos_dat.

Komunikační protokol:

```
let prenos_dat[0,0] = 0xFF    //0xFF jsem zvolil jako označení nového paketu
let prenos_dat[0,1] = 1       //číslo příchozího paketu
let prenos_dat[0,2] = pruh1_green   //přenášená proměnná; posílá se pouze 8
bitů, což je nastaveno ve funkčním bloku ComWrite->je to z toho důvodu, že
přenos 8 bitů nevykazuje chybu přenosu dat (na což má vliv i komunikační
rychlost), která může nastat

let prenos_dat[0,3] = 0xFF
let prenos_dat[0,4] = 2
let prenos_dat[0,5] = pruh2_green

let prenos_dat[0,6] = 0xFF
let prenos_dat[0,7] = 3
let prenos_dat[0,8] = pruh3_green

let prenos_dat[0,9] = 0xFF
```

```
let prenos_dat[0,10] = 4
let prenos_dat[0,11] = pruh4_green

let prenos_dat[0,12] = 0xFF
let prenos_dat[0,13] = 5
let prenos_dat[0,14] = pruh1_red
            .
            .
            .
            .
```

Funkční blok zajišťující přenos dat:

Návěstí zajišťuje propojení s funkčním blokem **ComInit** , kde je nastavena komunikace. Přenos dat je zajištěn funkčním blokem **ComWrite** , kde je nastaven rozsah vysílacího bufferu.

```
//      návěstí, přenášený byte (prenos_dat), délka bufferu (0-45)
ComWrite :01000, prenos_dat, 0, 45, NONE, NONE
```

4.2.3. Příjem dat v AVR Studiu

```
#include<avr/io.h>

#include <avr/interrupt.h>

#include<util/delay.h>   //knihovna pro zpoždění

#include "queue.h"

#include "rs232new-basic.h"

using kubas::rs232;

int main ()

        {
                DDRA = 127; //pruh1_red
                DDRB = 127; //pruh3_red
                DDRC = 255; //na PORT PC7  bude 5.Bit pruh4_red
                DDRD = 252; //pruh4_red od port PD2-PD7 (PD0,PD1 jsou Tx,Rx)

        //inicializace komunikace zajišťuje vše za mě
        //zadám pouze komunikační rychlost, která musí být totožná s AMiNI2D
                rs232.init(38400);
        //zapne přerušení
                sei();
        //komunikační protokol
                char ch;
                char data;
```

```
for(;;)
{
        //testuje jestli přišel znak,jestliže ano předá jej do ch
        if(rs232.peek(ch))
        {
                if(ch == 0xFF)
                {
                //get počká než přijde znak
                        switch(rs232.get())
                        {
                                case 5:
                                //pruh1_red
                                        data = rs232.get();
                                        if ((data & (1<<0)) !=0)
                                                PORTA |= (1<<0);
                                        else
                                                PORTA &= ~(1<<0);
                                        break;
                        }
                }
        }
}
```

4.2.4. Příklad použitého kódu v DetStudiu

```
//výpis části kódu z procesu
//kód pro povolování/zakazování jednotlivých jízdních pruhů
//pokud přijde impuls @pruh_X poprvé, tak se pruh uzavře
//jestliže po uzavření tento impuls opět přijde -> jízdní pruh se otevře
//a stále dokola je možno pruh povolovat/zakazovat
Let @pruh_one = (stav==1) and @pruh_1 and (stav_1==2)
let @pruh_two = (stav==1) and @pruh_2 and (stav_2==2)
let @pruh_three = (stav==1) and @pruh_3 and (stav_3==2)
let @pruh_for = (stav==1) and @pruh_4 and (stav_4==2)


If @pruh_one
      //jestliže stav==2 and přišel impuls @pruh_1 => stav_1==1
      //zapni pruh -> povolí jízdu daným pruhem
      let stav_1=1
else
      //podle příchozího impulsu se nastaví stav_1==2 =>zakáže jízdu daným
      //pruhem
      //z @normal stav=>vypni pruh
      let stav_1=if ((@pruh_1) and (stav==1),2,stav_1)
endif

If @pruh_two
      let stav_2=1
else
      //z normal stav=>vypni pruh
      let stav_2=if ((@pruh_2) and (stav==1),2,stav_2)
endif
```

4.3.Skriptování

Pomocí skriptu lze v editoru obrazovek, na základě událostí, které v řídicím systému nastaly (Tento proces je spouštěn vždy v okamžicích, kdy si žádný jiný proces nenárokuje procesor stanice. Je výhodné umisťovat v tomto procesu sekvence o minimální prioritě), ovlivňovat vlastnosti prvků na obrazovkách a hodnot proměnných či aliasů. Takovým ty pzpůsobem lze např. dynamicky ovlivňovat sled jednotlivých obrazovek, vzhled jednotlivých prvků.

Bez skriptu je možné s Basic prvky DetStudia vytvářet rozsáhlé, avšak funkčně omezené aplikace spíše ovládacího typu (s omezenou zpětnou odezvou). Pozice prvků, jejich viditelnost, jazyk a další vlastnosti jsou předem dány, sled jednotlivých obrazovek nelze dynamicky ovlivňovat. Bez skriptu lze jen těžko vytvářet reakce na chybové stavy a jejich následné kvitace.

Pří správném nasazení skriptu jsme schopni vytvářet mnohem efektnější a funkčně bohatší aplikace, než s Basic prvky. Obecně lze říci, že funkčnost, která není přímo obsažena v prvcích pro parametrizaci obrazovek v DetStudiu lze doprogramovat pomocí skriptu.

