Univerzita Pardubice
Dopravní fakulta Jana Pernera

Emulace funkčních algoritmů řadiče světelného signalizačního zařízení pro řízení silničního provozu

Bakalářská práce

2021 Miroslav Čadský

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miroslav Čadský**
Osobní číslo:	**D18219**
Studijní program:	**B3709 Dopravní technologie a spoje**
Studijní obor:	**Elektrotechnické a elektronické systémy v dopravě**
Téma práce:	**Emulace funkčních algoritmů řadiče světelného signalizačního zařízení pro řízení silničního provozu.**
Zadávající katedra:	**Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě**

Zásady pro vypracování

1. Volba dostupných HW a SW prostředků vhodných pro emulaci funkčního chování řadiče a příslušných periférií.
2. Specifikace požadovaného emulovaného funkčního chování a dalších souvisejících vlastností.
3. Stanovení architektury a struktury SW včetně interakcí s okolím a vzájemné interakce dílčích částí SW.
4. Definice testových případů pro ověření požadovaného chování a vlastností emulace.
5. Implementace SW, jeho odladění a spuštění na zvoleném HW.
6. Provedení a vyhodnocení testů požadovaného chování a vlastností emulace.

Rozsah pracovní zprávy:
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

- *Navrhování světelných signalizačních zařízení pro řízení provozu na pozemních komunikacích: technické podmínky : TP 81.* [Praha]: Ministerstvo dopravy a Ředitelství silnic a dálnic ČR, 2015. [cit. říjen 2020]. Dostupné online : http://www.pjpk.cz/data/USR_001_2_8_TP/TP_81.pdf
- ČSN EN 12675. *Řadiče světelných signalizačních zařízení - Funkčně bezpečnostní požadavky.* Česká agentura pro normalizaci, 2018.
- HEROUT, Pavel. *Učebnice jazyka C. 4.*, přeprac. vyd. České Budějovice: Kopp, 2004, 271 s. ISBN 8072322206.
- WILLIAMS, Elliot. *AVR programming: Learning to Write Software for Hardware.* Sebastopol, Calif.: Maker Media, 2014. Make.magazine.com. ISBN 1449355781.

Vedoucí bakalářské práce: **Ing. Jan Ouředníček, Ph.D.**
Katedra elektrotechniky, elektroniky a zabezpečovací techniky v dopravě

Datum zadání bakalářské práce: **12. listopadu 2020**
Termín odevzdání bakalářské práce: **17. května 2021**

L.S.

doc. Ing. Libor Švadlenka, Ph.D.
děkan

Ing. Dušan Čermák, Ph.D.
vedoucí katedry

V Pardubicích dne 8. března 2021

Prohlašuji:

Práci s názvem Emulace funkčních algoritmů řadiče světelného signalizačního zařízení pro řízení silničního provozu jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne

Miroslav Čadský v. r.

Poděkování:

Tímto bych chtěl poděkovat vedoucímu mé bakalářské práce Ing. Janu Ouředníčkovi, Ph.D. za odborné vedení, cenné rady, podměty, připomínky a věnovaný čas i v této nelehké době při zpracování mé bakalářské práce. Dále bych chtěl poděkovat Ing. Zdeňku Maškovi, Ph.D. za zapůjčení HW, který jsem při zpracování této bakalářské práce využil. Za podporu při mém dosavadním studiu bych chtěl také poděkovat své rodině, která mi byla vždy oporou.

Anotace

Cílem bakalářské práce je definice požadavků na signální plán pro emulaci řadiče světelného signalizačního zařízení a následné vytvoření SW, dle stanovených požadavků. Součástí práce je i spuštění a odladění činnosti daného programu na zvoleném HW. Poslední část práce je zaměřena na testování činnosti emulace dle nadefinovaných testů pro dané požadavky s následným vyhodnocením splnění požadovaného chování.

Klíčová slova

Emulace algoritmu řadiče, Řízení křižovatky, Světelné signalizační zařízení

Title

Functional algorithm emulation of Traffic light signals controller for traffic control

Annotation

The aim of this bachelor thesis is definition of requirements for signal plan for emulation of traffic light signals controller followed by creation of SW according to requirements. Next part is also SW activation on chosen HW and its debugging. Final part of the bachelor thesis is focused on testing functions of the emulation according to defined tests for given requirements.

Keywords

Functional algorithm for traffic controller, Intersection management, Traffic lights

OBSAH

Seznam obrázků

Seznam tabulek

Seznam zkratek

SW – Software

HW – Hardware

LED – Light-emitting diode

TP – Technické podmínky

SSZ – Světelné signalizační zařízení

FSM – Finite-state machine

ISR – Interrupt service routine

MHD – Městská hromadná doprava

RŘ – Ruční řízení

VA (VB, VC) – signál pro **V**ozidla

PA (PB) – signál pro **P**ěší

DPA (DPB) – dopravní **D**etektor

SC – signál doplňkové zelené **Š**ipky

ÚVOD

Hlavním cílem této bakalářské práce je vytvoření emulace algoritmu řízení světelného signalizačního zařízení na křižovatce. Jedná se o realizaci bezpečnostně kritického systému (SW), který má stanovené normativní postupy návrhu které se dělí na několik částí. První část představuje definice požadavků chování emulace, následuje tvorba SW dle zadaných požadavků. Dále jsou definovány testovací případy pro testování činnosti emulace. Tyto testy jsou následně provedeny a vyhodnoceny, čímž se porovná finální chování emulace se zadanými parametry. Jedním z podkladů pro tvorbu této práce byly zkušenosti, které jsem získal při tvorbě semestrální práce z předmětu mikroprocesorová řídící technika. V něm jsem si vyzkoušel jednotlivé funkce pro řízení signálů semaforu s jednoduchým signálním plánem.

Řízení silničních křižovatek prošlo historicky určitým vývojem. První způsoby řízení křižovatek obstarávali většinou policisté ať už pomocí gumového obušku sloužícího pro lepší viditelnost pokynů policisty nebo později ručním přepínáním signálu na mechanickém semaforu. První automatické světelné SSZ na území dnešní České republiky bylo nainstalováno v Praze v roce 1930. Jednalo se však o velmi jednoduché elektromechanické zařízení, které bylo umístěno uprostřed křižovatky a dle přesně definovaných časových intervalů střídalo červené, žluté a zelené signály. Zcela však chyběly směrové signály pro vozidla nebo samostatné semafory pro chodce. Hlavní nevýhodou těchto zařízení byly již zmíněné pevně dané časové intervaly, které často vedly naopak ke zpomalování silničního provozu (Hrábek, 2015). V dnešní době jsou už zařízení (tzv. řadiče) pro řízení signálů na křižovatkách mnohem důmyslnější. Pro každý dopravní proud mohou disponovat samostatnou signalizací a především dokáží reagovat na dynamiku silničního provozu. To zajišťují pomocí řady detektorů, které řadiči předávají informace například o přítomnosti silničních vozidel v jednotlivých jízdních směrech, či umožňují chodcům sdělit řadiči svůj záměr přejít po přechodu pro chodce. Díky všem těmto informacím je pak řadič schopen dle naprogramovaných algoritmů upravovat délky či pořadí jednotlivých zelených signálů a tím optimalizovat kapacitu křižovatky.

Při mém návrhu algoritmu SW jsem z hlediska dynamiky signálního plánu uvažoval právě vstupy od tlačítek chodců. Záměrem je, aby daný SW emulující řadič křižovatky dokázal reagovat i na externí vstupy.

