Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 6. 10. 2003 .....

podpis

Poděkování

Na tomto místě bych chtěl vyjádřit poděkování rodičům za jejich soustavnou péči a podporu při studiích. Dále bych chtěl poděkovat vedoucímu diplomové práce Ing. Richardu Šustovi, PhD. za mnoho podnětných připomínek při tvorbě této práce. V neposlední řadě patří můj dík Doc. Ing. Pavlu Přibylovi, CSc. za konzultaci v oblasti dopravy.

Abstrakt

Úkolem diplomové práce je navrhnout software simulující provoz na křižovatce. Křižovatka se skládá ze čtyř příjezdových komunikací pro vozidla a čtyř přechodů pro chodce. Světelné signalizační zařízení na křižovatce bude řízeno pomocí PLC (programmable logic controllers). Provoz na křižovatce je možné řídit na základě informací z příjezdových prahů a indukčních smyček. Příjezdové prahy se nacházejí na vjezdu do křižovatky ve všech směrech a indukční smyčky jsou umístěny na stání před křižovatkou. Sledováním signálů prahu a smyčky lze určit počet vozidel čekajících na průjezd křižovatkou v jednotlivých směrech a podle těchto hodnot upravit řízení.

Provozu na křižovatce se účastní též chodci, kteří se řídí světelnou signalizací a jejich příchod je indikován stiskem tlačítka.

Model bude reprezentován aplikací pro prostředí MS Windows a k propojení s PLC bude využívat kartu PCI-1750 firmy Advantech.

Abstract

The task of this diploma thesis is to design a software simulating traffic on crossroads. The crossing comprises four access roads for vehicles and four crossings for pedestrians. The traffic lights on the crossroads are controlled by PLC (programmable logic controller). The traffic control on the crossroads is based upon the information from the access-road sills and the inductive loop detectors. The sills are located on the access to the crossroads in all directions and the inductive loop detectors are located on the standings in front of the crossroads. By monitoring the signals of the sills and of the loops, it is possible to determine the number of vehicles waiting for the crossroads transit in respective directions, and based upon these values, it is possible to adjust the control.

In the crossroads traffic, pedestrians are involved as well. They are guided by the traffic lights, their arrival is indicated by pressing a button.

The model will be represented by an application for MS Windows platform and for the communication with PLC, an Advantech PCI-1750 card will be used.

Obsah

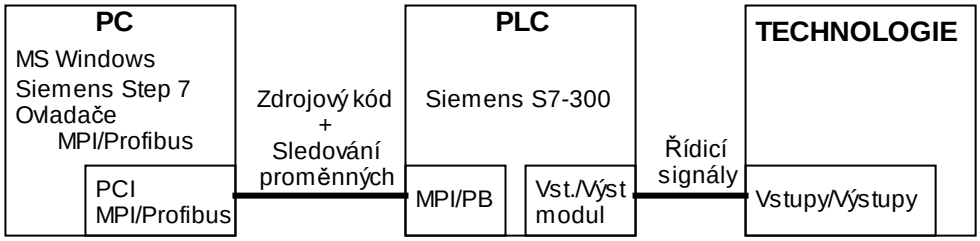
1 Úvod 5
2 Teoretický úvod 7
2.1 Karta PCI-1750 7
2.1.1 Charakteristika 7
2.1.2 Vstupy a výstupy I/O karty 8
2.1.3 Zapojení konektoru 9
2.1.4 Knihovny Advantech 10
2.2 Vlastnosti reálné křižovatky 12
2.2.1 Negativní exponenciální distribuce 13
2.2.2 Realizace detektorů v reálném dopravním uzlu 15
2.3 Požadavky na model 16
2.4 Srovnání modelu s reálným dopravním uzlem 17
3 Vlastní návrh modelu 18
3.1 Vstupy a výstupy modelu 18
3.2 Geometrie modelu 20
3.3 Realizace detektorů v modelu křižovatky 22
3.4 Datová reprezentace 23
3.5 Chování vozidel 27
3.5.1 Měření nejkratší vzdálenosti vozidel 28
3.5.2 Přednost na základě jednorozměrné souřadnice 29
3.5.3 Měření vzdálenosti od konfliktních bodů 29
3.5.4 Algoritmus chování vozidel použitý v tomto modelu 31
3.5.5 Chování chodců 36
4 Implementace 37
4.1 Inicializace 37
4.2 Vlákna 37
4.2.1 Model v komunikačním vlákně 38
4.2.2 Model v hlavním vlákně 38
4.3 Činnost hlavního vlákna 39
4.3.1 Použité komponenty 39
4.4 Činnost komunikačního vlákna 41
4.5 Generátor náhodných čísel 41
4.6 Vzhled aplikace a stručný popis funkcí 43
4.7 Požadavky na hardware 44
5 Charakteristiky modelu z hlediska řízení 46
5.1 Vlastnosti modelu vzhledem k řízení 46
6 Závěr 48
7 Použitá literatura 50
8 Použitý software 50

1 Úvod

Cílem této diplomové práce je navrhnout virtuální model křižovatky vhodný pro výuku na Katedře řídicí techniky Fakulty elektrotechnické, Českého vysokého učení technického v Praze. Virtuální model je program pro PC, který modeluje vnitřní i vnější chování reálného technologického procesu nebo reálného modelu technologického procesu. Tento model je třeba řídit stejně jako samotný technologický proces.

Pro řízení technologických procesů (např. výrobní linky, regulační soustavy apod.) se v praxi často používají programovatelné logické automaty (dále jen PLC). Uspořádání pracoviště pro řízení technologického procesu je schematicky znázorněno na Obr. 1.1.

K výuce programování PLC není třeba často tak rozsáhlých soustav, a proto se pro tyto účely užívá pouze modelů (mechanických nebo virtuálních) technologických procesů.



Obr. 1.1 Uspořádání pracoviště pro řízení technologie

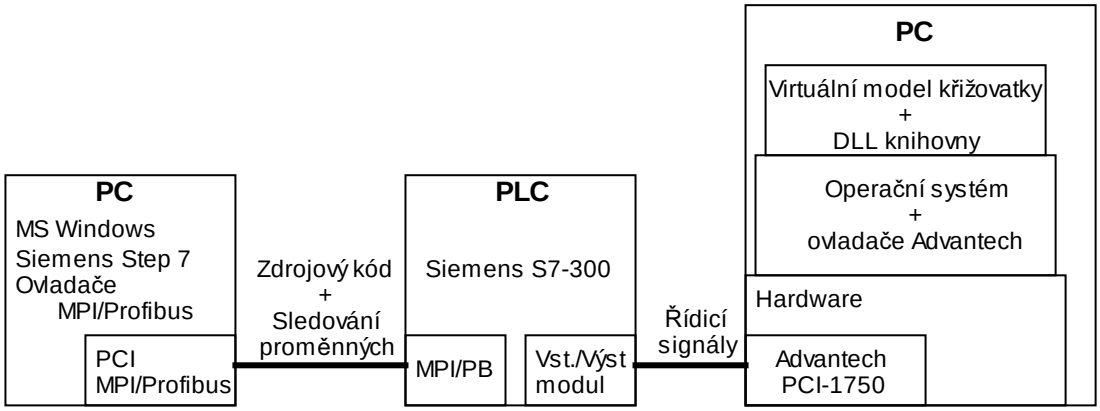
Nevýhodou mechanických modelů technologického procesu je omezená rozšiřitelnost a někdy relativně vysoké pořizovací náklady.

Dále existuje skupina úloh, které nelze reprezentovat mechanickým modelem, a to z následujících důvodů:

- Model obsluhuje nekonečný počet prvků (např. neomezený proud výrobků)
- Může dojít k chybě řízení, která by vedla k destrukci modelu
- Zpracování prvku se řídí náhodnou veličinou (příchod chodce z náhodného směru apod.)

Pro řízení virtuálního modelu technologie musíme počítač, na kterém je model spuštěn, vybavit vhodným rozhraním, které současně omezuje počet vstupů a výstupů modelu, což představuje hlavní nevýhodu virtuálních modelů.

Model křižovatky je navržen pro použití s laboratorní kartou Advantech PCI-1750, pro jejíž připojení k PLC je nutné použít převodník úrovní řídicích signálů, který podrobně popisuje [2]. Koncepce pracoviště pro řízení virtuálního modelu je naznačena na Obr. 1.2.



Obr. 1.2 Koncepce pracoviště pro řízení virtuálního modelu

Počítač s virtuálním modelem křižovatky musí být vybaven operačním systémem Windows, kartou Advantech PCI-1750 (dále jen I/O karta) a ovladači I/O karty pro Windows.

Použijeme-li pro přenos řídicích signálů kartu PCI-1750, je nutné při návrhu modelu počítat s jistým omezením. Karta disponuje 16 izolovanými vstupy a 16 izolovanými výstupy, a proto modely s komplikovanějším rozhraním potřebují vlastní komunikační protokol, který umožní po jednom kanálu přenést informace o více veličinách.

Model křižovatky je navržen v prostředí C++ Builder[1] jako aplikace pro Microsoft® Windows[2] a používá DLL knihovny firmy Advantech dodané s kartou.

Aplikace s modelem se skládá ze dvou nezávislých vláken:

- Hlavní vlákno (vlákno hlavního formuláře), které zajišťuje překreslování grafické reprezentace křižovatky a provádí aktualizační metody modelu.
- Komunikační vlákno, které čte vstupy I/O karty a zapisuje jejich hodnoty do datových struktur modelu.

Software je k diplomové práci přiložen na samostatném CD-ROM a podrobné informace o obsahu tohoto CD jsou uvedeny v příloze 1.

[1] Borland C++ Builder™ je registrovaná obchodní známka společnosti Inprise.

[2] Microsoft® Windows je registrovaná obchodní známka společnosti Microsoft.

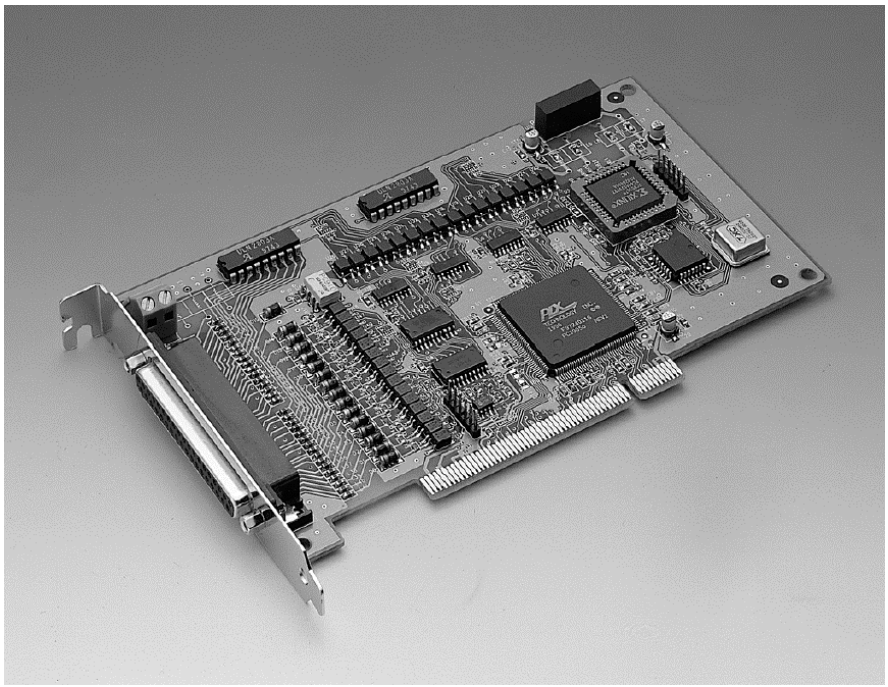
2 Teoretický úvod

Nejprve se seznámíme s použitelným rozhraním modelu, kartou PCI-1750.

Ve druhé části tohoto teoretického úvodu uvedeme základní vlastnosti reálného dopravního uzlu. Je třeba modelovat příjezdy vozidel, a proto budeme hledat distribuční funkci tohoto náhodného děje. Dále popíšeme realizaci detektorů instalovaných na příjezdech křižovatky.

2.1 Karta PCI-1750

Hodnoty výstupních veličin je třeba přenést do PLC a výstupy PLC je třeba připojit na vstupy virtuálního modelu. To zajistí karta PCI-1750 firmy Advantech, jejíž základní vlastnosti shrneme v následujících odstavcích. Podoba této karty je na Obr. 2.1.

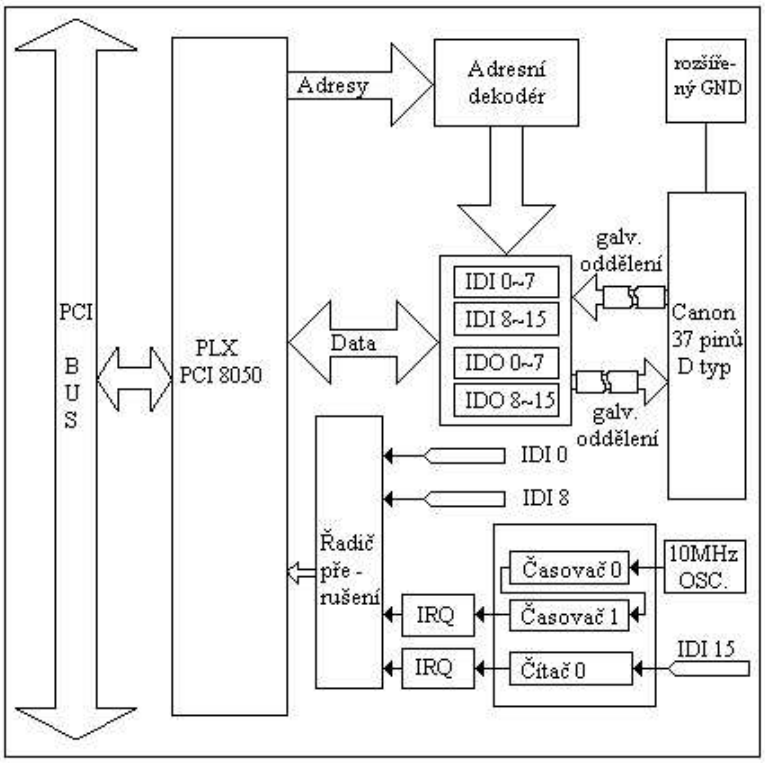


Obr. 2.1 Podoba karty Advantech PCI-1750 (dle www.advantech.com)

2.1.1 Charakteristika

Karta PCI-1750 je vybavena šestnácti izolovanými digitálními vstupy a šestnácti izolovanými digitálními výstupy. Karta dále disponuje jedním čítačem, dvěma časovači a umožňuje vyvolávat maskovatelná přerušení. To jsou však funkce, které v projektu nepoužíváme. Podrobnosti o těchto vlastnostech I/O karty jsou popsány v [4].

Pro připojení vstupů a výstupů je na kartě umístěn třiceti sedmi pólový konektor typu D (samice). Karta je vhodná pro průmyslové použití, díky optické izolaci na každém kanálu (až 2500 V stejnosměrně). I/O karta je též vybavena čítačem. Blokové schéma I/O karty je na Obr. 2.2.



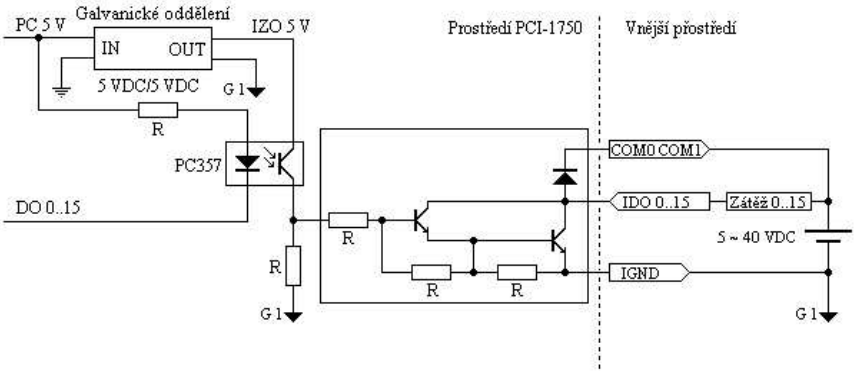
Obr. 2.2 Blokové schéma I/O karty (jak je uvedeno v [4])

2.1.2 Vstupy a výstupy I/O karty

Výstupní kanál je tvořen tranzistory v Darlingtonově zapojení pracujícím ve spínacím režimu. Externí zátěž je ve stavu logické jedna sepnuta k IGND a ve stavu logické nuly je od IGND odpojena.