K prvkům na obrazovkách se přistupuje jako k objektům, které mají své jméno (shodné se jménem prvku), své vlastnosti a metody.

4.3.1. Příklad skriptu

//skript obrazovky podmenu_dennich planu

```
event podmenu_dennich_planu_OnOpen()
      podmenu_dennich_planu.FocusFirstControl();
end;


event Menu1_Item0_OnPressEnter()
// po stisknutí enter na Item0 se zobrazí daná obrazovka
nastaveni_planu.Show();
end;

//zobrazí obrazovku podle letního/zimního času
event Menu1_Item1_OnPressEnter()

      if @letni_cas then
      graf_cas_planu_letni.Show();
else
      if @zimni_cas then
            graf_cas_planu_zimni.Show();
      endif;
endif;

end;
```

```
//skript obrazovky GLOBAL

event Global_OnRefresh()
//jestliže bude aktivován alarm, tak jsme automaticky přepnuti
//na obrazovku alarmy, kde se nám zobrazí všechny alarmové hlášky
if alarm > 0 then
   alarmy.Show();
EndIf;

end;
```

4.4. Přehled proměnných

Rozbor navázání proměnných/bitů na jednotlivé vstupy a výstupy.

4.4.1. Režim NORMAL

Režim normal, při kterém je provoz tunelem povolen. Další možností při režimu normal je možnost zakazovat/povolovat jízdu danými jízdními pruhy.

Stav_1 == 1 …… Povol jízdu prvním jízdním pruhem

Stav_1 == 2 …… Zakaž jízdu prvním jízdním pruhem

Stav_2 == 1 …… Povol jízdu druhým jízdním pruhem

Stav_2 == 2 …… Zakaž jízdu druhým jízdním pruhem

Stav_3 == 1 …… Povol jízdu třetím jízdním pruhem

Stav_3 == 2 …… Zakaž jízdu třetím jízdním pruhem

Stav_4 == 1 …… Povol jízdu čtvrtým jízdním pruhem

Stav_4 == 2 …… Zakaž jízdu čtvrtým jízdním pruhem

4.4.2. Režim END

Režim normal, při kterém je provoz zakázán tunelem. Další možností při režimu normal (podobně jako u režimu END) je možnost zakazovat/povolovat jízdu daným pruhem.

Stav_1 == 4 …… Povol jízdu prvním jízdním pruhem

Stav_1 == 3 …… Zakaž jízdu prvním jízdním pruhem

Stav_2 == 4 …… Povol jízdu druhým jízdním pruhem

Stav_2 == 3 …… Zakaž jízdu druhým jízdním pruhem

Stav_3 == 4 …… Povol jízdu třetím jízdním pruhem

Stav_3 == 3 …… Zakaž jízdu třetím jízdním pruhem

Stav_4 == 4 …… Povol jízdu čtvrtým jízdním pruhem

Stav_4 == 3 …… Zakaž jízdu čtvrtým jízdním pruhem

4.4.3. Matice

Matice s názvem green_normal je pro proměnné značení, kdy je provoz jízdními pruhy povolen. Každý sloupec matice je pro jeden jízdní pruh.

MI green_normal [ř*s]; green_normal[8*4]

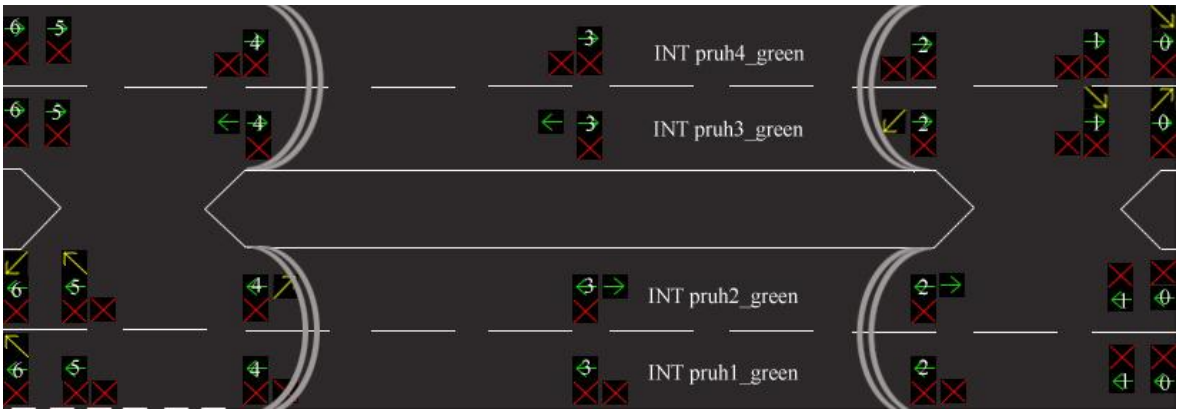
green_normal [a, 0] - > pruh1_green

green_normal [b, 0] - > pruh2_green

green_normal [c, 0] - > pruh3_green

green_normal [d, 0] - > pruh4_green

Podle hodnot proměnných **a,b,c,d** jsou určovýny dané řádky sloupce. Hodnota daného řádku je vždy přiřazena proměnné pro zvolený jízdní pruh.



obr. 6

Na obrázku (*obr. 6)* můžete vidět rozložení jednotlivých bitů. Obr. 6 je určen pro proměnné značení zelených šipek, které povolují jízdu daným pruhem.