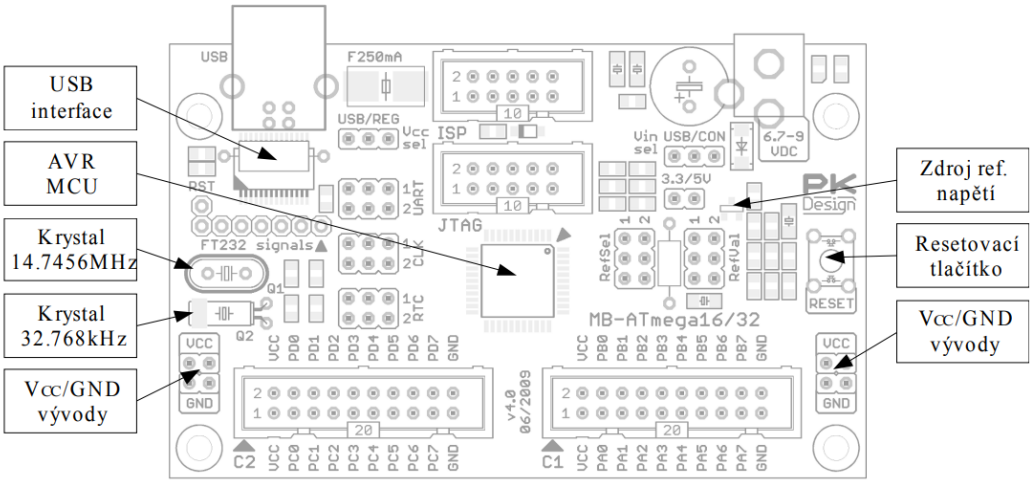
1. VOLBA DOSTUPNÝCH HW A SW PROSTŘEDKŮ VHODNÝCH PRO EMULACI FUNKČNÍHO CHOVÁNÍ ŘADIČE A PŘÍSLUŠNÝCH PERIFÉRIÍ

1.1. Volba vhodných HW prostředků pro emulaci chování křižovatky

1.1.1. Základová deska

Jako základní část systému, která celý návrh emulace SSZ ovládá jsem zvolil základovou desku ATmega32. Tuto základovou desku jsem vybral z důvodu mé zkušenosti s prací s ní, kterou jsem získal v předmětu mikroprocesorová řídící technika. Pro účely zpracování daného tématu je vhodná také z důvodu možnosti připojení externích periférií v podobě tlačítek či semaforů, které umožňují ovlivňovat a sledovat chování emulace a mají neopomenutelný demonstrační význam. Dalším z důvodů volby uvedené základové desky je i možnost přesného odměřování požadovaných časových intervalů pomocí časovačů mikrokontroleru, což umožňuje nastavování intervalu jednotlivých signálů na semaforech.

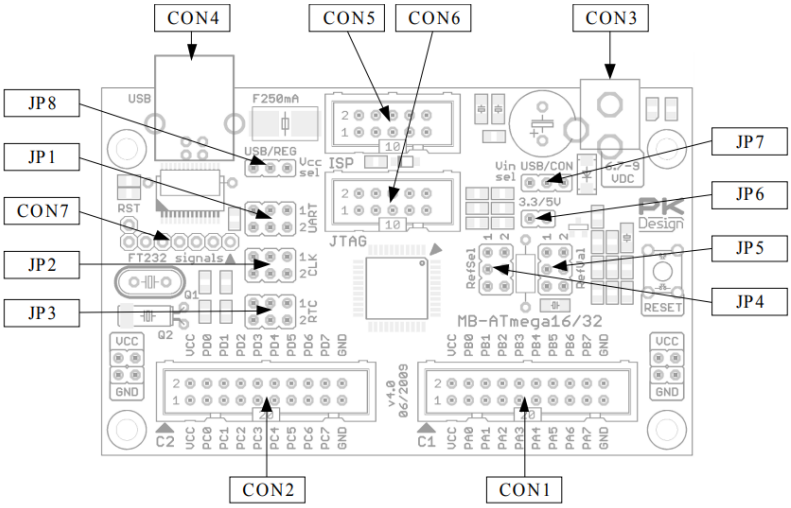
Hlavní částí této základové desky je RISC-ový mikrokontroler Atmel ATmega 32. Tento mikrokontroler se programuje přímo na základové desce připojením programovacího kabelu na jeden z konektorů ISP nebo JTAG. Napájení základové desky se připojuje přes konektor USB či POWER a pomocí napěťového regulátoru lze nastavit napájení mikrokontroleru a jeho periférií na hodnotu 3,3 V nebo 5 V. Základová deska dále disponuje konektory, na které se pomocí propojek dají připojit různé externí hardwarové periférie, které jsou následně spojené s I/O vývody mikrokontroleru. Základová deska též disponuje dvojicí krystalů, které slouží jako zdroj hodinového signálu mikrokontroleru. První krystal 14,7456 MHz a druhý odpojitelný krystal 32,768 kHz pro obvod hodin reálného času mikrokontroleru (PK Design, 2011, str. 4).



Obrázek 1 – Základová deska ATmega32

Zdroj: (PK Design, 2011, str. 5)

Obrázek 1 zobrazuje základovou desku ATmega32. Jsou zde popsány její jednotlivé části. Uprostřed základové desky se nachází AVR mikrokontroler. Po levé straně jsou umístěny oba krystaly s různými frekvencemi jako zdroj hodinového signálu využívané v časovačích mikrokontroleru. Pod nimi se nachází vývody napájení Vcc a GND, které slouží pro připojení napájení různých periférií. Na pravé straně základové desky se nachází stejná dvojice napájecích vývodu Vcc a GND. Nad nimi je resetovací tlačítko, pomocí kterého se dá mikrokontroler kdykoliv ručně restartovat. Dále je na základové desce umístěn zdroj napěťové reference pro interní AD převodník mikrokontroleru.



Obrázek 2 – Konektory základové desky ATmega32

Zdroj: (PK Design, 2011, str. 7)

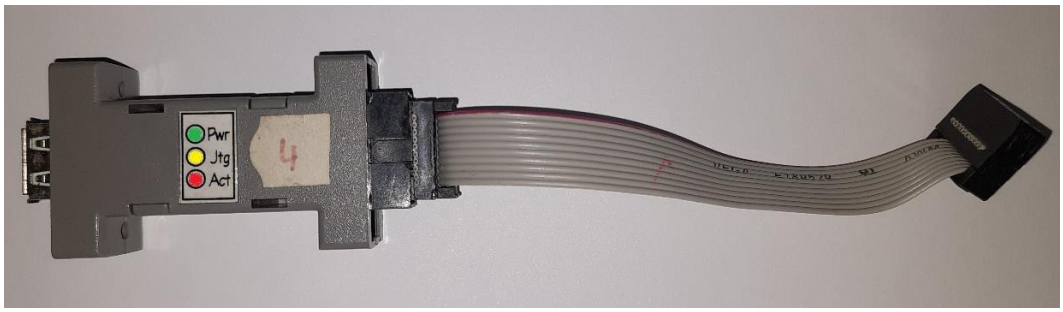
Obrázek 2 popisuje všechny konektory umístěné na základové desce. V dolní části se nacházejí konektory CON1 a CON2. Každý z této dvojice konektorů obsahuje 16 vstupně/výstupních pinů připojených k mikrokontroleru. Dále oba konektory disponují 2 piny s napájecím napětím Ucc (piny 1 a 2) a 2 piny označené GND připojené na nulový potenciál (piny 19 a 20). Konektor CON3 slouží jako jeden z možných napájecích konektorů. Druhá možnost napájení je pak pomocí konektoru CON4 typu USB - B, pomocí kterého lze základovou desku napájet například z počítače. Dvojice konektorů CON5 a CON6 označených ISP a JTAG slouží pro připojení programovacího kabelu UniProg-USB z počítače. Pomocí programovacího kabelu se do mikrokontroleru nahrává kód programu, podle kterého následně mikrokontroler ovládá své výstupy s periferiemi (PK Design, 2011, str. 7). Poslední konektor CON7 slouží pro připojení doplňkových signálů obvodu FT232RL (PK Design, 2011, str. 13).