Po zapnutí počítače jsou všechny výstupy I/O karty nastaveny na logickou nulu, a tak nedochází k poškození připojených zařízení po startu nebo při resetu počítače. [4]

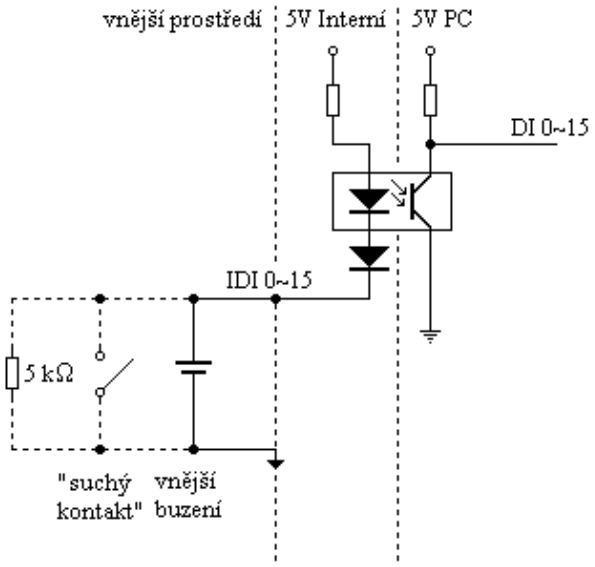
Schéma zapojení jednoho výstupního kanálu I/O karty je naznačeno na Obr. 2.3 [4].



Obr. 2.3 Připojení externí zátěže na výstupní kanál (jak je uvedeno v [4])

Při připojování externí zátěže je nutné mít na zřeteli, že proud, který protéká zemnícím pinem GND, nesmí přesáhnout 200 mA.

Vstupní kanál karty je též vybaven optickým oddělením; Obr. 2.4 ukazuje připojení vnějšího zdroje na jeden vstupní kanál I/O karty.



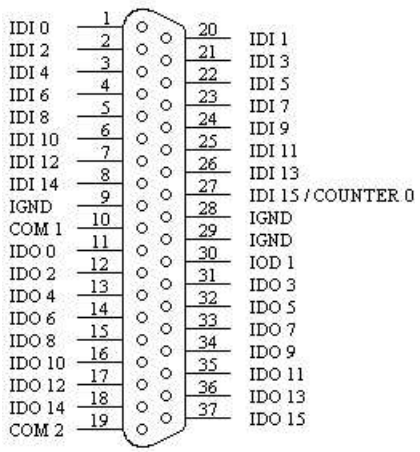
Obr. 2.4 Připojení vnějšího zdroje na jeden vstupní kanál I/O karty (jak je uvedeno v [4])

Použijeme-li tzv. „dry“ kontakt zapojení (zapojení spínače paralelně s rezistorem), pak je logická nula reprezentována spínačem v sepnutém stavu. Logická jedna odpovídá spínači rozepnutému.

V případě tzv. „wet“ zapojení (na Obr. 2.4 zdroj stejnosměrného napětí) logická nula odpovídá rozsahu napětí od 0V do 2V. Aktivní úrovni pak odpovídá napěťový rozsah 5V až 48V. **V případě, že na vstup karty připojíme záporné napětí, dojde k poškození vstupního kanálu.**

2.1.3 Zapojení konektoru

Jednotlivé kanály jsou přiřazeny na jednotlivé piny konektoru podle Obr. 2.5, význam symbolů je v Tab. 2.1. (převzato z [4])



Obr. 2.5 Jednotlivé vstupy a výstupy karty jsou připojeny konektorem DSUB-37F.

Označení	Popis
IDI0 – IDI15	Izolované logické vstupy
IDO0 – IDO15	Izolované logické výstupy
IDI15/Counter2	Izolovaný vstup čítače
IGND	Izolovaná zem
COM1, COM2	Společný pól pro připojení induktivní zátěže k výstupům IDO8 – IDO15

Tab. 2.1 Význam symbolů v Obr. 2.5

2.1.4 Knihovny Advantech

Pro odesílání výstupních veličin na vstupy PLC a příjem řídicích veličin na vstupech je nutné přistupovat k rozhraní I/O karty pomocí ovladačů Advantech. Tyto ovladače jsou dodány spolu kartou PCI a pro C++ Builder je vydán hlavičkový soubor, jehož připojení k projektu (`include`) zajistí přístup k funkcím ovladače. Funkce, které jsou v přiložených knihovnách k dispozici, jsou podrobně popsány v [3].

Ovladače podporují celou řadu produktů společnosti Advantech včetně analogových průmyslových karet PCI. Z tohoto důvodu je hlavičkový soubor poměrně rozsáhlý a při jeho použití je třeba respektovat následující vlastnost:

- Hlavičkový soubor obsahuje definice konstant (`#define`), které mají často jednoslovný, někdy dokonce jednopísmenný název. Je tedy třeba znát tyto definice a nepoužívat totožné názvy pro jména vlastních proměnných.

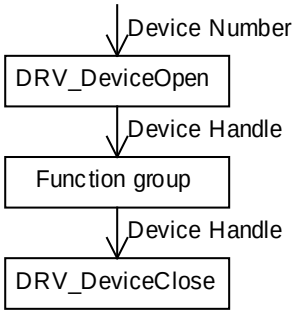
Některé obzvláště nepříjemné definice konstant ilustruje následující výpis části hlavičkového souboru[3]:

```
...
#define     C  0       //Celsius
#define     F  1       //Fahrenheit
#define     R  2       // Rankine
#define     K  3       // Kelvin
...
```

2.1.4.1 Použité funkce knihoven Advantech

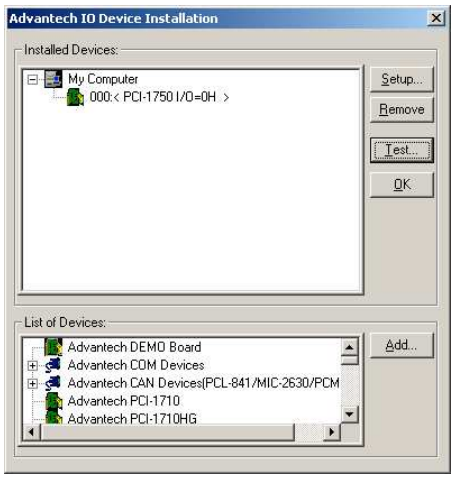
Při použití ovladače je nutné dodržet posloupnost kroků nastíněnou na Obr. 2.6.

[3] Při přechodu na Borland C++ Builder 6 je nutné vyřadit z překladu (například komentářem) řádek obsahující konstantu R, protože VCL6 (visual component library, verze 6) používá pro své účely proměnnou R, která je však díky ovladači dříve definována jako konstanta.



Obr. 2.6 Posloupnost kroků při použití ovladače Advantech PCI-1750 (dle [5])

Hodnota „Device number“ udává číslo zařízení Advantech. Toto číslo lze zjistit v nástroji pro instalaci karty (úplný popis instalace je uveden v [1] nebo [4]). V Obr. 2.7 je zařízení označeno řádkou „000:<PCI-1750 I/O=0H>“ , kde 000 odpovídá číslu zařízení.



Obr. 2.7 Zjištění čísla zařízení pomocí nástroje pro instalaci karty.

Známe-li hodnotu „Device number“ musíme pro inicializaci ovladačů volat funkci `DRV_DeviceOpen`, která vrací hodnotu „Device Handle“. Tato hodnota je typu `long` (4 bajty), proto je druhým vstupním parametrem funkce `DRV_DeviceOpen` ukazatel na proměnnou typu `long`. Následující výpis kódu demonstruje inicializaci ovladače:

```
long DevHandle;
DRV_DeviceOpen(0,&DevHandle);
```

Nyní můžeme volat jednotlivé funkce ovladače. Všechny tyto funkce vyžadují ve svých vstupních parametrech proměnnou `DevHandle`. V projektu používám následující funkce:

- **DRV_DioReadPortByte**

Použití:

```
status = DRV_DioReadPortByte(DevHandle,lpDioReadPortByte)
```

Uloží hodnotu vstupů I/O karty do struktury `lpDioReadPortByte`. Pro určení bajtu, který chceme ze vstupů načíst, nastavíme proměnnou `Port` stuktury `lpDioReadPortByte` na 0 nebo 1. Struktura `lpDioReadPortByte` je typu `PT_DioReadPortByte`, který je stručně popsán v Tab. 2.2. Do proměnné `status` se uloží informace o úspěšném provedení příkazu nebo chybové číslo.

Proměnná	Popis
Port	Udává číslo bajtu, který chceme ze vstupů načíst. Nula reprezentuje dolní byte, jednička naopak bajt horní.
Value	Hodnota vstupů. Do této proměnné se uloží hodnoty logických vstupů.

Tab. 2.2 Popis struktury PT_DioReadPortByte

- **DRV_DioWritePortByte**

Použití:

```
status = DRV_DioWritePortByte(DevHandle,lpDioWritePortByte)
```

Zapíše hodnotu v proměnné `value` (součást struktury `lpDioWritePortByte`) na výstupy I/O karty. Před voláním této funkce je třeba zapsat data do proměnné `lpDioWritePortByte` a v této struktuře opět specifikovat zapisovaný bajt. Proměnná `lpDioWritePortByte` je typu `PT_DioWritePortByte`, který je popsán v Tab. 2.3. Proměnná `status` má stejný význam jako při čtení vstupů.

- **DRV_GetErrorMessage**

Použití:

```
status = DRV_GetErrorMessage(Status,ErrorMsg)
```

Tato funkce nám umožní získat textové chybové hlášení pro chybu reprezentovanou číselnou hodnotou v proměnné `status`. Proměnná `ErrorMsg` je ukazatel na typ `String`.

Podrobný popis všech funkcí I/O karty naleznete v [5].

Proměnná	Popis
Port	Udává číslo bajtu, do kterého chceme zapisovat. Nula reprezentuje dolní byte, jednička naopak bajt horní.
Mask	Udává, které bity budou přepsány novou hodnotou a které zůstanou nezměněny.
State	Hodnota výstupů. Do této proměnné se uloží hodnoty logických výstupů.

Tab. 2.3 Popis struktury `PT_DioWritePortByte`

2.2 Vlastnosti reálné křižovatky

Pro vytvoření modelu je třeba zvolit distribuční funkci, která popisuje četnost příjezdů vozidel (a příchodů chodců) ke křižovatce.

Příjezdy vozidel k reálnému dopravnímu uzlu jsou dány denní hodinou, umístěním křižovatky v obci a dalšími vnějšími faktory. Často nelze četnost příjezdů vozidel popsat jedinou distribuční funkcí právě z těchto důvodů. V lit. [6] je podrobně diskutováno, kterou distribuční funkci použít v závislosti na poměru průměr/rozptyl.

V našem modelu křižovatky budeme pro jednoduchost uvažovat příjezdy vozidel a příchody chodců na základě jedné distribuční funkce. Budeme-li předpokládat slabý provoz, můžeme pro četnost příjezdu vozidel použít Poissonovu distribuci:

(2-1) $$P(X = x) = \frac{m^x e^{-m}}{x!}$$ nebo

(2-2) $$P(X = x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}$$, kde

P(X=x) – pravděpodobnost, že náhodná veličina X nabude hodnoty x. (x=1,2,3,..),

λ - průměrný počet výskytů náhodného jevu za jednotku času,

t – délka časového intervalu, ve kterém měříme počet výskytů náhodného jevu.

m= λt - průměrný počet výskytů náhodného jevu během periody čítání *t*.

Střední hodnota i rozptyl tohoto rozdělení udává parametr *m*: E(X)=D(X)=m.

Pro implementaci je výhodnější najít rozdělení časových mezer mezi příjezdy vozidel. Při přidávání vozidel do křižovatky totiž rozhodujeme o tom, **kdy** vozidlo přidat. Poissonova distribuce vyjadřující četnost příjezdů pak přejde v negativní exponenciální rozdělení.

Následující podkapitola, kterou cituji z [6] (kapitola 8.2.1 str. 8-199), ukáže vlastnosti negativní exponenciální distribuce a přechod z Poissonova rozdělení. Citovaný text je označen kurzívou.

2.2.1 Negativní exponenciální distribuce

Negativní exponenciální distribuce je základní distribucí pro intervaly mezi vozidly, kdy je velmi řídký provoz a není interakce mezi vozidly a přímo vyplývá z Poissonovy distribuce, rovnice (2-1). Dá se dokázat, že má-li soubor měřených dat Poissonovo rozdělení, tak doba mezi příchodem dat vyhovuje negativní exponenciální distribuci. Jestliže se neuskuteční příjezd vozidla v časovém intervalu t, bude odstup mezi průjezdem posledního vozidla a příjezdem dalšího vozidla h větší než t. S použitím rovnice (2-2) a substitucí λ=q/3600 (q – hodinová intenzita) dostáváme pro pravděpodobnost mezery rovné nebo větší t:

$$P(X = 0) = P(0) = P(h \geq t) = \frac{(\lambda t)^0 e^{-\lambda t}}{0!} = e^{-\lambda t} = e^{-\frac{qt}{3600}}$$

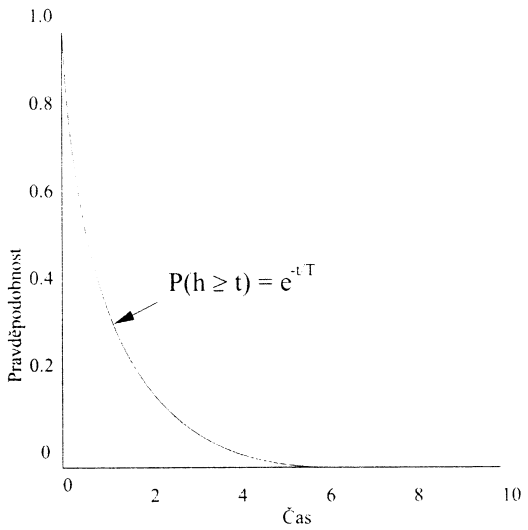
$$ale\ \lambda = \frac{q}{3600} = \frac{1}{\bar{h}}\ [\frac{voz}{s}]\ takže$$

(2-3) $$P(h \geq t) = e^{\frac{-t}{\bar{h}}}$$ *, kde*

$\bar{h}$ *- střední interval mezi vozidly [s]*

q – intenzita [voz h⁻¹]

Graf reprezentující negativní exponenciální distribuci danou vztahem (2-3) je na Obr. 2.8.

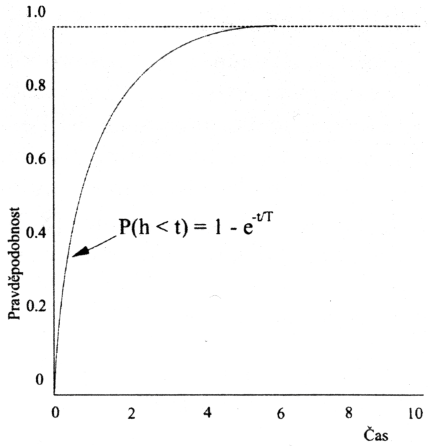


Obr. 2.8 Pravděpodobnost mezery mezi vozidly $h \geq t$ *pro* $\overline{h} = 1s$

V některých případech vyšetřujeme pravděpodobnost mezery menší než t.

(2-4) $$P(h \leq t) = 1 - e^{-\lambda t} = 1 - e^{-\frac{t}{\overline{h}}}$$

Na Obr. 2.9 je zobrazena pravděpodobnost h mezi vozidly mezery menší než t = distribuční funkce rozdělení.



Obr. 2.9 Pravděpodobnost mezery mezi vozidly $h \leq t$ *pro* $\overline{h} = 1s$

Hustota pravděpodobnosti negativní exponenciální distribuce je

(2-5) $$f(t) = \lambda e^{-\lambda t}$$

kdy pro případy využití v dopravě lze za λ *dosadit* $\frac{1}{\overline{h}}$.

Střední hodnota a rozptyl rozdělení jsou:

(2-6) $$E(X) = \overline{h} = \frac{1}{\lambda}$$

(2-7) $$D(X) = \frac{1}{\lambda^2} = \overline{h}^2 = E(X)^2$$

Jediný parametr této distribuce může být určen buď z diskrétní distribuce (λ) nebo ze střední hodnoty pozorovaných mezer ($\bar{h}$).

2.2.2 Realizace detektorů v reálném dopravním uzlu

Tato kapitola popisuje princip a použití indukčních smyček v reálných dopravních uzlech.

Vzhledem k tomu, že na katedře řídicí techniky nebylo v odborné knihovně k dispozici dílo, na které bychom mohli v této souvislosti navázat, cituji kapitolu 5.1. z lit. [6]. Citovaný text je opět zvýrazněn kurzívou.