MI red_normal [ř*s]; red_normal[8*4]

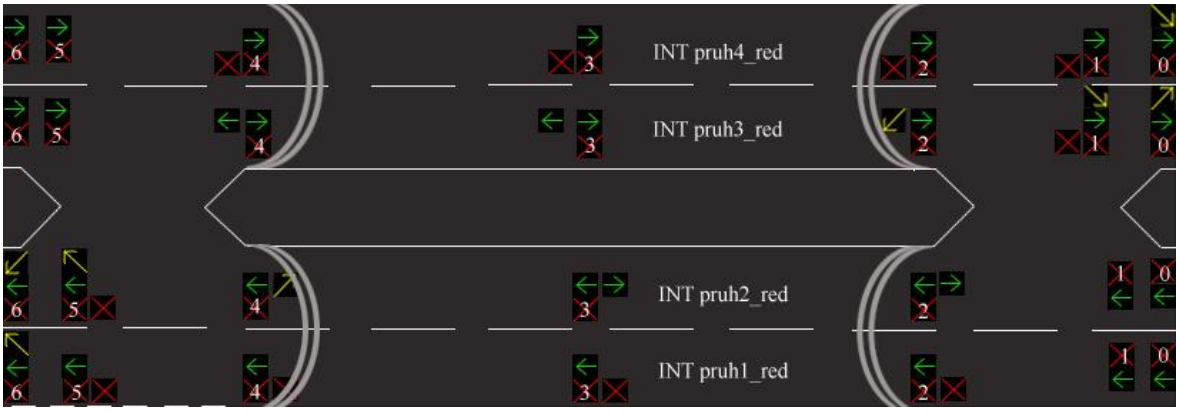
red_normal [e, 0] - > pruh1_ red

red _normal [f, 0] - > pruh2_ red

red _normal [g, 0] - > pruh3_ red

red _normal [h, 0] - > pruh4_ red

Podle hodnot proměnných **e,f,g,h** jsou určovýny dané řádky sloupce. Hodnota daného řádku je vždy přiřazena proměnné pro zvolený jízdní pruh.



obr. 7 Rozložení jednotlivých bitů na danou LED proměnného značení

Na obrázku (*obr. 7)* můžete vidět opět rozložení jednotlivých bitů. Obr. 7 je tentokrát určen pro proměnné značení červených křížků, které zakazují jízdu daným pruhem.

MI opacny_smer [ř*s]; opacny_smer [8*4]

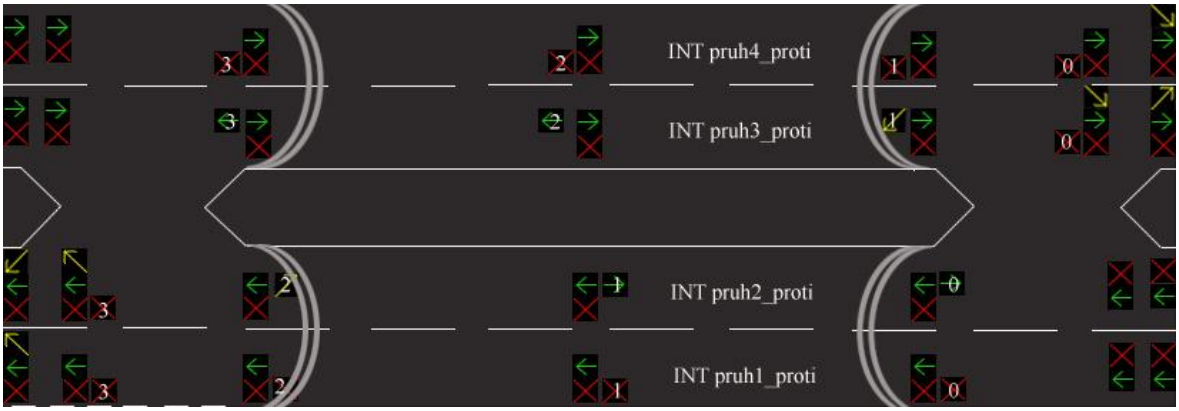
opacny_smer [w, 0] - > pruh1_ proti

opacny_smer [x, 0] - > pruh2_ proti

opacny_smer [y, 0] - > pruh3_ proti

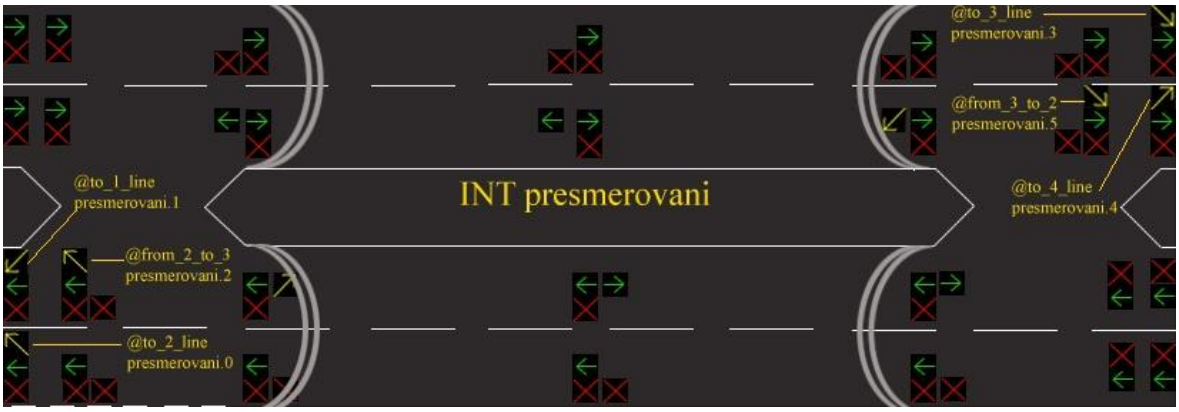
opacny_smer [z, 0] - > pruh4_ proti

Podle hodnot proměnných **w,x,y,z** jsou určovýny dané řádky sloupce. Hodnota daného řádku je vždy přiřazena proměnné pro zvolený jízdní pruh.