Piny s označením JP slouží pro volbu různých nastavení základové desky. Nastavování se uskutečňuje pomocí propojek, kterými se propojí příslušné piny, dle návodu k použití k základové desce ATmega32. Volí se zde třeba velikost napětí napájení základové desky a přídavných periférií, zdroj hodinového signálu pro mikrokontroler a časovače či různá další nastavení, která jsou uvedena v návodu k použití k základové desce ATmega32.

1.1.2. Programovací kabel UniProg-USB

Pro připojení mikrokontroleru k počítači a jeho následné programování jsem použil programovací kabel UniProg-USB. Jeho výhoda je, že umožňuje snadné připojení mikrokontroleru přímo k PC do USB portu a zároveň obsahuje všechen potřebný HW pro naprogramování mikrokontroleru.

Hardware kabelu UniProg-USB je umístěn v krabičce redukce CAN9-CAN9. Z jedné strany má kabel USB konektor typu a pro připojení k počítači, z druhé strany pak konektor MLW10, kterým se UniProg-USB připojuje k základové desce mikrokontroleru. Na horní straně krabičky programovacího kabelu jsou pod průsvitným štítkem umístěny 3 LED indikační diody, viz obrázek 3. Horní dioda je zelená a svitem indikuje připojené napájecí napětí. Druhá dioda je žlutá a svitem indikuje že je aktivní JTAG firmware nebo, v případě že nesvití, tak je aktivní Virtual processor firmware. Poslední dioda je červená a svým svitem indikuje aktivitu programovacího kabelu, například při programování připojeného mikrokontroleru (PK Design, 2007a, str. 6).



Obrázek 3 – Programovací kabel UniProg-USB

Programovací kabel UniProg-USB se skládá z několika hardwarových částí, viz obrázek 4 s blokovým schématem programovacího kabelu. První částí tvoří USB konektor typu a pro připojení programovacího kabelu k počítači. Následuje integrovaný obvod FT232 převádějící USB na UART-TTL. Další částí je galvanické oddělení, které zajišťuje oddělení programované aplikace od PC pro zajištění ochrany PC. Po galvanickém oddělení následuje mikrokontroler mega16L obsahující řídící program (firmware) neboli jádro programátoru. Poslední část je výstupní konektor MLW10, který se připojuje k programované aplikaci, v případě sestavy pro realizaci předmětné emulace do konektoru CON5 na základové desce mikrokontroleru AVR (PK Design, 2007a, str. 6).

Obrázek 4 – Hardwarové části UniProg-USB

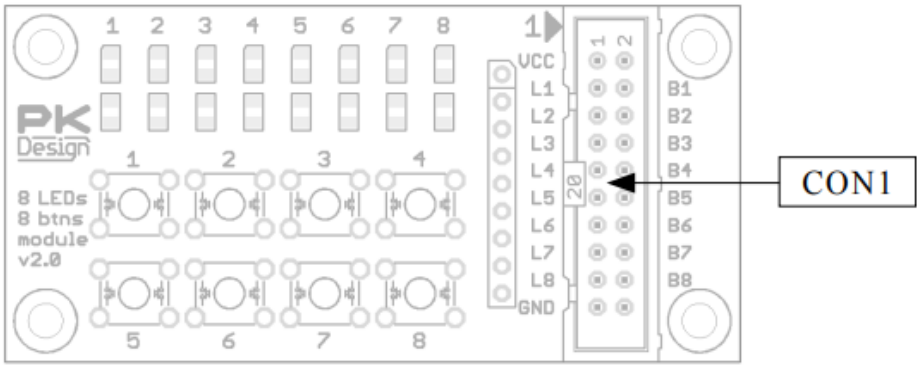
Zdroj: (PK Design, 2007a, str. 6)

1.1.3. Modul s 8 LED a tlačítky

Modul s 8 LED a 8 tlačítky slouží jako emulace tlačítek na přechodech pro chodce. Tento modul jsem zvolil z důvodu jeho kompatibility se zvoleným mikrokontrolerem. Jedna z jeho hlavních výhod je kombinace LED a tlačítek (viz obrázek 5), což umožňuje emulovat přesné chování skutečných chodeckých tlačítek, u kterých se po stisku rozsvítí indikace jeho zaregistrování. Tlačítka modulu lze také využít pro další vstupy do programu (konkrétní využití je popsáno dále v textu práce).

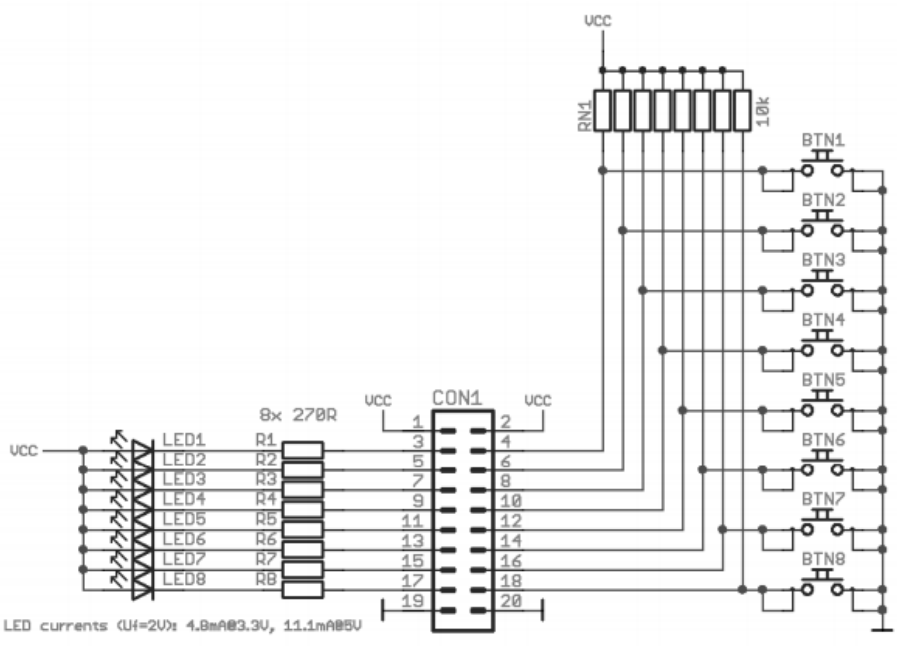
Tlačítka a LED diody jsou vyvedeny na konektor MLW20 (CON1), pomocí něhož se dají připojit k základové desce. Modul slouží pro zobrazování 8-bitových binárních signálu

pomocí LED diod, či k nastavování jiného 8-bitového binárního signálu pomocí jednotlivých tlačítek, kdy každé tlačítko představuje jeden bit. „Napájecí napětí tohoto modulu Ucc je možné volit v rozsahu 0 - 5 V“ (PK Design, 2007b, str. 6).