Indukční smyčky jsou nejvíce používané dopravní detektory, zvláště ve spojení s dopravními řadiči. Smyčky spolupracují s elektronickou částí, která je dnes již výlučně na bázi procesorů. To umožní implementovat řadu funkcí zvyšujících spolehlivost detekce (automatické přelaďování kmitočtů, nastavování úrovně apod.). Uspořádání s jednou smyčkou se používá jako přítomnostní a výzvový detektor nebo detektor obsazenosti. Z hodnoty obsazenosti lze usuzovat na tvorbu kolon. Pro měření rychlosti se obvykle používají smyčky dvě. Rychlost se počítá z časového odstupu náběžných hran obou smyček, při jejich konstantní vzdálenosti. Stejně tak lze pomocí dvou smyček kategorizovat vozidla na osobní a nákladní.

Hlavní výhodou indukčních smyček je jejich snadná aplikovatelnost a spolehlivá funkce. Také cena smyčky je ve srovnání s jinými způsoby detekce zatím relativně nízká. Mezi nevýhody patří to, že pomocí jedné smyčky lze spolehlivě vyhodnotit intenzitu a s poměrně velkou nepřesností rychlost. Pokud je nutné měřit rychlost přesněji je třeba použít smyčky dvě. K podstatným nevýhodám patří, že při instalaci smyček dochází k narušení vozovky drážkou cca 10 x 30 mm (š x h). Praktické zkušenosti navíc ukazují, že údržba smyček je poměrně nákladná, neboť v prostoru křižovatek, kde jsou obvykle umístěny (před stopčárou), dochází díky brždění těžších vozidel k „hrnutí“ asfaltu, což mívá za následek přerušení smyčky[4].

Pro zvýšení mechanické odolnosti smyčky se nyní experimentuje s jedno závitovými smyčkami tvořenými vodičem s relativně velkým průřezem (10 mm²). Tyto smyčky však vyžadují díky malé indukčnosti impedanční přizpůsobení transformátorem umístěným v jejich bezprostřední blízkosti. Dosavadní zkušenosti jsou dobré a jedno závitové smyčky lze doporučit pro praktické aplikace.

[4] Praktické zkušenosti z Prahy říkají, že je v dlouhodobém průměru až 20 % smyček v poruše.

2.2.2.1 Princip činnosti

Detekce vozidel na bázi indukčních smyček je tvořena vlastní smyčkou, která je vlastně vzduchovou cívkou s indukčnosti popsanou dle vztahu

$$L = \mu_0 \mu_r \frac{N^2 S}{l},$$

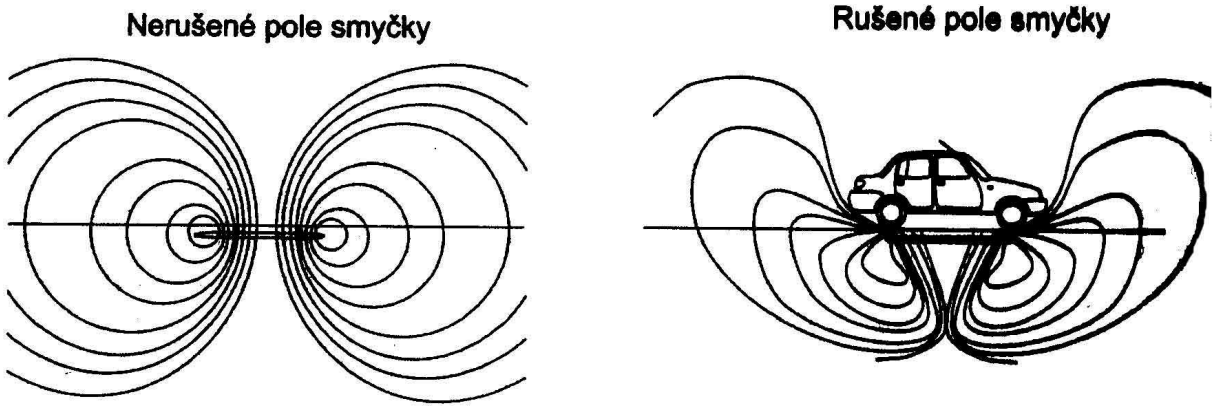
kde $\mu_0 = 4\pi 10^{-7}$

$\mu_r = 1$ *(vzduch)*

S= plocha cívky

L = výška cívky

Cívka je napájena prostřednictvím přizpůsobovacího členu z oscilátoru 20-150kHz. Tím je okolo závitů vytvořeno homogenní magnetické pole, které je narušeno přítomností kovové karoserie vozidla, Obr. 2.10 a vyvolaná změna je vyhodnocována v detektoru. Změna neboli rozladění je z fyzikálního hlediska vyvoláno vířivými proudy ve vodivých částech vozidla, které způsobí vzrůst ztrátového odporu cívky a zhoršení její kvality (poměr induktivního a ztrátového odporu).



Obr. 2.10 Princip detekce vozidel indukční smyčkou

Detektory vyhodnocují jednu nebo více změn vyvolaných přítomností vozidla:

- *Změnu amplitudy (přítomnost vozidla – menší amplituda)*
- *Posuv fáze díky přítomnosti vozidla*
- *Změnu kmitočtu (přítomnost vozidla – vyšší kmitočet)*

Výše uvedené změny mohou být poměrně malé, například při detekci cyklistů, a proto je nutné věnovat pozornost i ztrátám na přívodním kabelu. Indukční smyčky lze použít až do vzdálenosti 200-300m od detektoru. Je nutné si však uvědomit, že indukčnost přívodního kabelu je v sérii s indukčností cívky a zvyšuje tím nároky na citlivost detektorů.

2.3 Požadavky na model

Po dohodě s vedoucím diplomové práce vznikl souhrn následujících požadavků na model křižovatky:

- Křižovatka bude mít čtyři příjezdové komunikace.

- Každá příjezdová komunikace bude řízena semaforem pro vozidla.
- Na příjezdové komunikaci bude umístěn práh indikující příjezd vozidla.
- Před každým semaforem pro vozidla se bude nacházet indukční smyčka, indikující přítomnost vozidla na hranici křižovatky.
- Křižovatka bude obsahovat čtyři přechody pro chodce.
- Každý přechod bude řízen semaforem pro chodce.
- Pro každý přechod budou instalována dvě tlačítka pro chodce, indikující příchod chodce.
- Model bude generovat náhodně přijíždějící vozidla a náhodně přicházející chodce.
- Bude možné ruční generování vozidel a chodců.
- Model bude detekovat srážky vozidel zaviněné chybným řízením.
- Model bude detekovat srážky vozidel s chodci zaviněné chybným řízením.
- Model bude obsahovat možnost dohlídání červeného signálu.

2.4 Srovnání modelu s reálným dopravním uzlem

Vlastnosti modelu křižovatky jsou omezeny použitým hardwarem a požadavkem na jednoduchou implementaci (jedná se o školní práci). Souhrn těchto omezení je uveden v následujících odstavcích.

- Dopravní řadiče reálné křižovatky musí zajistit tzv. dohlídání signálu „stůj“ a signálu „volno“. Způsob jakým je toto dohlídání realizováno je popsán v [6]. U signálu „stůj“ je třeba zjistit, zda žárovkou reprezentující červenou opravdu protéká proud, pokud ji řadič svými výstupními obvody sepnul. U signálu „volno“ sledujeme naopak přítomnost napětí v době, kdy řadič zelený signál nesepnul. V těchto případech musí řadič okamžitě přejít do stavu blikavé žluté nebo musí signální skupiny úplně odpojit. Naproti tomu má laboratorní model k dispozici pouze 16 výstupů (díky použité I/O kartě), a proto umožňuje dohlídání pouze červeného světla.
- Z požadavku na jednoduchou implementaci není možné simulovat reálné chování řidičů. Model pro popis pohybu vozidla využívá předem definovaných trajektorií a dále pro jednoduchost uvažuje pouze dva režimy:
    a) vozidlo stojí
    b) vozidlo se pohybuje svojí maximální rychlostí
- V reálném dopravním uzlu se chování přijíždějícího vozidla obvykle mění v závislosti na stavu návěstidla. V případě, že řidič vidí na příjezdu zelené návěstidlo, zvyšuje rychlost, avšak při červeném návěstidle často brzdí předčasně. (viz [6] str. 2-51). Modelování rychlosti vozidla na základě výše zmíněného principu tuto skutečnost nerespektuje.
- Jak bude ukázáno dále, příjezdy vozidel k dopravnímu uzlu obecně nelze popsat jednou distribuční funkcí. Model využívá pro generování vozidel pouze jednu distribuci.

3 Vlastní návrh modelu

Vnější chování modelu je jednoznačně popsáno stavy jednotlivých vstupů a výstupů. Obsah datových struktur křižovatky a algoritmy, které provádí aktualizace těchto datových struktur popisuje vnitřní chování modelu. Postup, jakým byla reprezentace vnitřního chování navržena a význam jednotlivých vstupů a výstupů, bude podrobně rozebrán v této kapitole.

3.1 Vstupy a výstupy modelu

Vstupy modelu, jejichž význam je shrnut v Tab. 3.1, mají vliv na chování vozidel a chodců v křižovatce. Vyjadřují totiž, zda se jedná o řízenou, či neřízenou křižovatku, a dále popisují stavy semaforů pro vozidla a chodce v jednotlivých směrech

Vstup	Popis
0	1. bit stavu semaforu vozidel – jižní směr
1	2. bit stavu semaforu vozidel – jižní směr
2	Stav semaforu chodců – jižní směr
3	1. bit stavu semaforu vozidel – západní směr
4	2. bit stavu semaforu vozidel – západní směr
5	Stav semaforu chodců – západní směr
6	1. bit stavu semaforu vozidel – severní směr
7	2. bit stavu semaforu vozidel – severní směr
8	Stav semaforu chodců – severní směr
9	1. bit stavu semaforu vozidel – východní směr
10	2. bit stavu semaforu vozidel – východní směr
11	Stav semaforu chodců – východní směr
12	Nepřiřazeno
13	Ovládání oranžové v neřízené křižovatce
14	1. bit stavu křižovatky
15	2. bit stavu křižovatky

Tab. 3.1 Význam jednotlivých vstupů modelu

První bit	Druhý bit	Stav semaforu
1	0	Červená
0	0	Zelená
0	1	Oranžová
1	1	Oranžova+červená

Tab. 3.2 Přiřazení stavů semaforu vozidel hodnotám na vstupech

Stav semaforu vozidel je jednoznačně popsán hodnotou na dvou vstupech (první a druhý bit stavu viz Tab. 3.1). Toto přiřazení hodnot stavům je v uvedeno Tab. 3.2.

Stav semaforu pro chodce je dán jedním vstupem, kdy logická jedna odpovídá stavu „stůj“ a logická nula odpovídá stavu „volno“, jak je uvedeno v Tab. 3.3.

Hodnota bitu	Stav semaforu chodců
0	Červená
1	Zelená

Tab. 3.3 Přiřazení stavů semaforu chodců hodnotám na vstupu.

Tab. 3.2 ani Tab. 3.3 neobsahuje stav, ve kterém je semafor vypnut. Kdybychom tento stav chtěli přidat, museli bychom použít již tři vstupy pro popis jediného semaforu vozidel (a dva vstupy pro popis jednoho semaforu chodců), což by vyčerpalo omezený počet vstupů. Proto má model další dva vstupy, které udávají, zda je křižovatka řízena či nikoliv.

Proč používat dva vstupy pro rozhodování mezi dvěma stavy? Když totiž k I/O kartě není nic připojeno, je na všech vstupech logická 1. Připojíme-li k I/O kartě převodník logických úrovní, změní se logická hodnota na všech vstupech na logickou nulu. Přestože v obou případech ještě nedošlo k řízení křižovatky (algoritmem uloženým v PLC), mění se (při použití jednoho vstupu) stav křižovatky. Význam těchto vstupů shrnuje Tab. 3.4.

První bit	Druhý bit	Stav křižovatky
0	0	Neřízená - světelné signalizační zařízení vypnuto
0	1	Neřízená – režim blikavé oranžové
1	0	Řízená - světelné signalizační zařízení v provozu
1	1	Neřízená - světelné signalizační zařízení vypnuto

Tab. 3.4 Význam vstupů 14 a 15

V případě rovnosti obou bitů jsou všechny semafory na křižovatce vypnuty, neboť nelze určit, zda je model řízen PLC algoritmem. Zbývají dva případy, kdy jsou hodnoty bitů různé. V jednom z těchto případů uvažujeme řízenou křižovatku, a ve druhém předpokládáme režim blikavé oranžové. V tomto režimu očekáváme na vstupu č. 13 periodický signál, který ovládá oranžové světlo. Hodnoty tohoto vstupu jsou ke stavu oranžového světla přiřazeny na základě Tab. 3.5.

Hodnota vstupu 13	Význam
0	Oranžové světlo nesvítí
1	Oranžové světlo svítí

Tab. 3.5 Význam vstupu 13

Výstupy modelu poskytují informaci o provozu na příjezdových komunikacích (indikují přejezd vozidla přes práh a přes indukční smyčku) a dále informují o výskytu chodců čekajících na uvolnění přechodu. Čtyři výstupy slouží k simulaci detektoru proudu procházejícího červenou žárovkou. Tyto detektory proudu se použije řadič pro dohlídání signálu „stůj“. Konkrétní význam jednotlivých výstupů popisuje Tab. 3.6.

Výstupy generuje model v jednotlivých cyklech aktualizační metody a není tedy zaručena stejná šířka pulsu. Z pohledu PLC není však tato vlastnost rozhodující. Při čtení vstupů v PLC se často využívá detekce náběžné hrany signálu, nikoliv jeho úrovně. Je však třeba respektovat tuto vlastnost modelu a při vyhodnocování výstupů modelu v PLC používat detekci náběžné hrany.

Výstup	Popis
0	Práh indikující příjezd vozidla z jižního směru
1	Indukční smyčka na jižním vjezdu do křižovatky
2	Tlačítko chodců pro jižní přechod
3	Práh indikující příjezd vozidla ze západního směru
4	Indukční smyčka na západním vjezdu do křižovatky
5	Tlačítko chodců pro západní přechod
6	Práh indikující příjezd vozidla ze severního směru
7	Indukční smyčka na severním vjezdu do křižovatky
8	Tlačítko chodců pro severní přechod
9	Práh indikující příjezd vozidla z východního směru
10	Indukční smyčka na východním vjezdu do křižovatky
11	Tlačítko chodců pro východní přechod
12	Dohlídání červené v jižním směru
13	Dohlídání červené v západním směru
14	Dohlídání červené v severním směru
15	Dohlídání červené ve východním směru

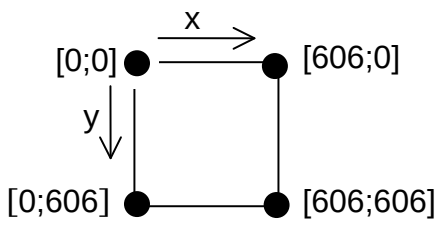
Tab. 3.6 Význam jednotlivých výstupů modelu

3.2 Geometrie modelu

Viditelná část modelu křižovatky je navržena pro čtvercovou oblast s délkou strany 606 obrazových bodů. Počátek souřadné soustavy kreslící oblasti se nachází v levém horním rohu čtverce. Na Obr. 3.1 je nakreslen rozsah hodnot souřadnic pro viditelnou oblast a umístění počátku souřadné soustavy. Jednotkou soustavy souřadnic je obrazový bod. Souřadný sytém a jednotky jsou zachovány i mimo viditelnou oblast.