obr. 8 Rozložení jednotlivých bitů na danou LED proměnného značení

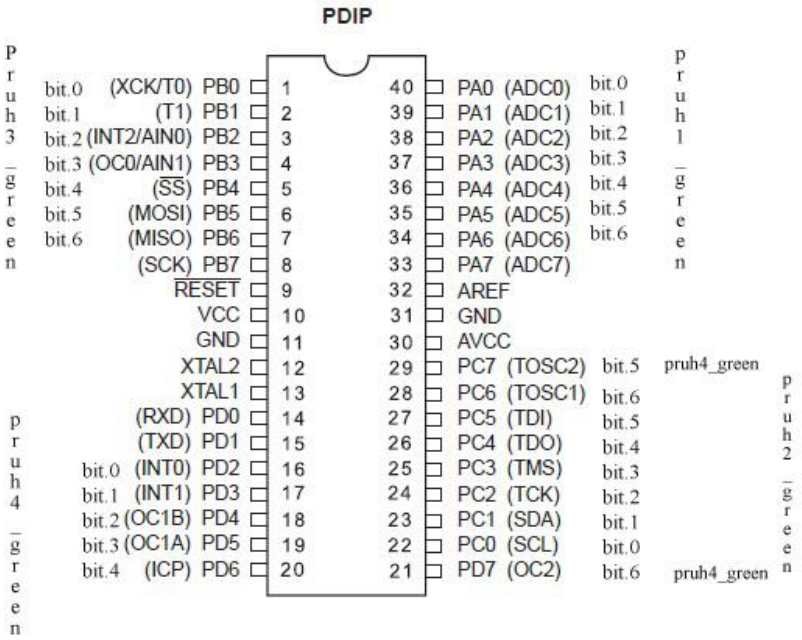
Na obrázku (*obr. 8) můžete* vidět opět rozložení jednotlivých bitů. Obr. 8 je určen pro proměnné značení v protisměru, které je spuštěno při přesměrování dopravy do jednoho z tubusů.



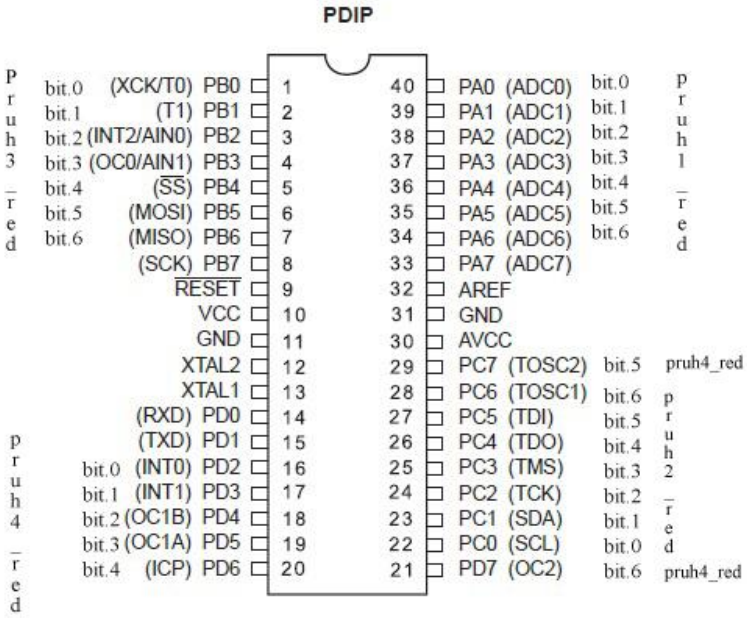
obr. 9 Rozložení jednotlivých bitů na danou LED proměnného značení

Na *obr. 9* můžete vidět rozložení jednotlivých bitů pro přesměrování dopravy do daných jízdních pruhů.