Obrázek 5 – Modul s 8 LED a 8 tlačítky

Zdroj: (PK Design, 2007b, str. 6)



Obrázek 6 – Zapojení modulu tlačítek a LED

Zdroj: (PK Design, 2007b, str. 10)

Obrázek 6 zobrazuje schéma zapojení tlačítek i LED diod. Anody LED diod jsou připojeny na napájecí napětí Ucc. Jejich katody jsou pak připojeny přes ochranné rezistory na piny označené lichými čísly na konektoru CON1. Pro to aby se rozsvítila požadovaná dioda, se musí nastavit příslušný pin konektoru CON1, ke kterému je dioda připojená, na napětí úrovně,

která představuje logickou nulu. LED diody jsou tedy zapojeny v obrácené logice, kdy při napětí, které se rovná logické úrovni jedna, LED dioda nesvítí a při napětí rovnajícímu se logické úrovni nula dioda svítí. Piny na základové desce, ke kterým se LED diody připojují, jsou pak nastaveny jako výstupní právě z důvodu, aby šlo měnit úroveň jejich napětí, které představuje jednu z logických úrovní. Napětí pro logické úrovně vypadá následovně, Ucc = 5 V představuje logickou úroveň jedna a napětí Ucc = 0 V představuje logickou úroveň nula.

Na pravé straně obrázku je pak vidět připojení tlačítek 1- 8. Tlačítka jsou na jedné straně připojena přes ochranné odpory R =10 kΩ na napájecí napětí Ucc = 5 V. Druhá strana tlačítek je připojena na GND. V případě, že dojde ke stisku jednoho z tlačítek, uzavře se tak elektrický obvod přes toto tlačítko na GND a na pinu konektoru CON1 se objeví nulové napětí, které představuje logickou úroveň nula. Mikrokontroler pak dle nastavení programu čte napěťové úrovně na jednotlivých pinech na základové desce, ke kterým jsou z konektoru CON1 připojena všechna tlačítka. Při čtení se z aktuálních logických úrovní pinů vytvoří 8-bitové číslo, kdy jednotlivé bity představují stav tlačítek. Bit, který se rovná 1, představuje nestisknuté tlačítko a bit, který se rovná 0 představuje tlačítko stisknuté. Logické úrovně jsou zde tedy obrácené oproti klasickému předpokladu.

1.1.4. LED světelné signalizační zařízení pro vozidla

Pro emulaci semaforů na křižovatce jsem zvolil modul LED semaforu (viz obrázek 7). Tento modul je vhodný pro emulaci semaforu, jelikož se skládá ze 3 barevných LED červené, žluté a zelené, neboli stejných světel jako na skutečném návěstidel SSZ, tj. semaforu. Vhodný je tento modul i z důvodu jeho kompatibility se zvolenou základovou deskou, pomocí které lze tento modul napájet a zároveň ovládat, svícení LED osazených na modulu.

Modul je opatřen i předřadnými odpory pro jednotlivé LED diody. Na spodní části modulu se pak nachází 4 připojovací piny. Jsou označeny jako G (green = zelená), Y (yellow = žlutá), R (red = červená) a poslední společný pin GND. Na jednotlivé piny G, Y a R, které představují anody diod, se připojuje kladné napájecí napětí 5 V. Dle požadavků na svícení jednotlivých LED diod se pak toto napětí přepíná mezi napěťovou úrovní přestavující logickou úroveň nula pro stav, kdy diody nesvítí a pro napěťovou úroveň představující logickou úroveň jedna a tedy stav, kdy LED diody svítí. Poslední pin na modulu je pin označený GND, ten se připojuje na nulový potenciál a jsou s ním spojeny katody všech tří LED na semaforu. Pin s nulovým potenciálem je na základové desce označený stejně jako GND.

Obrázek 7 – Modul LED semaforu

Pro zelenou LED diodu je nutné navrhnout vhodnější předřadný odpor, jelikož výrobce zde použil nulový předřadný odpor, což by způsobilo, že by vedlo k příliš velkému proudu protékajícímu diodou. Pro požadovanou velikost předřadného odporu platí obecně známý vzorec 1.1.4 – 1.

$$R = \frac{U_{CC} - U_{LED}}{I_{LED}} = \frac{5-2,6}{0,02} = 120\ \Omega \qquad (1.1.4 - 1)$$

Kde:

Ucc je napájecí napětí o velikosti, Ucc = 5 V

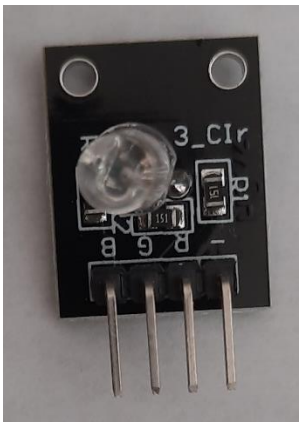
U_{LED} je požadovaný úbytek napětí pro zelenou LED, U_{LED} = 2,6 V

I_{LED} je požadovaný proud LED diodou v propustném směru, I_{LED} = 0,02 A

Po dosazení vychází velikost předřadného odporu 120 Ω.

1.1.5. LED světelné signalizační zařízení pro chodce

Jako emulaci chodeckého semaforu na křižovatce jsem zvolil modul RGB diody (viz obrázek 8). Tento modul se skládá ze speciální RGB diody, předřadných odporů diody a pinů, pomocí kterých se k modulu připojuje napájení. Teno modul jsem opět zvolil důvodu kompatibility se zvolenou základovou deskou, ze které budu modul napájet a ovládat svit a jeho barvu.



Obrázek 8 – Modul RGB diody

RGB dioda, jak už název napovídá, umožňuje svit 3 barvami, R jako red , G jako green a B jako blue. Protože že se dioda v podstatě skládá ze tří různě barevných diod, jsou u ní vyvedeny celkem 4 piny. První 3 piny označené písmeny R, G, a B jsou připojeny každý k jedné anodě diody přes předřadný rezistor. Připojuje se na ně tedy kladné napětí 5 V. V případě předmětné emulace jsou využity pouze piny pro LED diody červené a zelné barvy, protože modul je využit jako chodecký semafor, na kterém se nachází pouze dvě světla (červený symbol chodce, zelený symbol chodce). Poslední pin označený jako mínus či zkratkou GND je pak společný pin katod všech tří LED diod a připojuje se na nulový potenciál, který je na základové desce označen jako GND.

1.2. Volba vhodných SW prostředků pro emulaci chování křižovatky

1.2.1. Vývojové prostředí Atmel Studio

Jako vývojové prostředí pro tvorbu programu emulujícího chování křižovatky jsem zvolil program Atmel studio. Jedním z hlavních důvodů této volby, jsou mé zkušenost s tímto programem, které jsem získal během výuky předmětu Mikroprocesorová řídící technika. Dalším z důvodů byla také dobrá dostupnost daného vývojového programu pro účely mé práce.