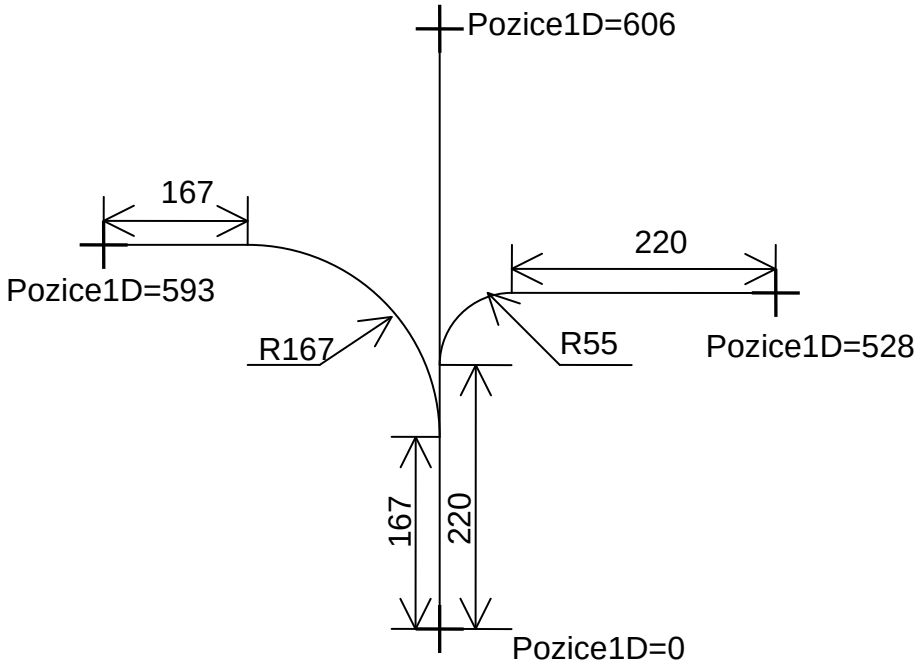
Pro každou příjezdovou komunikaci (dále jen směr) jsou definovány tři trajektorie, po kterých se mohou vozidla pohybovat. První trajektorie popisuje pohyb vozidla odbočujícího vlevo, druhá představuje přímý průjezd křižovatkou a třetí odpovídá odbočení vpravo. Každá z těchto trajektorií je pak ještě rozdělena na tři části:

- První část je popsána lineární funkcí (úsečkou) a definuje množinu bodů po kterých bude vozidlo přijíždět k hranici křižovatky.
- Druhá část trajektorie je, v případě že vozidlo odbočuje, popsána rovnicí kružnice. Projíždí-li vozidlo křižovatkou bez odbočení, pak je i tato část popsána úsečkou.
- Třetí část je popsána opět úsečkou umístěnou za křižovatkou a její úhel je dán cílovým směrem vozidla.



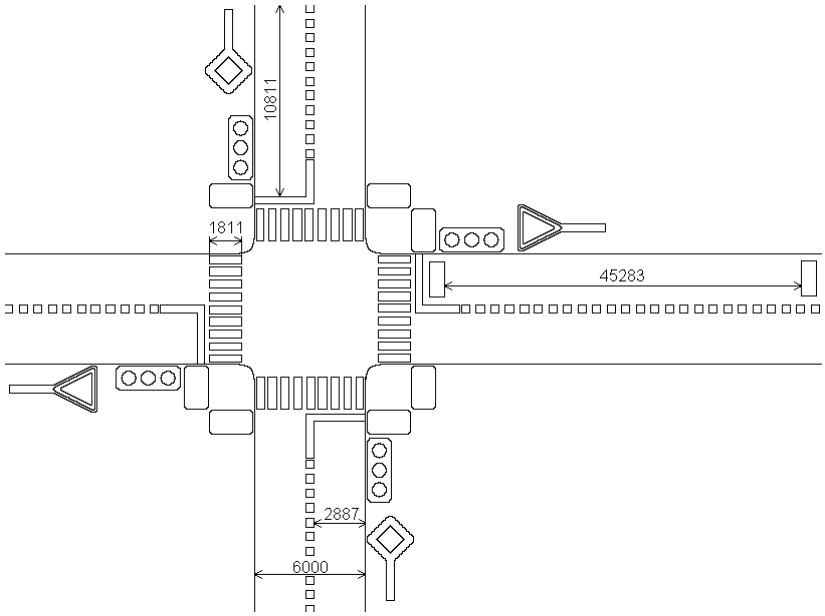
Obr. 3.1 Souřadná soustava kreslicího plátna

V rámci jedné trajektorie jsme pro jednoduchost zvolili další (jednorozměrnou) souřadnici (Pozice1D), která udává jakou vzdálenost vozidlo urazilo od počátku trajektorie. Parametry všech tří trajektorií jižního směru a volba jednorozměrné souřadnice (dále 1D) jsou na Obr. 3.2 .



Obr. 3.2 Trajektorie v jednom směru

Pro srovnání parametrů modelu křižovatky s parametry reálného dopravního uzlu, je žádoucí najít přibližný vztah mezi jednotkami modelu (obrazový bod) a reálného světa (metr). Šířka jízdního pruhu vozidel v modelu činí 53 obrazových bodů. Je-li tato šířka ve skutečnosti 3 metry, můžeme říci, že 1 metr odpovídá přibližně 17,6 obrazových bodů. Na Obr. 3.3 je znázorněna křižovatka kótovaná v milimetrech, přičemž rozměry zřejmé z Obr. 3.2 nejsou zakresleny. Na obrázku je zakreslena vzdálenost mezi detektory na příjezdu z východního směru. Tato vzdálenost je na všech příjezdových směrech stejná, jakož i šířka vozovky a jízdního pruhu.



Obr. 3.3 Znázornění křižovatky (kóty v mm)

Pro snadnější orientaci označíme jednotlivé příjezdové komunikace podle světových stran ve stejné konvenci jako na zeměpisných mapách.

3.3 Realizace detektorů v modelu křižovatky

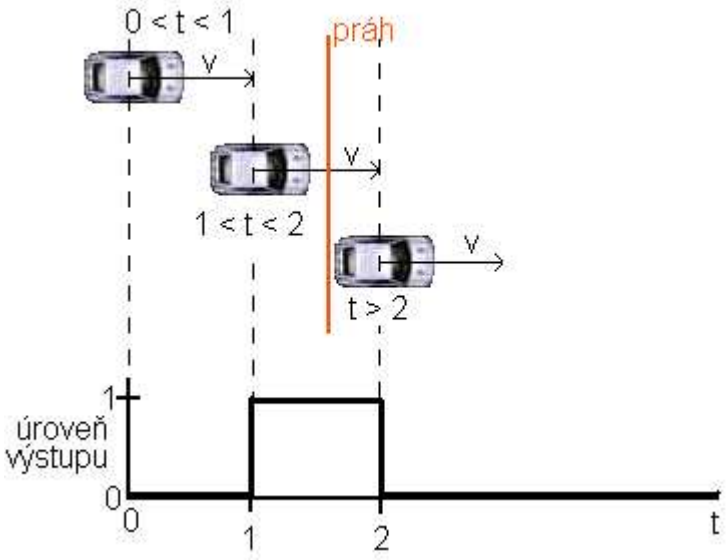
V této kapitole se zabývám realizací indukční smyčky (resp. prahu) v modelu křižovatky. Umisťujeme dva detektory na každém příjezdu: Jeden před křižovatkou a jeden na stopčáře křižovatky. Pomocí těchto detektorů je možné zjistit počet vozidel čekajících na vjezd do křižovatky v každém směru. Při řízení modelu je pak možné upravit časové konstanty signálu „volno“ tak, aby vyklizení křižovatky proběhlo rychleji.

Pozice příjezdového prahu je v reálné křižovatce odvozena od maximální povolené příjezdové rychlosti vozidel. Budeme-li přepokládat, že vozidla přijíždí ke křižovatce (průměrnou) rychlostí 60 km/h, pak by měl být práh umístěn 35 až 50 metrů před stopčárou křižovatky ([7] str. 26). Tato vzdálenost odpovídá (dle kapitoly 3.2) přibližně 618 až 883 obrazových bodů.

Vozidlo se v modelu pohybuje tak, že v každém aktualizačním kroku navýšíme hodnotu jeho polohy (1D) o hodnotu jeho aktuální rychlosti. Práh je možné reprezentovat jedinou hodnotou (1D) polohy. Detekci průjezdu vozidla přes práh provádíme porovnáním aktuální a následné polohy vozidla s hodnotou prahu. Je-li totiž poloha v současném aktualizačním kroku menší než hodnota prahu a zároveň hodnota v následném kroku větší než hodnota prahu, dochází k přejezdu vozidla přes práh. Tento přejezd trvá přesně jeden aktualizační krok (v Obr. 3.4 odpovídá $1<t<2$) a pouze po tuto dobu nastavím odpovídající výstup na aktivní úroveň. Ve všech ostatních případech je na výstupu logická 0.

Indukční smyčka je v modelu realizována stejně a princip určení přejezdu je stejný jako při přejezdu prahu. Umístění indukční smyčky na stopčáře křižovatky nám

zajistí aktivní hodnotu výstupu po dobu čekání vozidel na červenou. Je to dáno tím, že vozidla při signálu „stůj“ čekají těsně před hranicí smyčky.



Obr. 3.4 Přejezd prahu (či indukční smyčky) vyvolá v jednom aktualizační kroku nastavení log. 1 na příslušném výstupu.

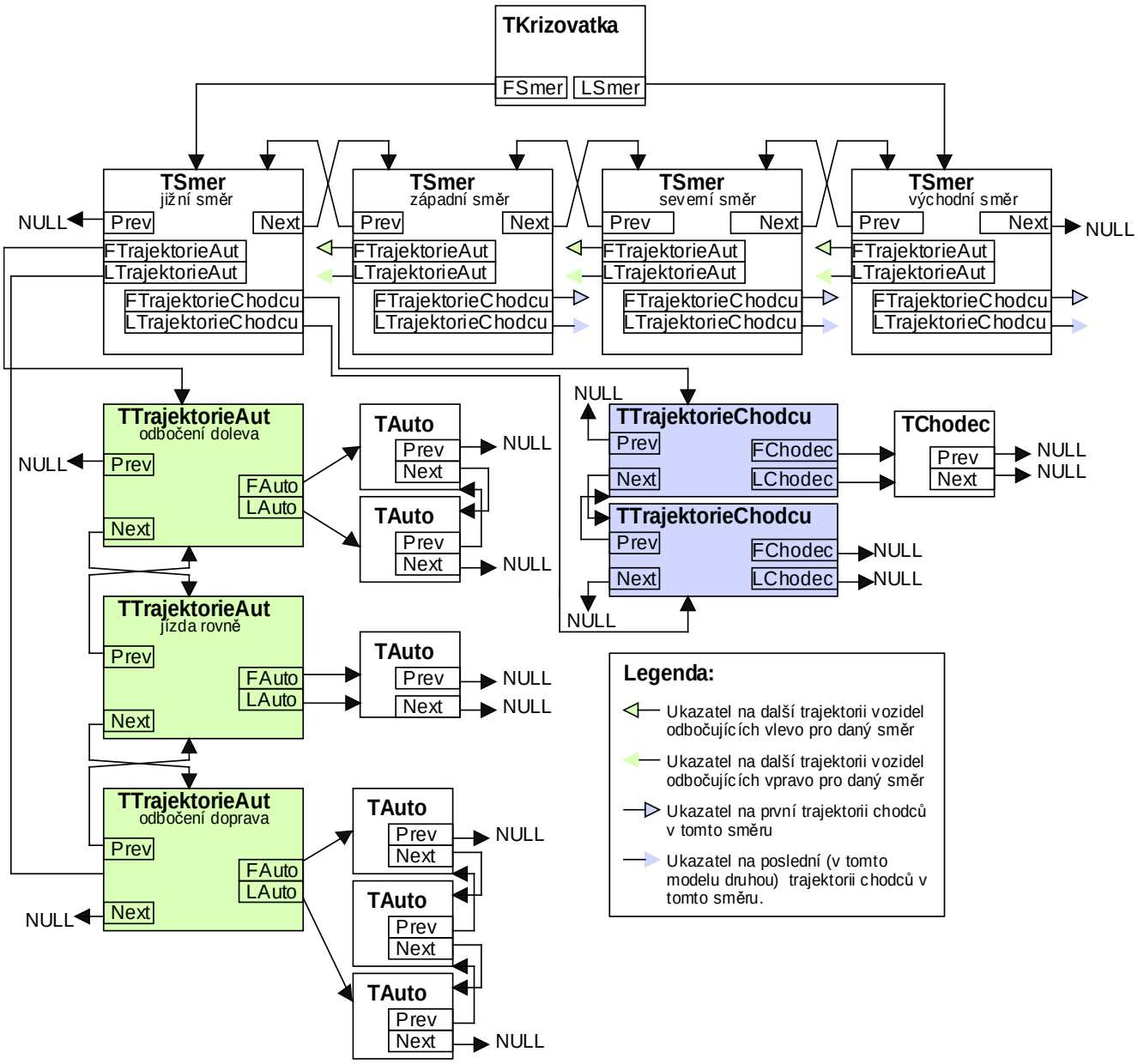
3.4 Datová reprezentace

Stav modelu křižovatky je v každém časovém okamžiku přesně popsán datovou strukturou uloženou v operační paměti počítače. Hierarchie a význam jednotlivých částí datové struktury popisuje tato kapitola.

Datové struktury jsou navrženy tak, aby bylo možné počet směrů a trajektorií při návrhu podobného modelu změnit. V každé úrovni jsou prvky spojeny obousměrně řetězeným seznamem, kde první prvek seznamu má nulový ukazatel na svého předchůdce a poslední prvek má nulový ukazatel na svého následníka.

Jednotlivé úrovně jsou propojeny způsobem ukázaným na Obr. 3.5. Všimněme si, že každá hierarchicky vyšší datová struktura obsahuje ukazatele na začátek a konec spojového seznamu nižších struktur. Tato koncepce je zachována až k významově nejnižší struktuře `TAuto` resp. `TChodec`. Z Obr. 3.5 je též patrné, že v této chvíli přijíždí celkem 6 vozidel z jižního směru (Dvě vozidla odbočují vlevo, jedno vozidlo projíždí křižovatku přímo a dvě odbočují vpravo.) a přichází chodec po první jižní trajektorii.

V dalších odstavcích popíšeme informace, které jsou v těchto propojených datových strukturách ukládány.



Obr. 3.5 Propojení datových struktur modelu křižovatky

Hierarchicky nejvyšší datovou strukturou je `TKrizovatka`. Tato struktura obsahuje souřadnice všech konfliktních bodů v křižovatce a ukazatele na první a poslední směr (`TSmer`) křižovatky. Tento model je reprezentován pouze čtyřmi směry, datová struktura však umožňuje dynamické přidávání a odebírání směrů. Krátký popis proměnných ve struktuře `TKrizovatka` shrnuje Tab. 3.7.

Proměnná	Význam
KoN[12][12]	Tabulka konfliktů vozidel (12 trajektorií vozidel koliduje též s 12 trajektoriemi)
KoC[8][12]	Tabulka konfliktů chodců s vozidly (8 trajektorií chodců koliduje s 12 trajektoriemi vozidel)
FSmer	Ukazatel na první strukturu Tsmer
LSmer	Ukazatel na poslední (čtvrtou) strukturu TSmer

Tab. 3.7 Význam proměnných TKrizovatka

V každé ze struktur `TSmer` uchováváme informaci o stavu semaforů vozidel a chodců v daném směru. Dále pro každý směr uchováváme ukazatele na obrázky jednotlivých semaforů, pozice, na které se semafory budou vykreslovat, pozice detektorů a nakonec opět dva ukazatele na každou z podřízených struktur: `TTrajektorieAut`, `TTrajektorieChodcu`. Význam důležitých proměnných této datové struktury je přiblížen v Tab. 3.8.

Proměnná	Význam
StavSemaforuAut	Celé číslo vyjadřující stav semaforu vozidel pro daný směr.
StavSemaforuChodcu	Celé číslo vyjadřující stav semaforu chodů pro daný směr.
PoziceSemaforuAut	Souřadnice semaforu vozidel, na které se bude bitmapa semaforu vykreslovat.
PoziceSemaforuChodcu0 PoziceSemaforuChodcu1	Souřadnice semaforů pro chodce.
BmpSemaforAut[5]	Pět ukazatelů na pět obrázků reprezentujících jednotlivé stavy semaforu vozidel.
BmpSemaforuChodcu0[3] BmpSemaforuChodcu1[3]	Tři ukazatele na obrázky semaforu chodců. Tyto obrázky odpovídají jednotlivým stavům semaforu. V jednom směru máme dva semafory, a proto jsou zapotřebí dva vektory ukazatelů.
PoziceSmycky	Jednorozměrná souřadnice indukční smyčky daného směru.
PozicePrahu	Umístění prahu vyjádřené jednorozměrnou souřadnicí.

Tab. 3.8 Popis struktury TSmer

Trajektorie chodců je v modelu křižovatky popsána počátečním bodem (dvojrozměrná souřadnice), úhlem natočení a délkou úsečky. Tyto hodnoty jsou uloženy v proměnných dle Tab. 3.9.

Proměnná	Význam
Interval	Přípustný rozsah 1D souřadnice
Start	2D souřadnice udávající počátek trajektorie
Fi0	Úhel trajektorie

Tab. 3.9 Reprezentace trajektorie chodců

Trajektorie vozidel se skládá ze dvou úseček a části kružnice. K jednoznačnému určení trajektorie je zapotřebí uchovávat intervaly jednorozměrné souřadnice pro jednotlivé úseky, začátky těchto úseků (ve dvojrozměrné souřadnici), poloměr a střed kružnice (v případě odbočující trajektorie). Natočení trajektorie je reprezentováno počátečním a koncovým úhlem přímých úseků. Proměnné struktury `TTrajektorieAut` jsou popsány v Tab. 3.10.