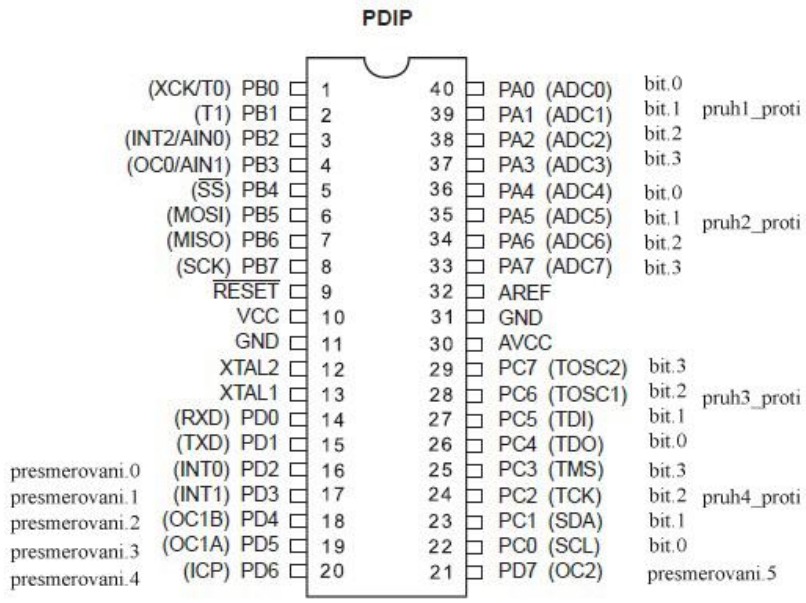
4.4.4. Výstupy mikroprocesorů



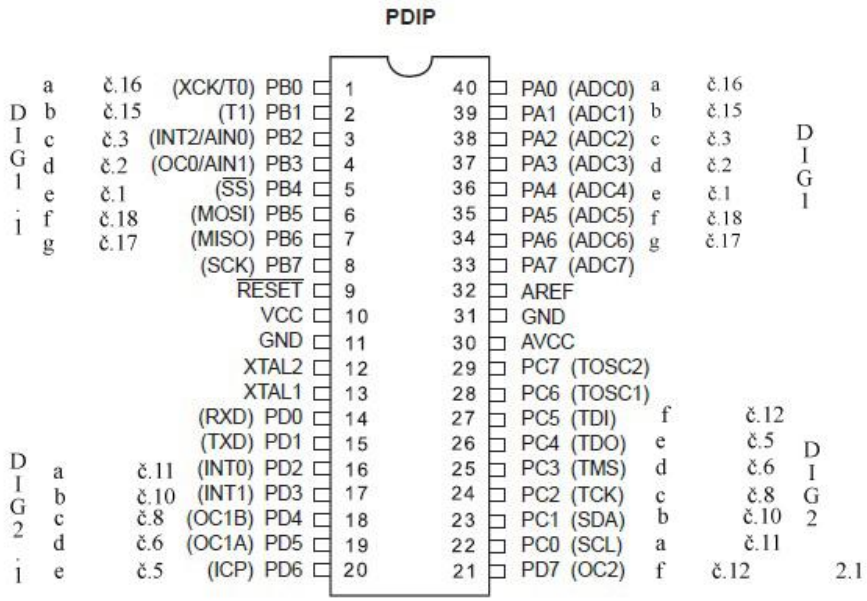
obr. 10 Výstupní nožičky pro ovládání proměnného značení (zelené šipky)



obr. 11 Výstupní nožičky pro ovládání proměnného značení (červené křížky)



obr. 12 Výstupní nožičky pro ovládání proměnného značení (protisměr)



obr. 13 Výstupní nožičky pro ovládání dvoumístných displejů (povolená rychlost)

4.4.5. I/O AMiNi2D

Pozn.:

(DI - digitální vstup, DO - digitální výstup, AnIn - analogový vstup, AnOut – analogový výstup)

- DI7 - `@SOS4Tub2 // SOS výklenek na výjezdu z 2. tubusu`
- DI6 - `@SOS3Tub2 // SOS výklenek na vjezdu do 2. tubusu`
- DI5 - `@SOS2Tub1 // SOS výklenek na výjezdu z 1. tubusu`
- DI4 - `@SOS1Tub1 // SOS výklenek na vjezdu do 1. tubusu`
- DI3 - `@vychod_2 // dveře únikového východu ve 2. tubusu`
- DI2 - `@vychod_1 // dveře únikového východu v 1. tubusu`
- DI1 - `@vjezdtubus2 // světelná závora na vjezdu do 2. tubusu`
- DI0 - `@vjezdtubus1 // světelná závora na vjezdu do 1. Tubusu`

- DO7 – `@LampyUlice //pouliční osvětlení`
- DO6 - `@signalizace //světelná signalizace alarmu`
- DO5 – nic nenavázáno

- DO4 - `@Vjezd2Tubus //ventilátor na vjezdu do 2. tubusu`
- DO3 - `@Vyjezd2Tub //ventilátor na výjezdu z 2. tubusu`
- DO2 - `@TlakUnik //ventilátor v únikovém východ (udržuje tlak)`
- DO1 - `@Vyjezd1Tub //ventilátor na výjezdu z 1. tubusu`
- DO0 - `@Vjezd1Tubus //ventilátor na vjezdu do 1. Tubusu`

- AnIn0 – `intenzita //fotorezisto, který měří intenzitu osvětlení`
- AnIn1 – `teplota_1 //teplota 1. Část 1. tubusu`
- AnIn2 - `teplota_2 //teplota 2. Část 1. tubusu`
- AnIn3 - `teplota_3 //teplota 1. Část 2. tubusu`
- AnIn4 - `teplota_4 //teplota 2. Část 2. tubusu`

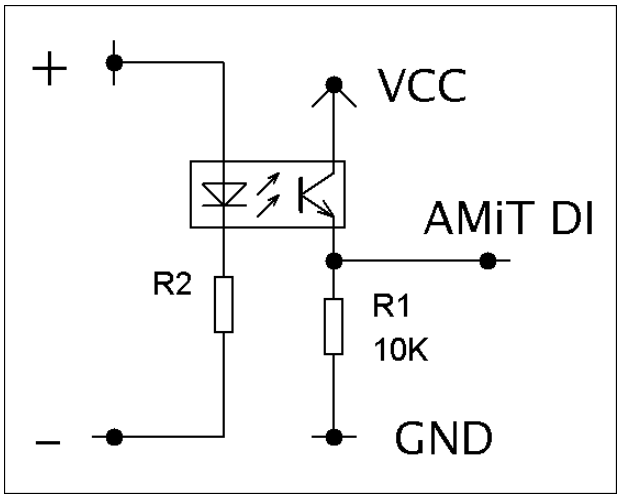
- AnOut0 - `osvetleni_1 //intenzita osvětlení v 1. části`

- AnOut1 - `osvetleni_2 //intenzita osvětlení ve 2. části`
- AnOut2 - `osvetleni_3 //intenzita osvětlení ve 3. části`
- AnOut3 - `osvetleni_4 //intenzita osvětlení ve 4. části`

4.5. Použité prvky

4.5.1. Světelná závora

Světelná závora je umístěna v místech u vjezdu do tunelového komplexu. Do tunelu mohou vjet pouze vozidla s povolenou bezpečnou výškou. Jakmile nastane případ, že některý řidič se zapomena a začne vjíždět do tunelové trouby a nemá povolenou výšku vozidla, tak je okamžitě přerušen světelný IR paprsek v uvedené výšce a ihned po této události je okamžitě upozorněn dispečink, který danou situaci vyhodnotí a dále řeší. Řešení spočívá v tom, že zakáže ostatním vozidlům jízdu v daném jízdním pruhu, nebo přesměruje dopravu do jedné tunelové trouby a v případě akutní nouze má dispečink možnost dopravu odklonit mimo tunely, nebo zcela pozastavit.