Atmel studio je integrované vývojové prostředí pro vytváření aplikací v jazyce C pro mikrokontrolery AVR. Výhoda tohoto programovacího studia je, že obsahuje debugger. Ten umožňuje programátorovi simulovat vytvořený kód a ověřit tak jeho základní funkčnost. Simulací je Debugger schopen najít méně závažné i závažnější chyby v programu, které by znemožňovali jeho funkčnost a programátora na ně upozorní vypsáním chybové hlášky error s krátkým popisem chyby a přímým odkazem na řádek, ve kterém se chyba vyskytuje (viz obrázek 16). To usnadňuje programátorovi detekci a odstranění této chyby. Jednou z výhod

Debuggeru je i možnost pomocí tzv. breaking points simulovat pouze programátorem zvolené části programu či simulovat kód programu řádek po řádku. To umožňuje testování funkčnosti dílčích částí programu. To je vhodné například, když kód programu ještě není celý dopsán, což znemožňuje jeho faktické spuštění. Další popis daného programu je uveden v příloze 1 (viz str. 72).

1.2.2. Programování přes ISP rozhraní

Po vytvoření požadovaného kódu, odladění chyb a úspěšného sestavení a nasimulování pomocí Debuggeru je možné kód nahrát do mikrokontroleru AVR. Pro toto nahrání jsem zvolil program AVR ISP Programmer od firmy PK Design. Tento software jsem vybral, protože s ním mám zkušenosti z předmětu Mikroprocesorová řídící technika a též z důvodu dobré dostupnosti daného programu.

Mikrokontroler AVR, do kterého je třeba nahrát vytvořený kód, se připojí k počítači pomocí programovacího kabelu UniProg-USB. Ten je osazen z jedné strany USB konektorem typu A, pro připojení k počítači a z druhé strany konektorem MLW10 pro připojení do konektoru ISP na základové desce mikrokontroleru. Popis programování mikrokontroleru pomocí daného programu je uveden v příloze 2 (viz str. 75).

2. SPECIFIKACE POŽADOVANÉHO EMULOVANÉHO FUNKČNÍHO CHOVÁNÍ A DALŠÍCH SOUVISEJÍCÍCH VLASTNOSTÍ

Řízení SSZ v praxi zajišťuje řadič, jehož hlavní činností je vykonávání tzv. signálního plánu. Ten definuje kombinace všech možných světelných signálů na jednotlivých semaforech křižovatky, jejich pořadí či délku trvání. Tyto parametry souvisejí především s podobou křižovatky, jejím dopravním kontextem a často i aktuální dopravní situací a požadavcích účastníků provozu (typicky chodců). Signální plán, podmínky k jeho sestavení a implementaci do konkrétního řadiče (resp. do řadiče konkrétní křižovatky) – potažmo do emulace řadiče – tak představují i hlavní informační bázi pro identifikace požadavků a jejich specifikaci na funkční vlastnosti (emulace) řadiče.

2.1. Popis signálního plánu

„Signální plán je program řízení světelného signalizačního zařízení, který určuje pořadí a délku signálů volno jednotlivých signálních skupin“ (TP 81, 2015, str. 22). Základ pro vytvoření signálního plánu je sestavení fázových schémat dané křižovatky, stanovení délky cyklu řízení a stanovení doby jednotlivých fází na světelném signalizačním zařízení.

Se signálním plánem, jeho tvorbou i implementací do řadiče jsou úzce spjaty následující základní pojmy (TP 81, 2015, str. 7–8).

- **SSZ** je zkratka pro pojem „světelné signalizační zařízení“.
- **Signální skupina** je soubor návěstidel, která udávají v každém okamžiku pro jeden vjezd vozidel nebo vstup chodců na jeden přechod stejný signální obraz. Signální skupinu může tvořit i jediné návěstidlo.
- **Stopčára** je vodorovná dopravní značka č. V 5 Příčná čára souvislá nebo č. V 6a Příčná čára souvislá se symbolem „Dej přednost v jízdě!“ nebo č. V 6b Příčná čára souvislá s nápisem STOP.
- **Signál stůj** je takový signál podle vyhlášky č. 30/2001 Sb., který zakazuje účastníkovi provozu na pozemních komunikacích vstup či vjezd (pokračovat v jízdě) do křižovatky.

- **Signál volno** je takový signál podle vyhlášky č. 30/2001 Sb., který umožňuje účastníkovi provozu na pozemních komunikacích vstup či vjezd (pokračovat v jízdě) do uzlu.
- **Signál pozor** má dvě varianty zobrazení s odlišnými významy. (TP 81, 2015, str. 14)
    - **Signál se současně svítícím červeným a žlutým světlem** znamená povinnost připravit se k jízdě.
    - **Signál se žlutým světlem** znamená povinnost zastavit vozidlo před stopčárou, a kde stopčára není vyznačena, před SSZ. Je-li však vozidlo při rozsvícení tohoto signálu již tak blízko, že by řidič nemohl vozidlo bezpečně zastavit, smí pokračovat v jízdě. Bezpečné opuštění křižovatky se pro takový případ zohledňuje v signálním plánu křižovatky.
- **„Fáze** je část cyklu během kterého mají určité jízdní proudy současně signál volno“ (Rábek, 2015, str. 121).
- „**Mezičas** je časový interval od konce signálu volno jedné signální skupiny po začátek signálu volno jiné kolizní signální skupiny“ (TP 81, 2015, str. 27).

2.2. Popis výpočtu mezičasu křižovatky

Mezičasy představují jeden z hlavních konfiguračních parametrů řadiče křižovatky přímo určující bezpečnost na křižovatce v době jejího řízení světelnými signály.

Na každé křižovatce se nacházejí kolizní plochy (body), které představují křížení dopravních proudů. Mezičas se stanovuje tak, aby poslední (vyklízející) vozidlo, které najede do křižovatky v době končící zelené, mělo dostatečný čas pro bezpečné opuštění (vyklizení) kolizní plochy dříve, než (najíždějící) vozidlo z kolizního směru dosáhne této plochy (Smělý, 2007, str. 23). Při nedostatečně stanovených mezičasech by mohlo dojít ke kolizi vozidla vyklízejícího křižovatku s vozidlem najíždějícím do křižovatky během následující fáze.

Pro všechny signální skupiny v kolizních směrech se tedy musí vypočítat požadované mezičasy. V případě že signální skupina řídí více dopravních proudů, čímž vzniká i více kolizních ploch, vybírá se vždy nejdelší mezičas vypočítaný pro tyto proudy. Obecně má pak určení doby mezičasu zásadní význam z hlediska bezpečnosti při řízení provozu světelnými signály a je proto nezbytné věnovat jejich stanovení maximální pozornost (TP 81, 2015 str. 27).

Při výpočtu mezičasů jsou využívány standardní hodnoty (viz tabulka 1 a tabulka 2). Tyto hodnoty jsou použity i při výpočtech jednotlivých mezičasů předmětné emulace.