Proměnná	Význam
Start0, Start1, Start2	Začátky jednotlivých částí trajektorie ve 2D souřadnicích.
Interval0 až Interval2	Intervaly jednotlivých částí trajektorie v 1D souřadnicích.
Polomer	Poloměr kružnice v případě odbočujících trajektorií.
Stred	Střed kružnice v případě odbočujících trajektorií.
Fi0, Fi1	Počáteční a koncový úhel trajektorie.

Tab. 3.10 Význam proměnných popisující trajektorii vozidel

Datová reprezentace vozidla (viz Tab. 3.11) je vyjádřena již jen rychlostí, polohou, úhlem natočení a doplňujícími údaji. Rychlost vozidla vyjadřuje počet obrazových bodů, které vozidlo urazí za jeden aktualizační krok. Údaj o rychlosti je rozdělen do dvou proměnných.

Proměnná *`MaxRychlost`* reprezentuje rychlost, kterou dostane vozidlo bezprostředně po přidání do modelu. V proměnné *`Rychlost`* se ukládá aktuální rychlost vozidla a při zastavení vozidla (např. na signál „stůj“) je hodnota této proměnné nula. V případě, že důvody které vedly k zastavení vozidla již pominuly, dojde k obnovení aktuální rychlosti z proměnné *`MaxRychlost`* (`Rychlost=MaxRychlost`) a vozidlo se dá opět do pohybu.

Pro úplný popis polohy a stavu chodce postačí uchovat informaci o rychlosti, poloze v rámci trajektorie a o úhlu natočení obrázku chodce. Dále struktura `Tchodec` obsahuje doplňující informace o účasti v kolizi a viditelnosti. Reprezentace rychlosti chodce je však odlišná od vozidel. Proměnná `Rychlost` zde vyjadřuje aktuální rychlost chodce. Maximální rychlost chodce je dána globálním nastavením programu, neboť všichni chodci se v modelu pohybují stejnou rychlostí. V Tab. 3.12 jsou uvedeny názvy proměnných a jejich význam.

Proměnná	Význam
Pozice1D	Vyjadřuje pozici vozidla v rámci nadřazené trajektorie.
Uhel	Z analytického vyjádření trajektorie se vypočítá úhel vozidla v místě, kde se vozidlo nachází a uloží se do proměnné Uhel. Při vykreslování se obrázek vozidla natočí na tento údaj.
Rychlost	Aktuální rychlost vozidla.
MaxRychlost	Rychlost vozidla při generování. Slouží též pro obnovu rychlosti po zastavení.
TypAuta	Index obrázku vozidla, který vyjadřuje podobu.
Viditelne	Logická proměnná vyjadřující výskyt vozidla ve viditelné části křižovatky. V případě log. 0 se bitmapa vozidla nekreslí.
Kolize	V případě střetu vozidla s jiným, nastavíme tuto logickou proměnnou na TRUE. Je pak možné vykreslit značku označující účast v konfliktu.

Tab. 3.11 Popis proměnných ve struktuře TAuto

Proměnná	Význam
Pozice1D	Vyjadřuje pozici chodce v rámci jeho trajektorie.
Uhel	Při vykreslování se obrázek chodce natočí na tento údaj.
Rychlost	Aktuální rychlost chodce. Nabývá hodnoty 0 nebo rychlosti stanovené pro všechny chodce v modelu.
TypChodce	Index obrázku chodce, který vyjadřuje podobu. (hodnoty 0 až 1)
Viditelne	Logická proměnná, která vyjadřuje zda se chodec nachází ve viditelné části trajektorie. V opačném případě by se chodec nacházel v oblasti semaforu pro chodce.
Kolize	V případě střetu vozidla s chodcem, nastavím tuto logickou proměnnou na TRUE. Pak dojde ke změně obrázku a odstranění chodce z modelu.

Tab. 3.12 Popis proměnných reprezentujících chodce v modelu

3.5 Chování vozidel

Vnitřní chování modelu je dáno pohybem vozidel a chodců a stavy jednotlivých semaforů. V době návrhu modelu bylo třeba najít jednoduchý algoritmus, který zajistí správné chování vozidel, je-li křižovatka neřízena. Tento algoritmus se stal základem pro chování vozidel v řízené křižovatce a usnadňuje detekci kolizí vozidel.

V následujících odstavcích popíšeme algoritmy, které jsme vyzkoušeli v době návrhu, ale jejich použití nepostačuje nebo je nevhodné. Závěrem této kapitoly podrobně popíšeme algoritmus, který zajišťuje chování vozidel v tomto modelu křižovatky.

3.5.1 Měření nejkratší vzdálenosti vozidel

Algoritmus hledá dvojici vozidel [V,V1] , která má nejmenší vzájemnou vzdálenost. Vzdálenost počítáme z dvourozměrných souřadnic pomocí (2-8). V případě, že je tato vzdálenost menší než stanovená mez, zastavíme jedno z vozidel tak, aby se v příštím kroku vzájemná vzdálenost této dvojice zvětšila.

Hranice minimální vzdálenosti vozidel je dána součtem: Δ_{min}= (délka V+ délka V1)/2 +δ , kde δ je konstanta.

(2-8)
$$\Delta = \sqrt{(x_{V1} - x_V)^2 + (y_{V1} - y_V)^2}$$

Výhody

Nezáleží na vzájemném uspořádání vozidel

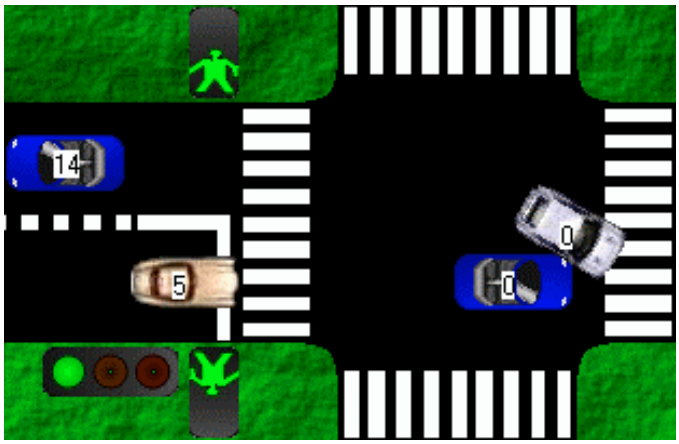
Lze aplikovat na příjezdových komunikacích i v křižovatce

Nevýhody

- Nutno ošetřit konflikty připojením
- Algoritmus neudává přednosti v jízdě, a proto nelze zavést hlavní a vedlejší komunikaci

Kolize:

Nefunguje na konfliktech připojením (konflikt připojením - [6] str. 2-47 obr. 2-4). Na Obr. 3.6 je situace vyznačena. Posunutí jakéhokoliv vozidla z dvojice o jeden krok nezpůsobí bezpečné zvětšení vzájemné vzdálenosti což vede k zastavení celé dvojice vozidel.



Obr. 3.6 Při sbíhavých konfliktech algoritmus blokuje dvojici vozidel. Číslo na vozidle představuje aktuální rychlost [počet pixelů/aktualizační krok]

3.5.2 Přednost na základě jednorozměrné souřadnice

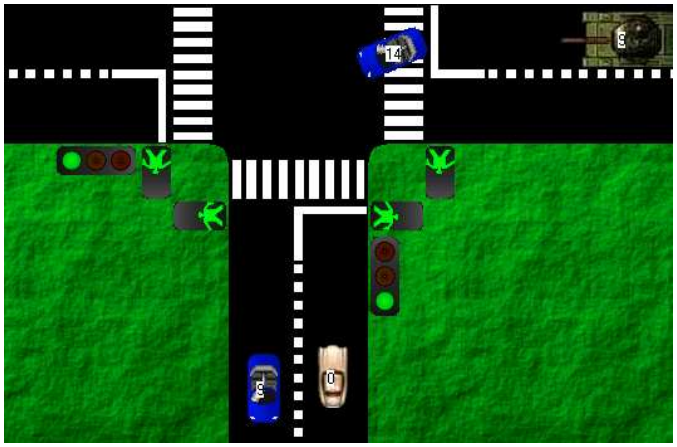
Tento algoritmus doplňuje předchozí algoritmus měření vzdálenosti. Pro každé vozidlo (V) hledáme druhé vozidlo (V1), které má nejmenší vzdálenost od V. V případě, že je jednorozměrná souřadnice V1 větší, pak vozidlu V nastavíme aktuální rychlost na nulu a vozidlu V1 nastavíme maximální rychlost. V opačném případě zastavíme V1 a maximální rychlostí pojede vozidlo V. V případě rovnosti jednorozměrných souřadnic nezáleží na tom, které z vozidel zastavíme.

Výhody:

- Jednoduchá implementace
- Rychlý algoritmus

Nevýhody:

- Vozidla nedodržují dopravní předpisy (náhodná přednost v jízdě)
- Nelze zavést hlavní a vedlejší komunikaci
- Nejbližší vozidlo, které způsobí zastavení V nemusí V vůbec ohrožovat. (Obr. 3.7)



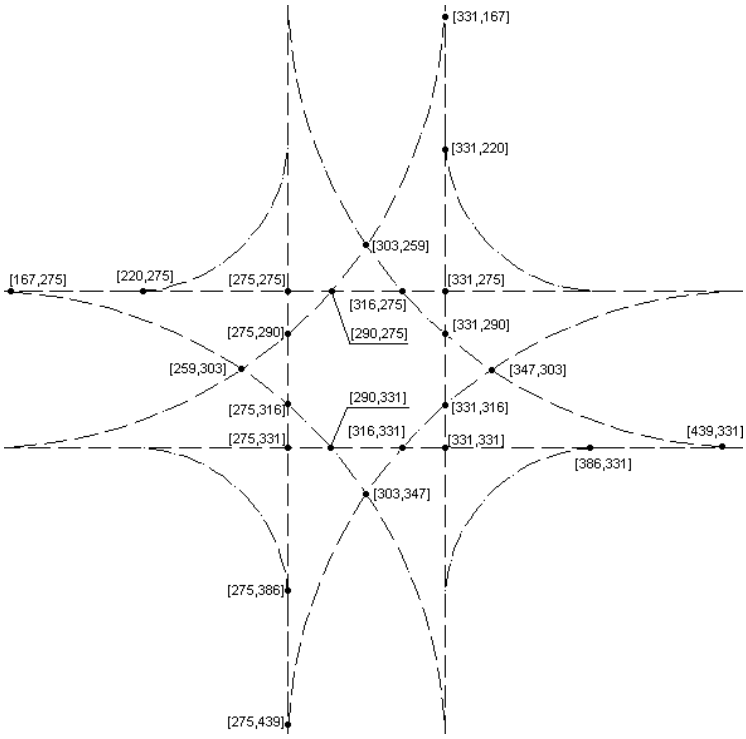
Obr. 3.7 Vozidlo dostane nulovou rychlost, neboť jeho nejbližší soused (jehož rychlost je 9 pixelů/krok) má větší jednorozměrnou souřadnici (je na výjezdu z křižovatky)

Kolize:

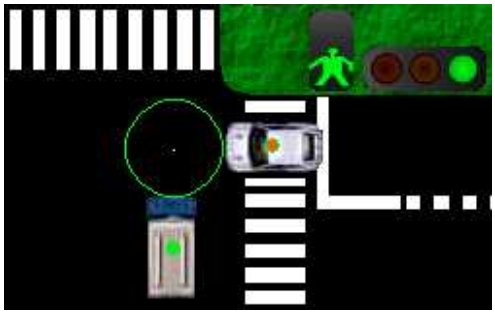
Algoritmus nezajistí ošetření všech kolizních situací v křižovatce. Pravidla pro zastavení se aplikují v nevhodný okamžik.

3.5.3 Měření vzdálenosti od konfliktních bodů

Průsečíky trajektorií označíme jako tzv. konfliktní body (jejich přehled je na Obr. 3.8). Každý průsečík má definovanou přednostní trajektorii. V případě, že se v okruhu konfliktního bodu nachází obě vodila přijíždějící po příslušných trajektoriích ke konfliktu, zastavíme vozidlo na trajektorii s menší prioritou a vozidlu, které jede po přednostní trajektorii nastavíme maximální rychlost. (Obr. 3.9)



Obr. 3.8 Konfliktní body křižovatky.



Obr. 3.9 V okolí každého konfliktního bodu zkoumám přítomnost vozidel na konfliktních trajektoriích. (zelená tečka představuje přítomnost na přednostní trajektorii)

Výhody:

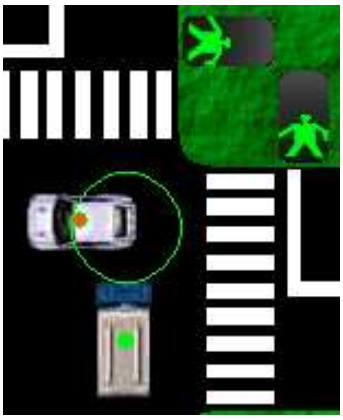
- Rychlý algoritmus
- Lze přesně definovat přednosti v jízdě

Nevýhody:

- Složitější implementace (nutno použít inicializační soubor pro souřadnice konfliktních bodů)

Kolize:

- V případě, že se vozidlo bez přednosti nachází za konfliktním bodem a přijede vozidlo po hlavní trajektorii je nuceno vozidlo s menší prioritou zastavit a tím může dojít ke srážce. (viz Obr. 3.10)



Obr. 3.10 Osobní vozidlo zastaví a nechá projet vozidlo nákladní. Dojde tak ke srážce.

3.5.4 Algoritmus chování vozidel použitý v tomto modelu

V této kapitole popíšeme celý algoritmus chování vozidel v neřízené a řízené křižovatce. Nejprve se budeme zabývat chováním vozidel v neřízené křižovatce, dále ukážeme změnu algoritmu pro řízenou křižovatku. Algoritmus chování vozidel v neřízené křižovatce neobsahuje detekci kolizí vozidel s chodci, neboť chování chodců není v neřízeném režimu křižovatky implementováno. Vývojový diagram celého algoritmu je na Obr. 3.11.

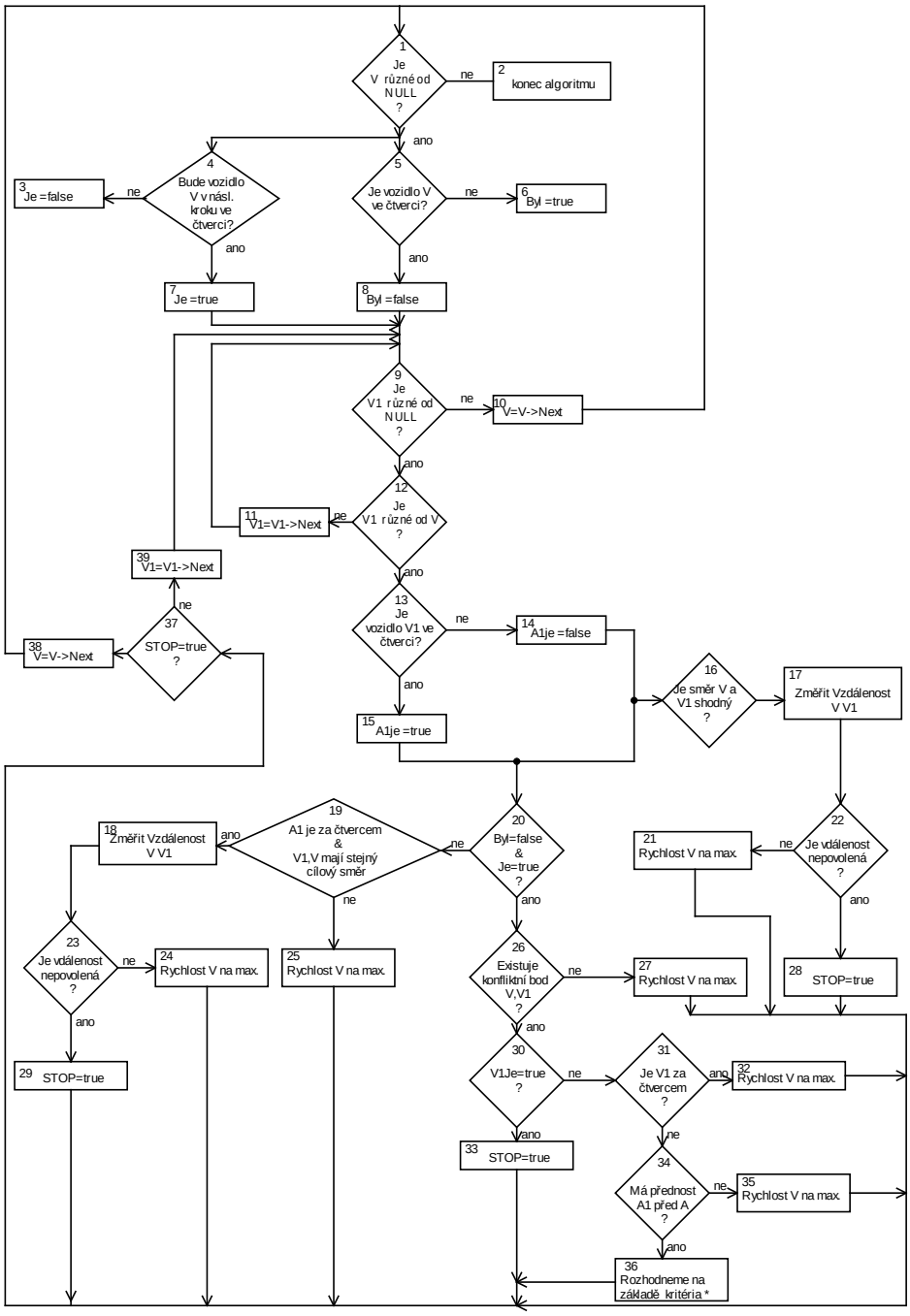
Křižovatku rozdělíme na dvě oblasti:

a) Příjezdové a odjezdové komunikace.

b) Oblast samotné křižovatky (dále též čtverec)

V cyklu procházíme všechna vozidla (*V*), která se v daný krok nachází na křižovatce. Po každé takové vozidlo najdeme (vnořeným cyklem) do dvojice vozidlo *V1* (opět z množiny všech vozidel v křižovatce). Poté zkoumáme vzájemné uspořádání vozidel ve dvojici a podle výsledku porovnávání zastavíme *V1*, nebo mu přidělíme maximální rychlost. Rychlost vozidla *V* algoritmus nemění (Je to dáno tím, že algoritmus vybírá dvojici pro každé vozidlo. Vozidlo *V* se tedy v některém příštím kroku ocitne v roli vozidla *V1*).