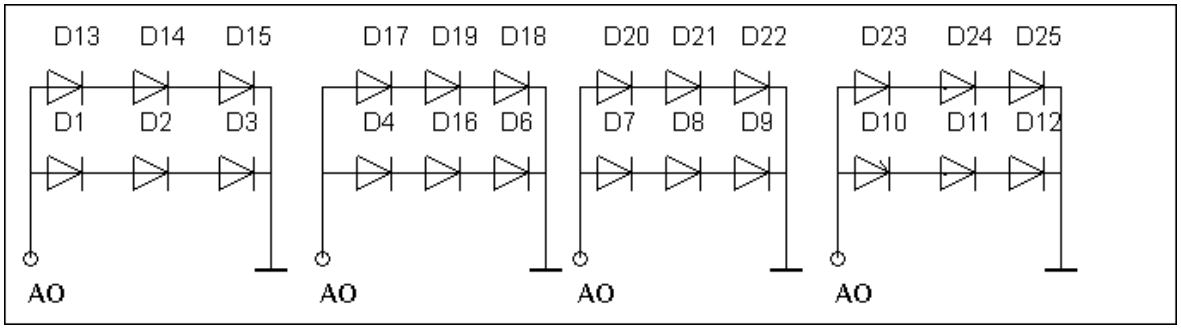


Schematické zapojení světelné závory

4.5.2. Osvětlení tunelového komplexu

Osvětlení tunelu je rozděleno v každém tubusu na čtyři části. Tohle osvětlení je regulovatelné a jeho regulace se odvíjí podle světla mimo tunelový komplex. Pro zjišťování citlivosti světla jsem v mém projektu zvolil fotorezistor, který je přiveden na AI na řídící jednotku AMiNi2D. Vstupní signál se vyhodnotí a podle toho se odvíjí regulace citlivosti osvětlení v tunelovém komplexu. Je to použito z toho důvodu, aby se řidič mohl přizpůsobit světelným podmínkám, které vzniknou při vjezdu do tunelového komplexu.

Pokud třeba řidič v létě vjede do tunelové trouby, kdy je jasno a na obloze jasně září slunce, tak aby řidič nevjel do nárazově temnějšího prostředí. Je to naprogramováno tak, že v první části je o něco méně citlivé světle, ovšem tak, aby to pro řidiče nebyla přílišná změna. Ve druhé a třetí fázi je osvětlení nejslabší a ve čtvrté části opět citlivost zesiluju z toho důvodu, aby po výjezdu z tubusu řidič pocítil co nejmenší změnu intenzity světla pro oči a neovlivňovalo to jeho soustředění se na jízdu. Tohle zařízení sice nezajišťuje dokonalý světelný přechod mezi prostředím, ale snaží co nejvíce přizpůsobit tak, aby řidiči pocítili co nejmenší změnu.



obr. 14 Zapojení osvětlení do čtyř regulovatelných částí

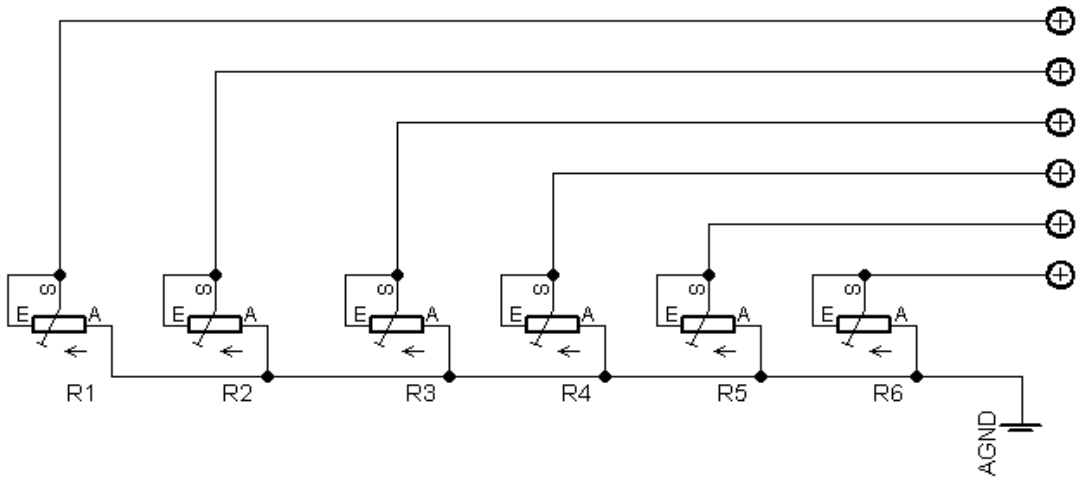
4.5.3. Pouliční osvětlení

Pouliční osvětlení, které je na modelu vyrobeno formou pouličních lamp. Pouliční osvětlení je naprogramováno pomocí prvku DAYPLAN, kdy jsem si do matice nadefinoval potřebné časy a v programu jsem si určil od kdy, do kdy budou pouliční lampy zapnuty a naopak. Dále jsem si vytvořil obrazovku, kde jsem využil graf pro denní plán pomocí prvku GPLAN, kde je možno vidět časovou osu a stav vypnuto či zapnuto, případně je zde také možnost manuálního zapínání a vypínání osvětlení. Pouliční osvětlení je zde řešeno pomocí LED diod s předřadným odporem.