Tabulka 1 – Standardní hodnoty rychlostí pro výpočet mezičasů

(TP 81, 2015, str. 161)

VYKLIZOVACÍ A NAJÍŽDĚCÍ RYCHLOSTI	V_v a V_n [km.h^{-1}]	[m.s^{-1}]
MOTOROVÁ VOZIDLA		
v přímém směru	35	9,7
v oblouku	25	7,0
CYKLISTÉ	15	4,2
CHODCI	5	1,4
TRAMVAJE		
v místech výhybek při jízdě proti hrotům do odbočné větve výhybky	10	2,8
v místech výhybek a kolejových křižovatek (s výjimkou jízdy proti hrotům do odbočné větve výhybky)	15	4,2
v úsecích bez kolejových konstrukcí v obloucích o poloměru menším než 25 m	15	4,2
v úsecích bez kolejových konstrukcí v obloucích o poloměru 25 až 60 m	20	5,6
v úsecích bez kolejových konstrukcí v obloucích o poloměru 61 až 100 m	25	7,0
v úsecích bez kolejových konstrukcí v obloucích o poloměru větším než 100 m a v přímých směrech:		
najížděcí rychlost	25	7,0
vyklizovací rychlost	35	9,7

Tabulka 2 – Standardní hodnoty délky vozidel a hodnoty bezpečnostní doby

(TP 81, 2015, str. 161)

DÉLKA VYKLIZUJÍCÍHO VOZIDLA A BEZPEČNOSTNÍ DOBA	l_{voz} [m]	t_b [s]
MOTOROVÁ VOZIDLA	5	2
CYKLISTÉ	0	1
CHODCI	0	0
TRAMVAJE	15	
pro vyklizovací rychlost 1–15 km/h		0
pro vyklizovací rychlost 16–20 km/h		1
pro vyklizovací rychlost 21–25 km/h		2
pro vyklizovací rychlost 26 a více km/h		3

Výpočty mezičasů pro jednotlivé kolizní plochy se stanovují dle rovnice 2.2 – 1 (TP 81, 2015, str. 157).

$$t_m = t_v - t_n + t_b \qquad (2.2-1)$$

Tato rovnice má celkem tři parametry t_v, t_n a t_b. Hodnota t_m [s] představuje vypočtený výsledný mezičas pro danou kolizní plochu. Parametr t_v [s] se označuje jako vyklizovací doba. Tato doba určuje minimální požadovaný časový interval, který vyklízející vozidlo potřebuje na to, aby se od projetí stopčáry dostalo celou svojí délkou za stanovenou kolizní plochu. Tento časový interval se stanovuje stejně i v případě chodce, kdy se jedná o dobu od vstupu chodce na přechod až po dobu, kdy dojde za stanovenou kolizní plochu. Parametr t_n [s] naopak představuje najíždějící dobu vozidla. Jedná se o minimální časový interval od projetí stopčáry vozidlem, až po jeho dosažení začátku kolizní plochy. Parametr t_b [s], neboli bezpečnostní doba zohledňuje průjezd vozidla stopčárou během signálu „Pozor!“ po skončení signálu volno a jeho hodnota je pevně stanovená, viz tabulka 2. (TP 81, 2015, str. 157)

Jak už bylo řečeno, první parametr z rovnice 2.2 – 1 je parametr vyklizovací doby t_v [s]. Rovnice pro výpočet vyklizovací doby vozidla vypadá následovně (TP 81, 2015, str. 157).

$$t_v = \frac{L_v + l_{voz}}{v_v} \qquad (2.2 - 2)$$

Proměnná Lv [m] v rovnici 2.2 – 2 představuje tzv. vyklizovací dráhu. Jedná se o úsek od stopčáry, kterou vozidlo projelo až za konec kolizní plochy. U chodců pak vzdálenost od vstupu chodce na přechod až na konec kolizní plochy. Tento parametr se udává v metrech a je závislý pouze na geometrii vozovky, resp. trajektorii pohybu vozidel v křižovatce, ze které se určuje požadovaná vyklizovací dráha.

Parametr l_{voz} [m] představuje délku vyklizujícího vozidla. Pro standardní výpočet mezičasů je standardní délka vyklizujícího vozidla v případě motorového vozidla stanovená na 5 m. Z této hodnoty je vidět, že se na dané křižovatce předpokládá provoz pouze osobních automobilů, jelikož pro nákladní by byla tato hodnota větší a bylo by nutné vypočítané hodnoty mezičasů přepočítat. Pro tramvaj je tato hodnota 15 m a pro cyklisty či chodce má hodnotu 0 m (viz tabulka 2).

Poslední parametr v_v [m/s], z rovnice 2.2 – 2, představuje rychlost vyklízejícího vozidla či chodce uvedenou vždy v m/s. Z tabulky pro standardní výpočet mezičasů se lze opět dozvědět standardní hodnoty tohoto parametru (viz tabulka 1). Pro motorová vozidla se uvažují rychlosti v závislosti na směru jízdy vozidla. První případ je jízda v přímém směru, pro tu je obecná vyklizovací či najížděcí rychlost stanovena na 9,7 m/s. Druhá možnost je, že dráha motorového vozidla vede do oblouku a v tomto případě tabulka stanovuje rychlost 7 m/s. Pro

chodce je stanovená pevná rychlost a to vždy 1,4 m/s. To stejné i v případě cyklistů, pro ty je stanovena pouze jedna rychlost a to 4,2 m/s nehledě na směr jejich jízdy. Pro tramvaje tabulka specifikuje hned několik rychlostí, dle konstrukce kolejí, po kterých se pohybuje. Například pro přímý směr či oblouk o poloměru větším jak 60 m je rychlost stanovena na 7 m/s. Ale například pro oblouky o menším poloměru jak 25 m je výpočetní rychlost stanovena na 4,2 m/s. Zvláštní rychlost je u tramvají stanovena i pro jízdu do oblouku současně s jízdou proti hrotu výhybek, zde tabulka udává rychlost 2,8 m/s.

Dalším parametrem z rovnice 2.2 – 1 je najížděcí doba t_n [s]. Tento parametr představuje minimální časový interval, od projetí stopčáry až po dosažení kolizní plochy najíždějícím vozidlem. Či v případě chodců časový interval od vstupu chodce do křižovatky až po dosažení kolizní plochy. Rovnice pro určení najížděcí doby vypadá následovně (TP 81, 2015, str. 157).

$$t_n = \frac{L_n}{v_n} \qquad (2.2 - 3)$$

Proměnná L_n v této rovnici 2.2 – 3 představuje dráhu najíždějícího vozidla od stopčáry až po začátek místa kolizní plochy v křižovatce. Určuje se tedy změřením dráhy mezi stopčárou a kolizní plochou. Druhý parametr rovnice 2.2 – 3 je pak rychlost najíždějícího vozidla v_n [m/s]. Obdobně jako pro rychlost vyklizovací v rovnici 2.2 – 2 se zde hodnota rychlosti stanovuje dle druhu dopravního prostředku a směru jeho pohybu. Například pro motorové vozidlo v přímém směru je standardní hodnota rychlosti 9,7 m/s.

Posledním parametrem z rovnice 2.2 – 1 je tzv. bezpečnostní doba t_b [s]. Tento parametr zohledňuje možnost najetí vozidla do křižovatky během signálu „Pozor!“ po skončení zelené fáze. Dle tabulky se standardními hodnotami pro výpočet mezičasů má parametr t_b [s] velikost pro motorová vozidla 2 s, pro cyklisty 1 s a pro tramvaje i chodce 0 s (viz tabulka 2). Pro tramvaje i chodce se uvažuje hodnota 0 s z důvodu toho, že nemají signál „Pozor!“, během kterého by mohli najet/vstoupit do křižovatky, jelikož v té době jim již svítí signál stůj.