Bloky 1 až 15 představují inicializační část algoritmu. Zde zjišťujeme, zda se vozidla *V* a *V1* nachází ve čtverci v současném a v následujícím aktualizačním kroku. Dále tyto bloky (1 a 9) reprezentují koncové podmínky cyklů. Výraz *V1 je různý od NULL* plyne z uspořádání datových struktur modelu (viz 3.4).



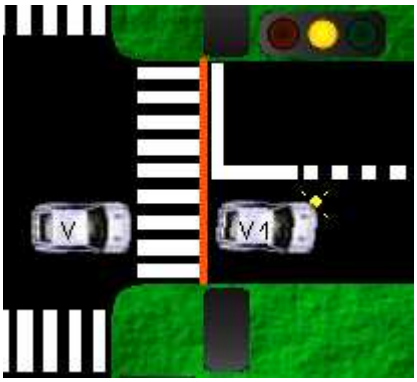
Obr. 3.11 Vývojový diagram algoritmu chování vozidel

V případě, že vozidla mají shodný směr jízdy (tzn. pohybují se po trajektoriích jednoho směru viz kap. 3.4), ohrožují se pouze, když mají rozdílnou rychlost, a tedy, když jedno vozidlo musí bezpečně dobrzdit za druhým. Rozhodneme, zda je jejich vzdálenost bezpečná (blok 22) a podle toho upravíme rychlost *V1* (bloky 28 a 21).

V případě, že vozidla nejedou po trajektoriích jednoho směru, mohou nastat tyto situace:

a) Vozidlo *V* není na vstupním rozhraní křižovatky
b) Vozidlo *V* je na vstupním rozhraní křižovatky a nemá konfliktní bod s V1
c) Vozidlo *V* je na vstupním rozhraní křižovatky a má konfliktní bod s V1

V případě a) je nutné zjistit, zda se vozidlo V1 nachází za čtvercem a zda V1 má stejný cílový směr s V (blok 19). To by znamenalo, že vozidlo V ještě může ohrozit cestu V1 na konfliktu připojením za čtvercem. Tato situace je naznačena na Obr. 3.12. Ohrozí-li vozidlo V cestu V1 (určíme na základě měření vzájemné vzdálenosti), nastavíme rychlost V na nulu.



Obr. 3.12 Vozidlo V ještě může ohrozit V1 na konfliktu přiblížením. Červená linka značí hranici čtverce.

Ve druhém případě (bod b)) je situace jednoduchá. Vozidlo V může pokračovat v jízdě do čtverce, neboť V1 jeho trajektorii neprotíná (blok 27).

Nejsložitější situace je v případě, že vozidlo V se nachází na rozhraní křižovatky a jeho trajektorie je v konfliktu s trajektorií vozidla V1. Je-li V1 již v křižovatce, musí vozidlo V zastavit a umožnit bezpečný průjezd vozidlu V1 (blok 33). Pakliže se V1 nenachází v křižovatce, mohou nastat dvě situace:

- Vozidlo V1 je na výjezdu z křižovatky.

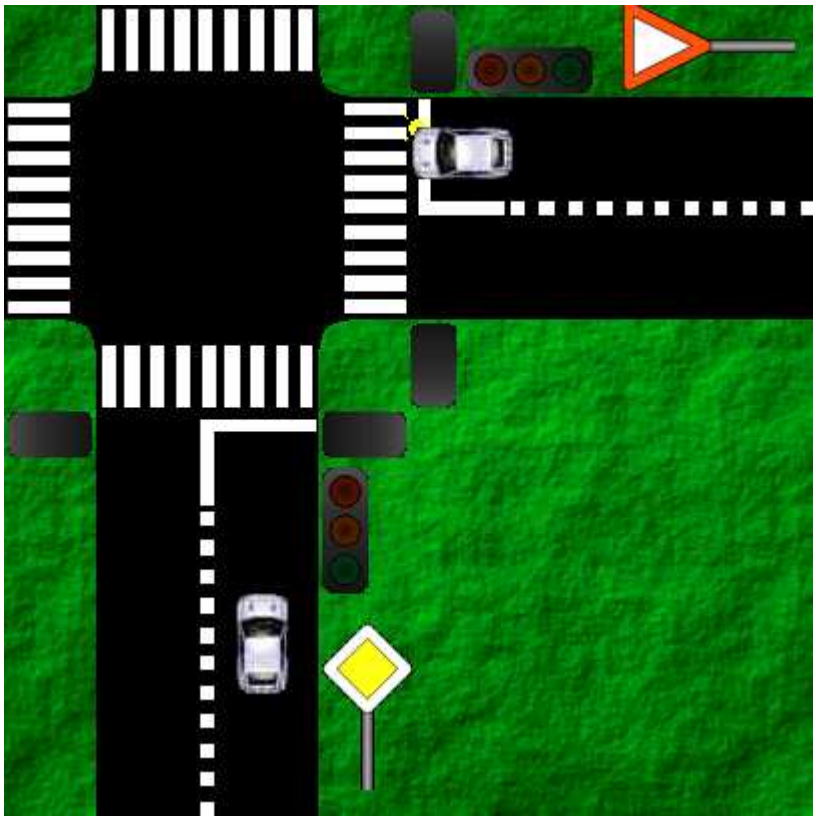
Vozidlo V může pokračovat v jízdě.

- Vozidlo V1 se blíží ke křižovatce po příjezdové komunikaci.

Jestliže naše vozidlo V má přednost (jeho trajektorie má větší prioritu) před přijíždějícím vozidlem, může pokračovat v jízdě.

V opačném případě musíme rozhodnout na základě nějakého kritéria, zda je přijíždějící vozidlo natolik vzdáleno, aby V bezpečně projelo křižovatkou a neohrozilo vozidlo (V1) s předností v jízdě (situace na Obr. 3.13).

Z kapitoly 3.4 víme, že rychlost vozidla udává počet obrazových bodů, které vozidlo urazí v jednom aktualizačním kroku. Pokud známe okamžitou vzdálenost obou vozidel od místa konfliktu, můžeme na základě jejich rychlosti vypočítat přibližný počet kroků nutný k dosažení konfliktního bodu. Tato hodnota představuje kriterium, na jehož základě se algoritmus rozhoduje v této situaci.



Obr. 3.13 O tom, zda vozidlo z vedlejší komunikace vjede do křižovatky, rozhodneme na základě kritéria

Výhody:

Vozidla dodržují dopravní předpisy

Algoritmus zajistí ošetření všech kolizí v křižovatce

Nevýhody:

Složitější implementace

Nutnost ošetření uváznutí

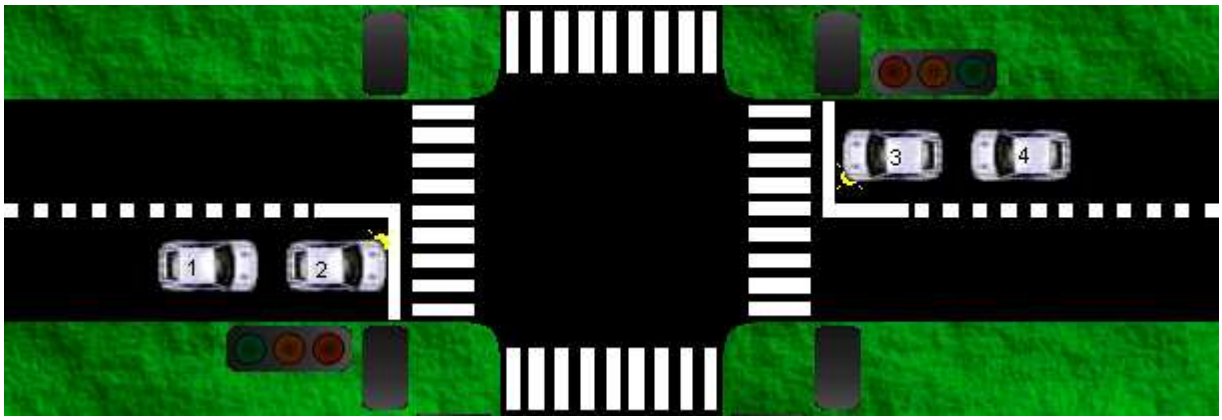
Kolize:

Algoritmus zajistí ošetření všech kolizí v křižovatce, avšak může dojít k uváznutí, jak popíšeme níže.

Uváznutí:

V případě, že se na protilehlých příjezdech nachází vozidla odbočující vlevo a zároveň za každým z těchto vozidel přijíždí vozidla jedoucí rovně, může dojít k uváznutí vozidel (situace je nakreslena na Obr. 3.14).

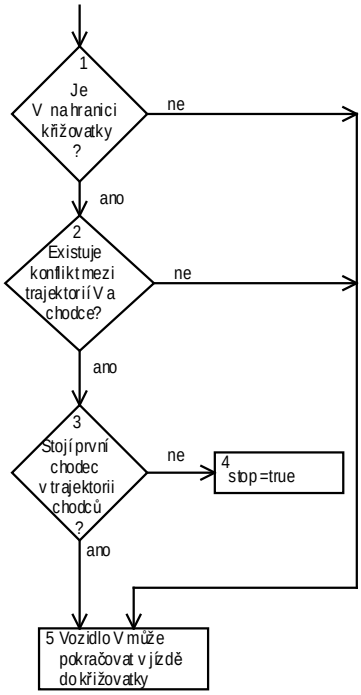
Vozidlo č. 2 nevjede do křižovatky (blok 36), neboť vozidlo č. 4 se nachází na přednostní trajektorii a je blízko čtverce. Ze stejného důvodu nebude pokračovat v jízdě vozidlo č. 3 blokované vozidlem č. 1. Celá čtveřice vozidel tedy stojí a provoz neuvolní ani příjezd dalšího vozidla.



Obr. 3.14 Uváznutí vozidel

Pro řešení této situace stačí v bloku 36 našeho vývojového diagramu přidat následující podmínku: *Jestliže V1 bude v následujícím kroku ve čtverci a zároveň V se nachází na protisměrné trajektorii a zároveň V1 stojí, pak rychlost V nastav na maximum.*

V řízené křižovatce je situace podobná: Vozidla V respektují všechna výše zmíněná pravidla, ale jen pro vozidla V1 na protilehlých trajektoriích. Dále je algoritmus chování vozidel pro řízenou křižovatku rozšířen o kontrolu kolizí s chodci. Kontrola kolizí s chodci se provádí na základě vývojového diagramu nakresleného na Obr. 3.15.

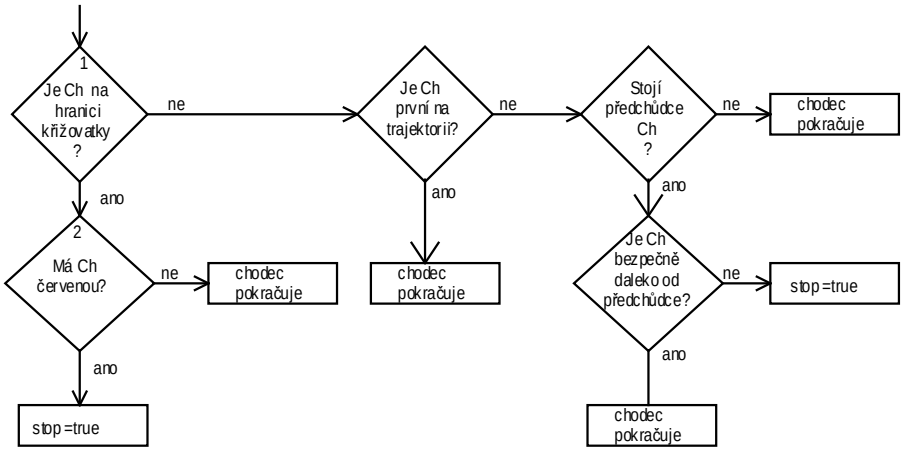


Obr. 3.15 Zajištění přednosti chodcům ve volném směru.

Při zjišťování, zda existuje kolize mezi trajektorií vozidla a chodce uvažujeme pouze ty konflikty, které vzniknou při odbočování. Při správném řízení křižovatky by totiž chodci na ostatních konfliktních trajektoriích měli čekat na uvolnění přechodu.

3.5.5 Chování chodců

Podobně jako u chování vozidel je třeba definovat pravidla, podle kterých se budou chodci pohybovat v křižovatce. Tato pravidla jsou velice jednoduchá, neboť model křižovatky nesimuluje chování chodců v neřízeném režimu. Chování chodce se řídí algoritmem, jehož vývojový diagram je na Obr. 3.16.



Obr. 3.16 vývojový diagram algoritmu chování chodců

Algoritmus provádí rozhodovací mechanismus vyjádřený vývojovým diagramem postupně pro všechny chodce v modelu. V případě, že je chodec na hranici křižovatky, rozhodne se na základě světelného signalizačního zařízení. V ostatních případech jde rychlostí předchůdce, nebo předchůdce dojde na bezpečnou vzdálenost.

4 Implementace

Program jsme vytvořili v prostředí Borland C++ Builder, přičemž pro využití I/O karty používáme ovladače od firmy Advantech. Projekt se skládá z hlavního formuláře, který obsahuje datový popis modelu a metody pro kreslení, a z komunikačního vlákna využívajícího knihovny Advantech. Důvody rozkladu aplikace na vlákna jsou popsány v 4.2.

4.1 Inicializace

Při startu programu (událost `OnCreate` hlavního formuláře) je nutné provést následující úkoly:

- načtení obrázků dopravních značek
- tvorba `TBimap` pro mezipaměť a pro uložení originálního pozadí
- Založení datové reprezentace modelu
- Mapování obrázků (uložených v DAT souborech) do paměti [C]
- Přenos obrázků v rámci operační paměti
- Odstranění mapování DAT souborů [C]
- Otevření deklaračního souboru křižovatky a naplnění datových struktur modelu
- Vytvoření tabulek konfliktů na základě informací uložených v inicializačních souborech

4.2 Vlákna

Na vstupy I/O karty přichází řídicí signály asynchronně, a proto je nutné cyklicky načítat hodnoty na jednotlivých vstupech. Načítání hodnot není nijak podmíněno, a tak by byla většina přiděleného procesorového času využita na funkci pro načítání vstupů, a aplikace by již nebyla schopna obsloužit výpočet modelu a jeho vykreslení.

Z tohoto důvodu je nutné pro načítání vstupů založit oddělený proces, který bude do rozvrhování procesorového času (operačním systémem) vstupovat samostatně.

Nejjednodušší způsob, jakým lze vytvořit takový oddělený proces, je přidat do projektu objekt typu vlákno (`TThread` viz [3]). Aplikace tedy nyní obsahuje vlákno hlavního formuláře a (nově přidané) komunikační vlákno, přičemž vykonávání každé z těchto částí programu probíhá nezávisle.