4.5.4. Měření teploty

Čidla měření teploty jsou v projektu simulovány dvěma potenciometry pro každý tubus. Potenciometry jsem použil z důvodu lepší a rychlejší simulace. V případě originálního čidla bych zřejmě využil čidlo Ni1000, které bych v řídící jednotce naprogramoval pomocí jednoho řádku, poněvadž tento řídicí systém tyto čidla podporuje. Příkaz by vypadal takto (`Ni1000 #0.0, teplota, 6180`). Kde `#0.0 je analogový vstup, teplota je proměnná typu float nebo matice float, 6180 – citlivost snímače.`

A dané čidlo už by se dále jen připojilo na AI. Jak je zřejmé, tak pomocí originálního čidla by se nedaly dokonale simulovat krizové stavy, proto volba potenciometrů. Každý tubus bude mít pro simulaci tedy tři potenciometry. Měření teploty funguje tak, že pokud během několika sekund bude zjištěn prudký nárůst teploty, tak je to považováno za krizovou situaci např. požár a tubusy jsou ihned automaticky uzavřeny, čímž se provoz úplně zastaví. Dále je spuštěna potřebná ventilace, únikové východy se přetlakují vzduchem (z toho důvodu, aby při otevření dveří se nedostali zplodiny do únikových cest), spustí se potřebná osvětlení a upozorní se dispečink, který danou situaci vyhodnotí a dále zpracuje. Pro dispečink jsou na AMINi2D naprogramovány detailní obrazovky pro řízení tunelového komplexu.



obr. 15 Zapojení potenciometrů (první čtyři pro simulaci teploty)

4.5.5. Proměnné dopravní značení

Proměnné dopravní značení, které dovoluju dispečinku plynulý zásah do dopravního provozu a to tak, že dispečink pomocí nadefinovaných obrazovek na řídicím systému AMINi2D zvolí dopravní stav, mezi které patří:

- Normální provoz (každý tubus má dva jízdní pruhy v jednom směru)

- Tubusy uzavřeny
- Jednotlivé pruhy se dají povolovat a zakazovat pokud je stav normal (provoz povolen) nebo end (provoz zakázán), kdy dispečink může povolit jízdu danými jízdními pruhy
- Přesměrování dopravy do jednoho či druhého tubusu
- Přesměrování dopravy mimo tunelový komplex

Mezi další proměnná značení patří:

- Displeje, na kterých se zobrazuje nejvyšší povolená rychlost

Proměnné značení je dokonalé v tom smyslu, že obsluha může z dispečinku přesměrovávat dopravu do jednotlivých pruhů, do jednotlivých tubusů či zastavení provozu apod. Jak už jsem v úvodu zmiňoval, jedná se o dvoutubusový komplex o dvou jízdních pruzích. Každý tubus má jeden směr jízdy. Ovšem v případě potřeby má dispečink možnost přesměrovat dopravu do jednoho tubusu => doprava tubusem bude obousměrná. Tahle možnost je výhodná v tom, že v případě potřebných oprav či povinných technických kontrol systému, je možnost přesměrování dopravy tak, aby nemusela být na určitou dobu pozastavena, ba dokonce nebyla přesměrována delší objízdnou trasou. Jedná se o inteligentní technický komplex.

4.5.6. SOS výklenky

Jsou umístěny po určitých částech v každém tubusu. Slouží hlavně pro přivolání pomoci, kdy po stisku tlačítka je informován dispečink, který pomocí kamerového systému situaci zhodnotí a dále ji řeší.

4.5.7. Kamerový systém

Je umístěn na modelářských servomechanismech, kdy jsou vždy dvě serva připevněny k sobě. Jedno servo ovládá pohyb doleva a doprava, druhé servo ovládá pohyb nahoru a dolů. Pohyb serv je opět ovládán pomocí displeje na řídicí jednotce AMiNi2D a data jsou dále posílána po sériové lince mikroprocesoru, kde se data zpracují. V mikroprocesoru je potom naprogramován kód časovače s přerušením pro ovládání serv.

4.5.8. Únikové východy

Při krizové situaci umožňují přechod osob z jedné tunelové trouby do druhé. Únikové cesty jsou natlakovány vzduchem z toho důvodu, aby po otevření dveří byl tak velký tlak, aby se zplodiny nedostal do únikové cesty. Po otevření únikových dvěří je ihned upozorněn dispečink.