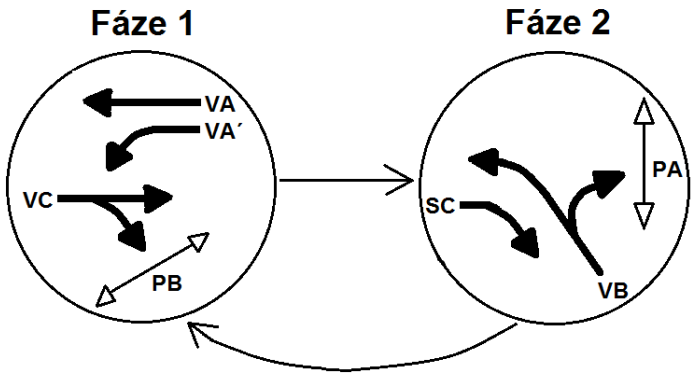
2.3. Popis tvorby signálního plánu křižovatky

Signální plán jsem vytvářel pro předem zvolenou křižovatku, situační schéma dané křižovatky je vidět v příloze 3 (viz str. 76). Při tvorbě signálního plánu jsem postupoval dle TP 81 pro navrhování světelných signalizačních zařízení pro řízení silničního provozu na pozemních komunikacích.

2.3.1. Určení počtu a pořadí fází

Počet fází vyplývá z rozdělení fází neboli z rozhodnutí o rozčlenění dopravních pohybů na dané křižovatce. Pro řízení dopravy SSZ na křižovatce je stanoven dle technických podmínek pro navrhování světelných signalizačních zařízení minimální počet fází na dvě. V tomto případě však nejsou odbočující dopravní proudy v porovnání s přímými bezkolizní. Pro dosažení dokonalého bezkolizního řízení by pro tříramennou křižovatku bylo nutné vytvořit minimálně tři fáze. Přednostně se však obecně požaduje vytvořit jednoduché fázové schéma. Při větším počtu fází dochází totiž ke snižování kapacity křižovatky a k prodloužení čekání vozidel, což může vést ke tvorbě kolon na příjezdech ke křižovatce. Z těchto důvodů je vhodné omezovat navrhované počty fází na minimální nezbytnou míru dle intenzit jednotlivých dopravních proudů (TP 81, 2015, str. 25).

V uvedeném případě jsem navrhoval signální plán pro tříramennou křižovatku pouze s jedním odbočením vlevo, které je pak kolizní s přímým dopravním proudem. Pro toto odbočení vlevo jsem pracoval s předpokladem menší intenzity vozidel a z tohoto důvodu jsem nakonec zvolil pouze dvoufázové schéma. Na následujícím obrázku (viz obrázek 9) jsou vidět fázová schéma, která jsem vytvořil pro řízení dané křižovatky a která jsou dále použita pro sestavení signálního plánu.



Obrázek 9 – Schéma a pořadí fází

Stanovení pořadí fází je obvykle závislé na logickém navazování fází, kdy některé fáze musejí probíhat za sebou, pro plynulé navazování signálů volno. Také je důležité volit vhodné pořadí jízdních směrů, aby například nedocházelo k hromadění vozidel v křižovatce. Důležitým parametrem je však u navrhování pořadí fází minimalizace mezičasů křižovatky, což vede ke

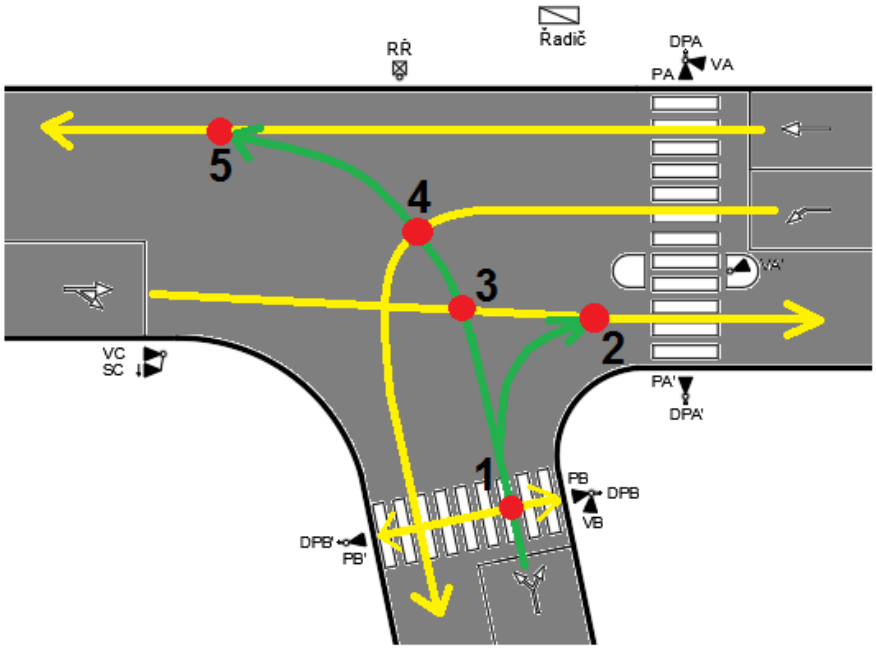
zvýšení kapacity křižovatky. Volbou minimální hodnoty mezičasu však nesmí dojít k porušení některých z předchozích podmínek.(TP 81, 2015, str. 25)

Jelikož jsem pro danou křižovatku stanovil pouze 2 fáze, nemusím řešit pořadí, ve kterém budou spínány jednotlivé fáze. Nicméně v uvedeném případě bude potřebné stanovit počáteční fázi po náběhu křižovatky do řízeného stavu. V tomto případě bude vhodné u dané křižovatky začínat zelenou fází v hlavním směru, ve kterém je přepokládána i vyšší intenzita vozidel. Následně po první fázi v hlavním směru bude následovat fáze ve vedlejším směru a dále už bude docházet k periodickému střídání obou fází.

2.3.2. Výpočet mezičasů pro vedlejší směr

Jak už bylo zmíněno v kapitole 2.2 je parametr mezičasů důležitý z hlediska bezpečnosti provozu na křižovatce. Proto jsem tento parametr potřeboval stanovit i pro emulaci řízení křižovatky. Výchozím bodem je schéma fází zvolené křižovatky, které je určující pro počet kolizních bodů na dané křižovatce. Pro všechny tyto body jsem následně vypočítal potřebné mezičasy.

Dále je proveden výpočet doby mezičasu pro stav cyklu, kdy končí fáze v hlavním („vodorovném“) směru a začíná fáze ve vedlejším směru („svislém“). Neboli, když křižovatku vyklízí vozidla, která doposud jela v hlavním směru a do křižovatky začínají najíždět vozidla z vedlejšího směru.



Obrázek 10 – Kolizní body pro vedlejší směr

Možné dráhy jízdy vozidla z vedlejšího směru jsou v obrázku zvýrazněny zelenou barvou (viz obrázek 10). Žlutou barvou jsem naznačil jízdní dráhy vozidel a chodců z přechozí zelené fáze. Tam kde se zelené jízdní dráhy najíždějících vozidel protínají se žlutými dráhami vyklízejících vozidel nebo chodců jsou kolizní body (zvýrazněné červeně a označeny číslem). Pro všechny tyto body se musí stanovit hodnoty mezičasů.

Výpočet mezičasu kolizního bodu číslo 1 (přechod pro chodce)

První kolizní bod najíždějícího vozidla je na přechodu pro chodce. Výpočet mezičasu pro bezpečné vyklizení křižovatky chodcem je následující. Do vzorce pro vyklizovací dobu se dosadí délka přechodu, v tomto případě 6,5 m a za vyklizovací rychlost 1,4 m/s, což je dle tabulky pro standardní výpočet mezičasů rychlost chodce. Pro výpočet najížděcí doby jsem uvažoval vzdálenost přechodu pro chodce od stopčáry 1,5 m, což je nejmenší dovolená vzdálenost přechodu od stopčáry dle technických podmínek (TP 81, 2015, str. 21). Rychlost najíždějícího vozidla jsem pak uvažoval 7 m/s, jelikož jeho dráha vede do oblouku.