Při volání metod (resp. vlastností) komponent VCL z komunikačního vlákna se běh vlákna pozastaví, dokud se obsluha volané metody (resp. vlastnosti) nedokončí[5]. Proto je účelné vykreslovací metody deklarovat jako metody hlavního formuláře a též je odtud i volat.

[5] To se provádí metodou Synchronize, kterou používáme pouze v případě, když volaná metoda není tzv. vláknově bezpečná [3].

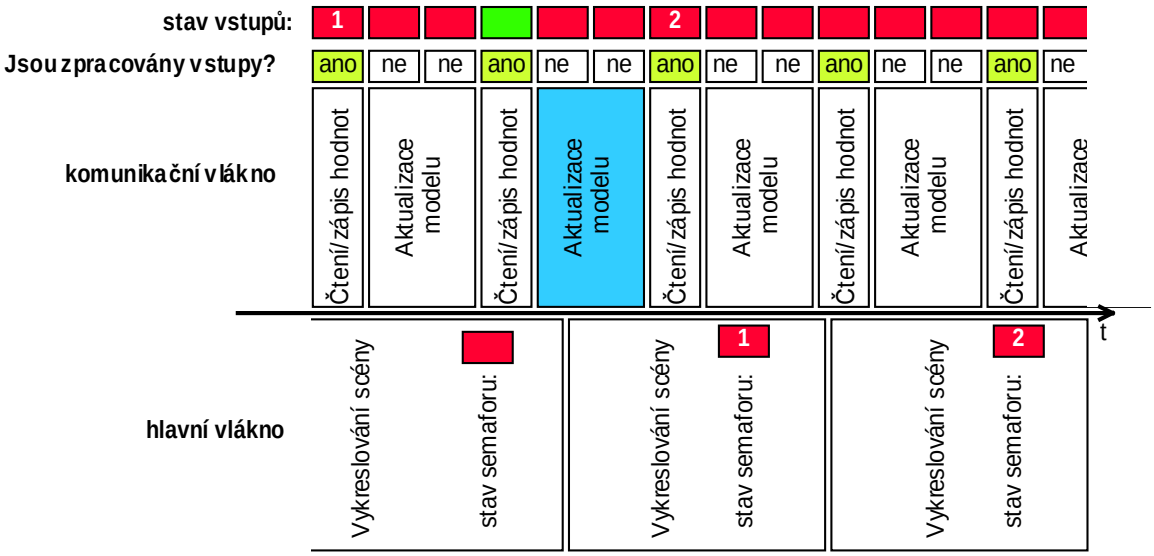
Aktualizační metody a datová reprezentace modelu mohou být tedy deklarovány v hlavním nebo v komunikačním vlákně. Na tom, kde bude model aktualizován a uložen prakticky nezáleží. Výhody a nevýhody obou řešení shrneme v následujících odstavcích.

4.2.1 Model v komunikačním vlákně

Každé načtení vstupů I/O karty se okamžitě projeví v datových strukturách modelu, neboť vlákno je vykonáváno paralelně s vykreslovacími metodami. To může způsobit nekonzistentní zobrazení v případě rychlých změn na vstupech modelu (takové změny na vstupech mohou nastat pouze v případě naprosto nevhodného řízení).

Kdybychom přesto uvažovali rychlou změnu stavu semaforů ze stavu „stůj“ na stav „volno“ a zpět, došlo by k tomu, že data modelu se budou aktualizovat okamžitě (vozidla čekající jako první se dají do pohybu) a po dokončení časově náročného kreslení by se začala vykreslovat scéna, při níž by na semaforu svítila červená a některá vozidla by byla v pohybu.

Celá situace je rozkreslena v Obr. 4.1. V modře vybarveném bloku aktualizační metody dochází k rozjezdu vozidel, přičemž ve všech metodách pro vykreslování scény je kreslen stav „stůj“. Čísla v obrázku upřesňují, který stav vykreslujeme ve scéně. Vykreslujeme totiž ty stavy, jejichž zpracování (blokem „Aktualizace modelu“) bylo dokončeno co nejpozději.



Obr. 4.1 Úlohy jednotlivých vláken (model deklarován v komunikačním vlákně).

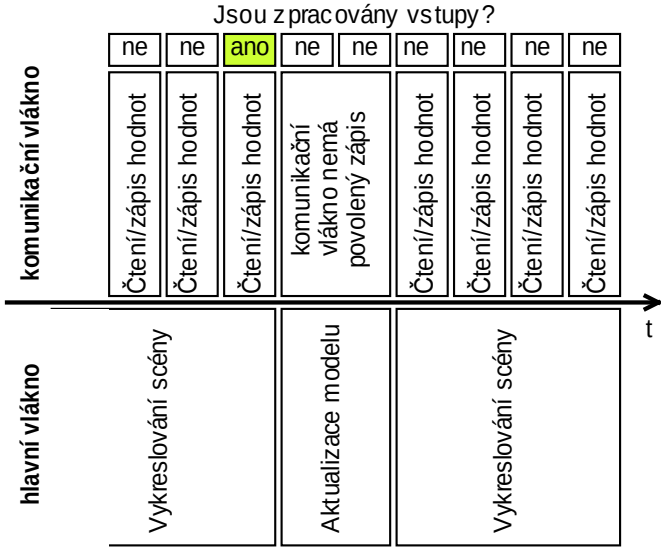
4.2.2 Model v hlavním vlákně

V tomto případě se celý model přepočítá pomocí aktualizačních metod a poté se vykreslí. Hodnoty vstupů se sice do datových struktur zapisují prakticky neustále, ale aktualizační metoda (která tato data využívá) se volá jednou za dobu vykreslení celé scény. Nedochází sice ke klamnému zobrazování, ale rychlost aktualizace dat

v modelu je závislá na kreslicí funkci. Časový průběh tohoto řešení je nastíněn na Obr. 4.2.

V obou případech je nutné synchronizovat činnost vláken, aby nebyla při zápisu do datových struktur narušena konzistence dat.

V práci jsme deklarovali aktualizační metody a datové struktury v hlavním formuláři. Bude platit, že hodnoty výstupů setrvají po dobu trvání kreslicí metody (resp. po dobu určenou časovačem kreslení viz Tab. 4.1) a tedy déle než jeden tzv. scan-cyklus PLC[6]. Synchronizaci vláken zajišťuji pomocí tzv. kritických sekcí (TCriticalSection viz [3]). Kritická sekce programu je část kódu, ve které operační systém (OS) neprovede přeplánování na úlohu využívající stejnou sdílenou paměť. Tím zaručíme, že OS nepředá čas procesoru kódu, který by způsobil nekonzistenci dat.



Obr. 4.2 Rozdělení úloh bude-li model deklarován v hlavním formuláři

4.3 Činnost hlavního vlákna

Hlavní vlákno má za úkol provést nad datovými strukturami modelu aktualizační výpočty a bezprostředně poté pomocí komponenty `TPaintBox` vykreslit stav modelu křižovatky na hlavní formulář. Na hlavním formuláři jsou též umístěny ovládací prvky modelu (např. menu, panel nástrojů apod.) a prvky pro zobrazení hodnot vstupů a výstupů. Dále hlavní formulář obsluhuje kód pro spouštění a ukončování činnosti komunikačního vlákna, generování vozidel a chodců na základě Poissonova rozdělení a inicializaci modelu.

4.3.1 Použité komponenty

V Tab. 4.1 uvádíme krátký přehled komponent použitých na hlavním formuláři.

[6] Programovatelný logický automat (PLC) v každém scan-cyklu uloží obraz vstupů do své paměti. V případě že nezaručím délku impulsu delší než tento scan-cyklus, nemusí PLC hodnotu zaznamenat.

Grafická reprezentace	Název	Popis
semafor aut J	TShape	Slouží k zobrazení stavu vstupů a výstupů
Soubor Zobrazit Nástroje	TMainMenu	Nastavování parametrů programu
	TPaintBox	Oblast pro vykreslování scény. Metoda Draw umožňuje vykreslit bitmapy ve formátu „.bmp“ na libovolnou souřadnici plátna. (počátek souřadnic se nachází v levém horním rohu)
	TTimer (nevizuální)	Program využívá tři časovače: Vykreslení scény, pomocný časovač pro odstranění kolizních bitmap chodců a pomocný časovač pro znamení o změně směru jízdy.

Tab. 4.1 Použité komponenty v hlavním vlákně

Popis komponent omezíme na popis praktického využití v modelu křižovatky. Podrobný popis všech metod a vlastností jednotlivých komponent naleznete v [3] nebo [8].

4.3.1.1 Způsoby kreslení modelu

Jak jsme již naznačili v kap. 4.2 jedná se o časově nejnáročnější metodu hlavního formuláře, a proto bylo třeba navrhnout algoritmus, který zajistí únosnou periodu vykreslování scény i při vysokém počtu vozidel v křižovatce. Metoda pro kreslení je volána v hlavním formuláři pomocí časovače (událost `OnTimer`), jehož perioda je delší než vykreslování samotné.

Nejprve popíšeme postup, jak lze scénu vykreslit pomocí komponenty `TPaintBox` s využitím mezipaměti.

Mezipaměť tvoří obrázek (`TBitmap`), na který nanesu pozadí a všechny grafické prvky, (pomocí metody `Bitmap->Canvas->Draw`). Pak (pomocí metody `PaintBox->Canvas->Draw`) vykreslíme celý obraz scény na kreslicí plátno komponenty `PaintBox`. Kdybychom kreslili přímo na plátno komponenty `PaintBox` jednotlivé grafické entity, výsledná animace by nepříjemně blikala.

Objekt `TCanvas` umožňuje vykreslit obrázek na libovolnou souřadnici pouze s nulovým úhlem natočení. Je tedy třeba každý obrázek před vložením na plátno natočit, a pak jej teprve umístit voláním metody `Draw`. Otáčení bitmapy vyžaduje, abychom pro každý obrazový bod původního nenatočeného obrázku spočítali nové souřadnice a hodnotu barvy pixelu přenesli na tyto nové souřadnice (blíže v [9]). (Je třeba založit nový obrázek, do kterého budeme kreslit výsledek rotace). Tato operace je velmi náročná na čas procesoru, a tak se doba potřebná pro kreslení modelu výrazně zvyšuje s rostoucím počtem vozidel v křižovatce.

Chceme-li eliminovat tento nedostatek, je třeba uložit otočené obrázky na pevný disk, a při startu programu je načíst do vhodných struktur. Při kreslení pak vyhledáme v paměti obrázek s příslušným úhlem a ten vykreslíme.

Je zřejmé, že za běhu programu budou předtočené obrázky zabírat mnoho místa v operační paměti. Za zrychlení vykreslovací části programu tedy zaplatíme vysokou paměťovou náročností.

4.4 Činnost komunikačního vlákna

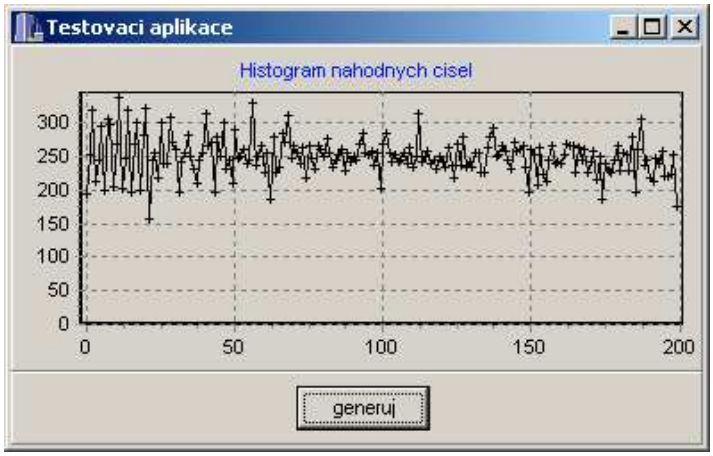
Komunikační vlákno koná v zásadě tyto 4 činnosti:

1) Čtení aktuálních výstupů modelu.
2) Zápis výstupů modelu a na výstupy I/O karty
3) Čtení vstupů I/O karty
4) Zápis informace ze vstupů do datových struktur křižovatky

Tyto činnosti se provádějí v nekonečné smyčce a to ve výše zmíněním pořadí. Je třeba zdůraznit, že čtení výstupů modelu můžeme provádět pouze v době, kdy model tyto informace svou činností nemění. Podobně zápis do datových struktur je možný pouze v době, kdy model tyto informace nevyužívá pro své výpočty nebo pro kreslení. To lze zaručit pomocí již zmíněných kritických sekcí, které zajistí, že po dobu čtení výstupů modelu nedojde k přeplánování na hlavní vlákno, a tedy nedojde k změně čtených dat. Použitím kritické sekce v době zápisu do datových struktur modelu zajistíme, aby model nezačal používat informace dříve, než je zápis ukončen. Mohlo by totiž dojít k tomu, že stav některých vstupů je z předchozího kroku a stav ostatních vstupů je již v tomto kroku aktualizován.

4.5 Generátor náhodných čísel

Generátor náhodných čísel implementovaný v prostředí Borland C++ Builder (funkce `rand()`) umožňuje generovat pseudonáhodná čísla podle rovnoměrného rozdělení. Toto tvrzení jsme ověřili malou aplikací, která generuje náhodná čísla v rozsahu 0 až 200. Na Obr. 4.3 je vykreslen histogram náhodných jevů vygenerovaný tímto programem.



Obr. 4.3 Histogram generátoru pseudonáhodných čísel (funkce `rand()`)

V kapitole 2.2 je požadováno, aby doba mezi příjezdy vozidel (nebo doba mezi příchody chodců) měla negativní exponenciální rozdělení. To lze zajistit pomocí vztahu z [8]:

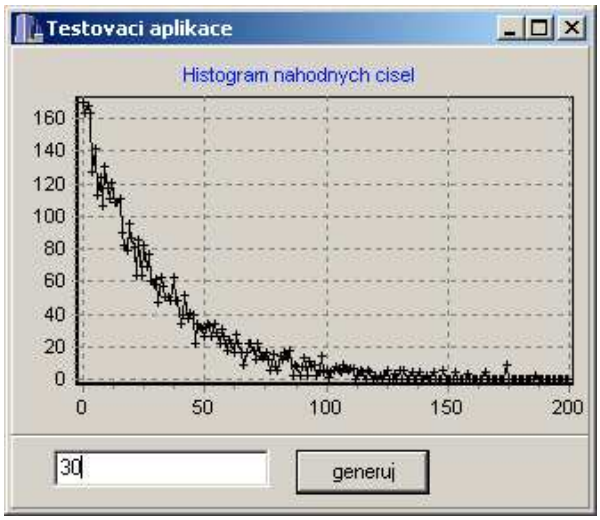
$r = -M \cdot \ln(U)$, kde

r je náhodná veličina s exponenciálním rozdělením (celé číslo),

M odpovídá střednímu intervalu mezi vozidly $\bar{h}$ (kap. 2.2)

a U je náhodná veličina (reálné číslo) s rovnoměrným rozdělením $U \in (0,1)$.

Po aplikaci tohoto výrazu (při $\bar{h} = 30$) a omezení generovaných čísel v rozsahu 0 až 200 dává testovací aplikace výsledky na Obr. 4.4.



Obr. 4.4 Histogram generátoru náhodných čísel s negativním exponenciálním rozdělením

Nyní máme k dispozici generátor náhodných čísel s negativním exponenciálním rozdělením. V projektu křižovatky periodicky voláme aktualizační metodu, která přepočítá a vykreslí stav modelu. Zavedeme-li čítač volání této metody, máme dispozici časovou osu jejíž jednotkou je aktualizační krok. Přidáme vozidlo do modelu a vynulujeme tento čítač. Zároveň necháme vygenerovat náhodné číslo, které udává za jaký počet kroků budeme přidávat další vozidlo. Jakmile uplyne tento počet aktualizačních kroků, vynulujeme čítač času, přidáme vozidlo a necháme vygenerovat nový časový interval.

Těchto generátorů časových intervalů používám v projektu křižovatky celkem sedm. Při generování je totiž třeba respektovat příjezd vozidla odbočujícího vlevo, jedoucího rovně a odbočujícího vpravo (tedy tři generátory). Intenzity příjezdů se budou značně lišit na hlavní komunikaci a na vedlejší komunikaci. Zavedeme tedy tři generátory pro hlavní i vedlejší komunikaci zvlášť (máme tedy celkem šest generátorů). Sedmý generátor použijeme pro přidávání chodců do modelu.