4.5.9. Alarm

Mimo světelnou signalizaci alarmu je zde použita sirénka, která zajišťuje zvukové upozornění dispečinku v případě:

-stisknutí tlačítka v SOS výklenku

-světelná brána -> vysoké vozidlo

-otevření dveří únikového východu

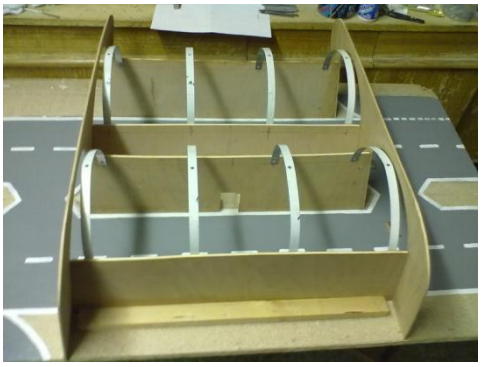
-zvýšení teploty

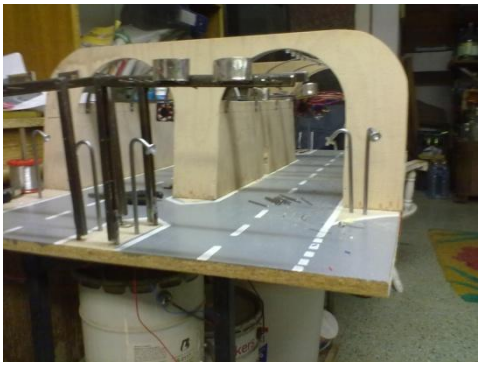
Pokud nastane některý/é z výše zmiňovaných stavů, tak je spuštěna světelná signalizace, zvuková sirénka a dále jsou všechny alarmové hlášky zobrazeny na obrazovce AMiNi2D. Pro detekci alarmů jsem použil funkční blok **ErrSig** a pro zobrazení alarmových textů je zde použit blok ALARM.

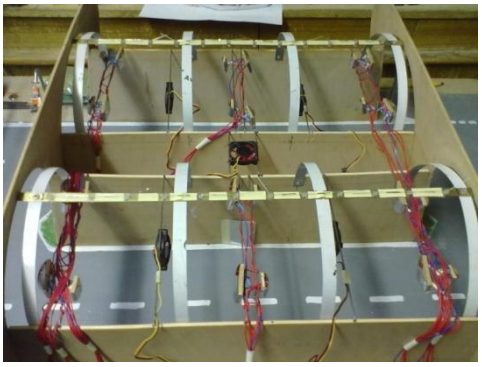
```
//jestliže je aktivován daný bit proměnné "error" tak se spustí alarm.X
//nastaví se do log.1, čímž je vyvolán alarm, který nás přepne na danou
//obrazovku a zobrazí chybové/krizové hlášky
//pomoci nastavení proměnné "kvitace" na 0xFF, tak chybu podle nastavení
//např. potlačíme a pokud se chyba do určité doby neodstranění, tak nás to
//vrátí zpět na obrazovku alarmy nebo jinou námi určenou

ErrSig error, 0x0020, kvitace, 0x0020, alarm.5,NONE.0, 0, 15, 1000, 1, 0, 0
```

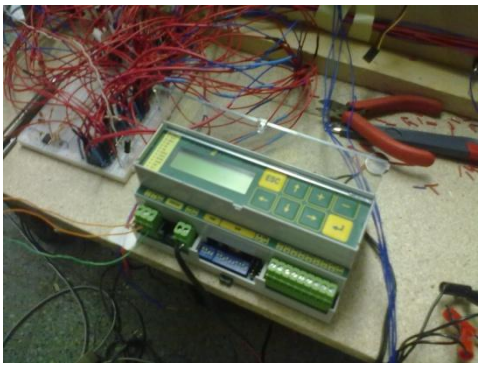
4.6. Fotografie postupu práce

Postup práce při stavbě modelu.

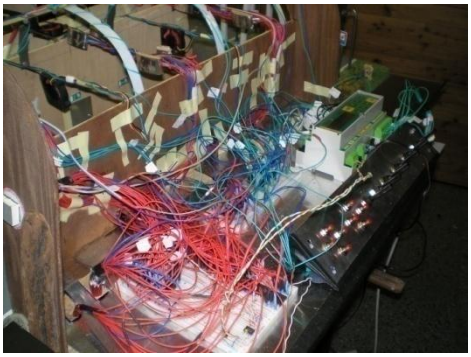








4.7. Fotografie zhotoveného modelu




- ***Fotografie jsou v příloze k dokumentaci, včetně videodokumentace***

5. Splnění cílů

Z mého pohledu se projekt vyvíjel přesně podle mých představ, i když někdy bylo potřeba něco pozměnit, tak se mi celý projekt jeví jako úspěšný.

6. Závěr

Projekt byl dokončen podle mých představ. Programová část modelu je plně funkční včetně sestaveného modelu a všechny technické problémy, které se při zhotovení práce objevily, byly úspěšně odstraněny. I když projekt není zpracován totožně s realitou, tak hlavní myšlenky a postupy, byly při práci použity. Jako hlavní řídicí jednotka byla použita řídicí jed. Podřadnou jednotkou byly mikroprocesory ATMEL, které jsem převážně využil jako rozšíření výstupů, ale při mých konzultacích jsem se s nimi naučil pracovat i hlouběji. Za pomoci mého konzultanta jsem úspěšně zprovoznil komunikace po RS485 mezi PLC a mikroprocesorem a dále už nic nebránilo cestě k dokončení projektu.

7. Použitá literatura

- Brněnské komunikace a.s.
- Technické podmínky TP 154 = > provoz, správa a údržba tunelů pozemních komunikací
- Technické podmínky TP 98 = > Technologické vybavení tunelů pozemních komunikací
- www.tunelblanka.cz
- www.dopravni-znaceni.eu
- www.amit.cz
- http://forum.amit.cz

8. Použitý software

- DetStudio - > vývojové prostředí pro řídicí systémy firmy AMiT
- AVR Studio - > vývojové prostředí firmy ATMEL
- ASIX UP - > software, který zajišťuji nahrání programu do mikroprocesoru
- Eagle - > návrhové prostředí elektronických schémat
- ProfiCad -> volně dostupný nástroj pro kreslení elektronických schémat