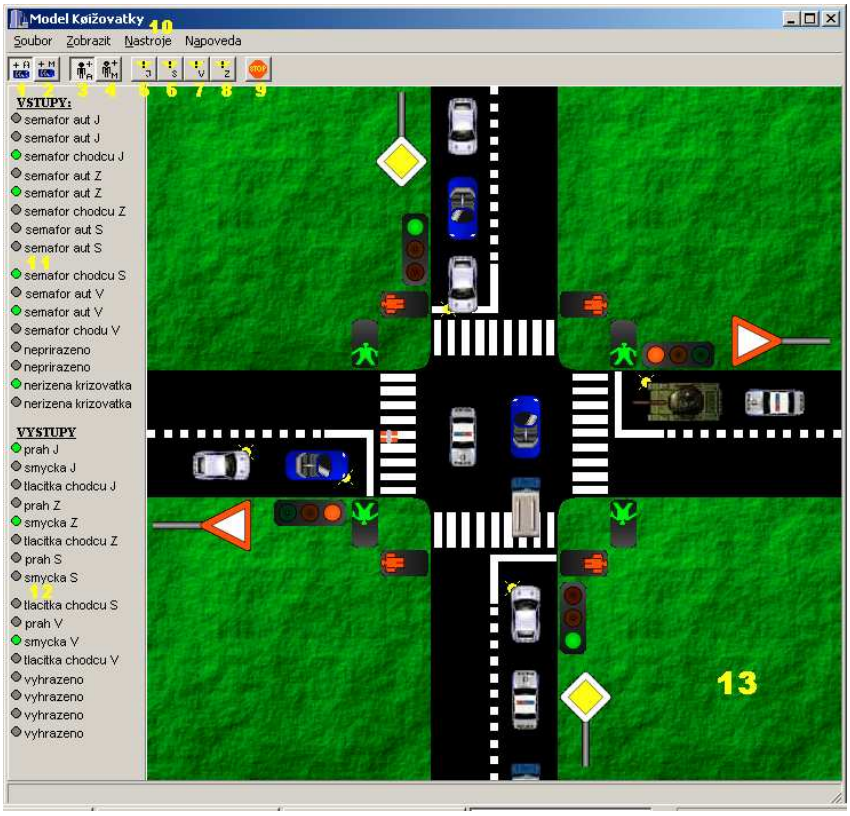
Každý z těchto generátorů má tři parametry:

- Minimální interval mezi vozidly
- Střední interval mezi vozidly
- Maximální interval mezi vozidly

Všechny parametry lze v programu nastavit nebo je možné použít předvolby pro jednotlivé druhy řízení.

4.6 Vzhled aplikace a stručný popis funkcí

Aplikace s modelem křižovatky je navržena pro systém Microsoft Windows a tím je částečně předurčen její vzhled. Na Obr. 4.5 je sejmutá obrazovka programu. Žlutá čísla slouží pro popis ovládacích prvků programu v tomto textu.



Obr. 4.5 Sejmutá obrazovka aplikace s virtuálním modelem křižovatky.

První tlačítko nástrojové lišty (označené číslem 1) slouží k aktivaci generátorů vozidel. Po klepnutí na toto tlačítko začne program přidávat do modelu vozidla způsobem popsaným v kapitole 4.5. Parametry generování lze kdykoli upravit v dialogovém okně přístupném z hlavní nabídky (10) pomocí volby „Nástroje“->“Parametry generátorů”. Generování vozidel můžeme ukončit opětovným klepnutím na tlačítko 1. Je též možné přidat vozidlo ručně, což umožňuje ikona 2, která vyvolá dialogové okno pro přidání vozidla. Zde nastavíme rychlost, obrázek a trajektorii vozidla. Stejný význam mají tlačítka 3 a 4, přičemž tlačítko 3 spustí generátor chodců a tlačítko 4 vyvolá dialogové okno pro přidání chodce na požadovanou trajektorii.

V případě, že je povoleno dohlídání červeného signálu (pomocí volby „Nástroje“->“Nastavení“), je možné uměle vyvolat poruchu světel. To umožňují tlačítka 5 až 8, která vyvolají poruchu červeného světla v daném směru (J, Z, S, V).

Tlačítko 9 provede reset modelu. To znamená, že se odstraní všechna vozidla a všichni chodci z modelu.

V oblasti 11 jsou periodicky vykreslovány stavy všech vstupů. Zelená barva indikátoru představuje aktivní úroveň, šedá představuje úroveň pasivní. Stavy výstupů jsou vykreslovány v oblasti 12 a indikace logické úrovně probíhá stejně jako u vstupů.

V každém aktualizačním kroku je stav křižovatky překreslován. Kreslení scény probíhá v oblasti označené číslem 13.

Model zapisuje významné události do souboru „krizovatka.log“ v adresáři programu. Tento soubor je vytvářen při běhu programu a jeho obsah lze zobrazit volbou „Zobrazit“->“Historii událostí“. Do souboru model zapisuje tyto informace:

- Spuštění modelu
- Přidání vozidla na trajektorii s danou rychlostí
- Odebrání vozidla z modelu
- Kolize mezi vozidly
- Kolize mezi vozidlem a chodcem
- Reset modelu
- Ukončení modelu

Informace o trajektorii vozidla je v souboru historie vyjádřena číslem v rozsahu 0 až 11. Přiřazení trajektorií k těmto hodnotám je vysvětleno v Tab. 4.2.

Virtuální model dále umožňuje sledovat počty vozidel a chodců v jednotlivých směrech, histogramy jednotlivých generátorů a četnosti kolizí. Tyto informace se zobrazí v podobě grafu v okně „Statistika“, které lze vyvolat volbou „Zobrazit“ -> „Statistiku“. Histogramy usnadní volbu parametrů jednotlivých generátorů (volba „Nastaveni“ –> „Parametry generátorů“).

Hodnota	Popis
0	Vozidlo přijíždí z jižního směru a odbočuje vlevo
1	Vozidlo přijíždí z jižního směru a jede přímo
2	Vozidlo přijíždí z jižního směru a odbočuje vpravo
3	Vozidlo přijíždí ze západního směru a odbočuje vlevo
4	Vozidlo přijíždí ze západního směru a jede přímo
5	Vozidlo přijíždí ze západního směru a odbočuje vpravo
6	Vozidlo přijíždí ze severního směru a odbočuje vlevo
7	Vozidlo přijíždí ze severního směru a jede přímo
8	Vozidlo přijíždí ze severního směru a odbočuje vpravo
9	Vozidlo přijíždí z východního směru a odbočuje vlevo
10	Vozidlo přijíždí z východního směru a jede přímo
11	Vozidlo přijíždí z východního směru a odbočuje vpravo

Tab. 4.2 Označení trajektorií v souboru historie

4.7 Požadavky na hardware

Minimální požadavky na hardware počítače na kterém bude virtuální model spuštěn jsou shrnuty v Tab. 4.3.

Parametr	Minimální hodnota
Procesor	Intel celeron 566Mhz
Paměť	128 MB
Pevný disk	35 MB volného místa
Grafická karta	4 MB, kompatibilní s MS Windows
Optická mechanika	CD-ROM libovolné rychlosti
Operační systém	Testováno na MS Windows NT 4.0, MS Windows 2000 a MS Windows XP Professional

Tab. 4.3 Požadavky na hardware počítače s virtuálním modelem

5 Charakteristiky modelu z hlediska řízení

Úkolem této diplomové práce není hledání samotného algoritmu pro řízení, ten je podrobně popsán v [2], a proto uvedeme pouze přehled vlastností modelu, pro usnadnění volby řídicího algoritmu.

5.1 Vlastnosti modelu vzhledem k řízení

Při návrhu řídicího algoritmu je vhodné dodržet následující doporučení:

- Při dohlídání červené je třeba počítat se zpožděním potvrzovacího signálu o 200ms.
- Mezičas [6] pro vozidla byl stanoven [2] na 5 sekund.
- Mezičas pro chodce byl stanoven [2] na 6 sekund.
- Pro režim „řízená křižovatka“ je nutné nastavit 14. vstup modelu na logickou 1 a 15. vstup na logickou 0.
- Pro režim „blikavá oranžová“ je nutné nastavit 14. vstup modelu na logickou 0 a 15. vstup na logickou 1. Dále je třeba na 13. periodicky přepínat úrovně pro ovládání oranžového světla.
- Ostatní stavy vstupů 14 a 15 vyvolají vypnutí světelného signalizačního zařízení.
- Model umožňuje zvolit dva přednastavené režimy pro přidávání účastníků silničního provozu (volba „Nastavení“ –> „Parametry generátorů“ –> „Předvolby“). První volba představuje mírně rozvážený provoz na hlavní a vedlejší komunikaci a je tedy vhodný pro implementaci pevného fázového programu. Druhý je naopak vhodný pro implementaci dopravně závislého řízení, neboť provoz na hlavní komunikaci je výrazně vyšší než provoz na komunikaci vedlejší.

V Tab. 5.1 je souhrn všech vstupů modelu. Dále tabulka obsahuje význam jejich hodnot (barvy pozadí buněk odpovídají barvám na semaforu). Přehled a význam výstupů je uveden v Tab. 5.2.

Vstup	Význam vstupu	Význam hodnot			
0	semafor vozidel JIH	1	0	0	1
1		0	0	1	1
2	semafor chodců JIH	1	0		
3	semafor vozidel ZÁPAD	1	0	0	1
4		0	0	1	1
5	sem. chodců ZÁPAD	1	0		
6	semafor vozidel SEVER	1	0	0	1
7		0	0	1	1
8	sem. chodců SEVER	1	0		
9	semafor vozidel VÝCHOD	1	0	0	1
10		0	0	1	1
11	sem. chodců VÝCHOD	1	0		
12	nepřiřazeno				
13	oranžová v neř. módu	1	0		
14	mód křižovatky	1	0	0	1
15		0	0	1	1

Mód křižovatky:

- Řízená křižovatka
- Neřízená křižovatka - semafory vypnuty
- Neřízená křižovatka - mód blikavé oranžové

Tab. 5.1 Přehled vstupů modelu

Výstup	Popis
0	Práh indikující příjezd vozidla z jižního směru
1	Indukční smyčka na jižním vjezdu do křižovatky
2	Tlačítko chodců pro jižní přechod
3	Práh indikující příjezd vozidla ze západního směru
4	Indukční smyčka na západním vjezdu do křižovatky
5	Tlačítko chodců pro západní přechod
6	Práh indikující příjezd vozidla ze severního směru
7	Indukční smyčka na severním vjezdu do křižovatky
8	Tlačítko chodců pro severní přechod
9	Práh indikující příjezd vozidla z východního směru
10	Indukční smyčka na východním vjezdu do křižovatky
11	Tlačítko chodců pro východní přechod
12	Dohlídání červené v jižním směru.
13	Dohlídání červené v západním směru.
14	Dohlídání červené v severním směru.
15	Dohlídání červené ve východním směru.

Tab. 5.2 Přehled výstupů modelu

6 Závěr

V rámci diplomové práce jsme navrhli funkční virtuální model křižovatky, který je možné řídit pomocí PLC nebo logických obvodů TLL (při využití převodníku úrovní řídicích signálů popsaného v [2]). Tento model je určen pro podporu výuky v předmětech vypsaných Katedrou řídicí techniky Fakulty elektrotechnické, Českého vysokého učení technického v Praze.

Provoz vozidel je umožněn v řízeném i neřízeném módu křižovatky. V neřízeném módu není křižovatka ovlivněna chodci. Chování vozidel v neřízeném provozu probíhá na základě pravidel silničního provozu a vozidla se vzájemně neohrožují. V případě chybného řízení křižovatky světelným signalizačním zařízením může mezi účastníky provozu dojít ke kolizi.

Model využívá pro popis pohybu vozidel a chodců předem definovaných trajektorií. Rychlost pohybu účastníků silničního provozu nabývá dvou hodnot. Vozidla se pohybují svojí maximální rychlostí nebo mají rychlost nulovou. Chodci stojí nebo se pohybují maximální rychlostí, která je pro však všechny stejná.

Počet účastníků silničního provozu za nějaký časový interval se v této křižovatce řídí náhodnou veličinou s Poissonovým rozdělením.

Navržený model používá pro svůj vnitřní popis obecnou datovou strukturu (znázorněnou na Obr. 3.5), která umožňuje popis prakticky libovolné topologie křižovatky.

Virtuální model obsahuje simulaci bezpečnostního vybavení pro dohlídání červeného signálu. Při poruše červeného návěstidla musí řídicí algoritmus zajistit okamžitý přechod do režimu blikavé oranžové. Dále program obsahuje předvolby modelování provozu, které jsou vhodné pro řízení pevnou fází nebo pro řízení dopravně závislé řízení. Tyto vlastnosti modelu reprezentují různé obtížnosti řízení.

Na tuto práci lze navázat rozšířením těchto vlastností modelu:

- Distribuční funkce udávající počet účastníků silničního provozu je v případě kongesce značně odlišná od použité Poissonovy distribuce [6]. Bylo by proto účelné vytvořit v modelu generátor náhodných čísel s negativním binomickým rozdělením.
- Pro lepší modelování bezpečnostního vybavení křižovatky je třeba zavést dohlídání zeleného signálu. Při použití I/O karty Advantech PCI-175 by bylo nutné navrhnout komunikační protokol, který by redukoval počet použitých vstupů a výstupů.
- Pro vykreslování stavu křižovatky používáme standardní komponenty vývojového nástroje. Použitím rozhraní DirectX by se zřejmě vylepšila plynulost animací. Urychlované vykreslování by snížilo celkové hardwarové nároky programu.

- Dále lze model rozšiřovat v oblasti chování vozidel a chodců jako např. modelování rozjezdů vozidel, vznik kolizí na příjezdových komunikacích, přiřazení různých rychlostí chodcům a podobně.

7 Použitá literatura

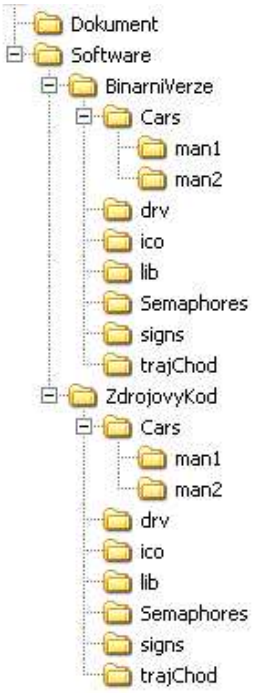
[1] Šišlák, D.: Řízení výtahu. Diplomová práce FEL ČVUT 2003, Praha
[2] Rampas, M.: Výkonové přizpůsobení PCI-1750. Diplomová práce FEL ČVUT 2004, Praha
[3] Inprise Corporation: Borland C++ Builder Help
[4] Advantech Corporation: PCI – 1750 32 channel Isolated Digital I/O card User's Manual, Advantech Co., Ltd. 1998
[5] Advantech Corporation: Advantech DLL drivers user's manual and programmer's reference (3rd Edition) , Advantech Co.,Ltd. 2000
[6] Přibyl, P., Mach, R.: Řídicí systémy silniční dopravy, skriptum ČVUT-FD, 2003
[7] Silniční vývoj s. r. o.:Navrhování světelných signalizačních zařízení pro řízení silničního provozu, Centrum dopravního výzkumu Brno, 1996
[8] Devroye, L.: Non-Uniform Random Variate Generation, Springer-Verlag, 1986.
[9] Šusta, R.: Programovací jazyky pro řízení ve Windows, ČVUT, 1999

8 Použitý software

[A] Inprise Corporation: Borland C++ Builder™ version 5.0
[B] Advantech Corporation: Advantech DLL drivers
[C] Ing. Richard Šusta: Objekt pro mapování obrázků do paměti

Příloha 1

Adresářová struktura přiloženého CD-ROM je znázorněna na obrázku 1.



Obrázek 1 Adresářová struktura CD-ROM.

V tabulce 1 je stručný popis obsahu jednotlivých adresářů.

Adresář	Popis obsahu
Dokument	Obsahuje diplomovou práci včetně této přílohy ve formátu MS Word.
\software\BinarniVerze	Přeložený program. Je možné jej spustit bez nutnosti instalace.
\software\ZdrojovyKod	Zdrojové texty programu určeny pro překlad pomocí C++ Builder 5.
Podadresáře projektu	
\BinarniVerze\cars	Bitmapy vozidel DAT souborech a bitmapy chodců v BMP souborech (adresáře `man1` a `man2`).
\BinarniVerze\drv	Hlavičkový soubor ovladače Advantech.
\BinarniVerze\ico	Ikony pro nástrojovu lištu.
\BinarniVerze\lib	Knihovna Advantech (soubor LIB).
\BinarniVerze\Semaphores	Bitmapy semaforů (vozidel i chodců) pro všechny směry.
\BinarniVerze\signs	Obrázky se značkami pro jednotlivé směry.
\BinarniVerze\trajChod	Obrázky použité v dialogovém okně pro přidání chodce.

Tabulka 1 Obsah adresářů přiloženého kompaktního disku.