Výpočet vyklizovací doby chodce:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{6{,}5+0}{1{,}4} = 4{,}64\ s \qquad (2.3.2-1)$$

Výpočet najížděcí doby vozidla:

$$t_n \frac{L_n}{v_n} = \frac{1{,}5}{9{,}7} = 0{,}16\ s \qquad (2.3.2-2)$$

Následně jsem obě tyto hodnoty dosadil do vzorce pro finální výpočet požadovaného mezičasu pro tento kolizní bod. Za parametr t_b jsem dosadil 0 s, jelikož se jedná o chodce, který nemůže vstoupit do křižovatky během signálu „Pozor!“, jelikož už v té době bude mít červený signál.

Výpočet finální hodnoty mezičasu pro kolizní bod č. 1:

$$t_m = t_v - t_n + t_b = 4{,}64 - 0{,}16 = 4{,}48\ s \qquad (2.3.2-3)$$

Výpočet mezičasu kolizního bodu číslo 2

Tento kolizní bod představuje křížení jízdní dráhy vyklízejících vozidel jedoucích rovně v hlavním směru zleva doprava a vozidel najíždějících do křižovatky z vedlejšího směru odbočujících doprava.

Vyklízející dráha vozidla je v tomto případě 20 m a rychlost pro vozidla jedoucích v přímém směru je stanovena na 9,7 m/s. Pro vozidlo najíždějící do křižovatky je dráha od stopčáry ke koliznímu bodu 10 m a najížděcí rychlost do oblouku 7 m/s.

Výpočet vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{20+5}{9,7} = 2,58\ s \qquad (2.3.2 - 4)$$

Výpočet najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{10}{7} = 1,43\ s \qquad (2.3.2 - 5)$$

Pro výpočet finální hodnoty mezičasu jsem opět dosadil tyto vypočítané hodnoty a za parametr bezpečnostní doby t_b jsem dosadil 2 s pro zohlednění možného projíždění signálu „Pozor!“ vyklízejícím vozidlem.

Výpočet finální hodnoty mezičasu pro kolizní bod č. 2:

$$t_m = t_v - t_n + t_b = 2,58 - 1,43 + 2 = 3,15\ s \qquad (2.3.2 - 6)$$

Výpočet mezičasu kolizního bodu číslo 3

Třetí kolizní bod představuje křížení jízdní dráhy vozidel jedoucích v hlavním směru rovně zleva doprava a jízdní dráhy najíždějících vozidel z vedlejšího směru odbočujících doleva (viz obrázek 10).

Vyklizovací dráhu od stopčáry za kolizní bod jsem ze schéma křižovatky pro tento směr stanovil na 16 m. Dle tabulky pro obecný výpočet mezičasu jsem pak dosadil délku motorového vozidla 5 m a vyklizovací rychlost pro motorové vozidlo v přímém směru 9,7 m/s. Při stanovení najížděcí doby jsem za rychlost najížděcího vozidla dosadil 7 m/s z důvodu jízdy do oblouku a délku dráhy jsem dle situačního schématu křižovatky stanovil na 10 m.

Vyklizovací doba vozidla:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{16+5}{9{,}7} = 2{,}17\ s \qquad (2.3.2-7)$$

Najížděcí doba vozidla:

$$t_n \frac{L_n}{v_n} = \frac{10}{7} = 1{,}43\ s \qquad (2.3.2-8)$$

Vypočítané hodnoty jsem následně dosadil do vzorce pro finální výpočet mezičasu. V tomto případě jsem opět za parametr t_b dosadil 2 s, zohledňující možnost projetí vozidla na signál „Pozor!“, po skončení zeleného signálu.

Výpočet finální hodnoty mezičasu pro kolizní bod č. 3:

$$t_m = t_v - t_n + t_b = 2{,}17 - 1{,}43 + 2 = 2{,}74\ s \qquad (2.3.2-9)$$

Výpočet mezičasu kolizního bodu číslo 4

Čtvrtý kolizní bod představuje křížení jízdní dráhy vozidel najíždějících z vedlejšího směru, která odbočující doleva s jízdní drahou vozidel vyklízejících, která odbočující z pravé strany křižovatky v hlavním směru doleva.

U tohoto kolizního bodu při odbočování vozidel doleva dochází k zastavování najíždějících vozidel v křižovatce z důvodu dávání přednosti v jízdě protijedoucím vozidlům. To způsobuje jejich následný delší vyklizovací čas. Dle technických podmínek pro navrhování SSZ je pro zajištění bezpečného opuštění prostoru křižovatky těmito vozidly nutné prodloužit vypočítanou hodnotu mezičasu o 2 – 4 s, dle intenzity odbočujících vozidel (TP 81, 2015, str. 29). Z tohoto důvodu jsem k vypočítané hodnotě mezičasu pro tento kolizní bod přičetl 4 s. V reálu se tyto intervaly upravují dle zjištěných provozních zkušeností.

Výpočet mezičasu jsem začal opět určením vyklízející doby. Jelikož v tomto případě bude vozidlo odbočovat, proto jsem za jeho rychlost dosadil hodnotu 7 m/s. Dráha vozidla od stopčáry za kolizní bod je 18 m. Dráha najíždějícího vozidla od stopčáry ke koliznímu bodu je 14 m, uvažovaná rychlost 7 m/s z důvodu jízdy do oblouku.

Výpočet vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{18+5}{7} = 3{,}29\ s \qquad (2.3.2-10)$$

Výpočet najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{14}{7} = 2{,}00\ s \qquad (2.3.2 - 11)$$

Vypočítané hodnoty jsem dosadil do finálního vzorce pro mezičas. Za parametr bezpečnostní doby t_b jsem dosadil 2 s, jelikož opět může dojít k najetí vozidla do křižovatky i při signálu „Pozor!“.

Výpočet finální hodnoty mezičasu pro kolizní bod č. 4:

$$t_m = t_v - t_n + t_b + 4 = 3{,}29 - 2{,}00 + 2 + 4 = 7{,}29\ s \qquad (2.3.2 - 12)$$

Výpočet mezičasu kolizního bodu číslo 5

Čtvrtý kolizní bod představuje křížení dráhy vozidla jedoucího z pravé strany v hlavním směru rovně přes křižovatku s dráhou vozidla jedoucího z vedlejšího směru a odbočujícího doleva.

Výpočet vyklizovací doby vozidla v hlavním směru od stopčáry za kolizní bod. Vyklizovací dráha od stopčáry za kolizní bod je v tomto případě 22 m a jelikož vozidlo jede v přímém směru, tak za rychlost dosazuji 9,7 m/s. Pro najíždějící vozidlo jsem dle situačního schéma určil najížděcí dráhu vozidla na 19 m a uvažovanou najížděcí rychlost vozidla 7 m/s, z důvodu jízdy do oblouku.

Výpočet najížděcí doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{22+5}{9{,}7} = 2{,}78\ s \qquad (2.3.2 - 13)$$

Výpočet vyklizovací doby:

$$t_n \frac{L_n}{v_n} = \frac{19}{7} = 2{,}71\ s \qquad (2.3.2 - 14)$$

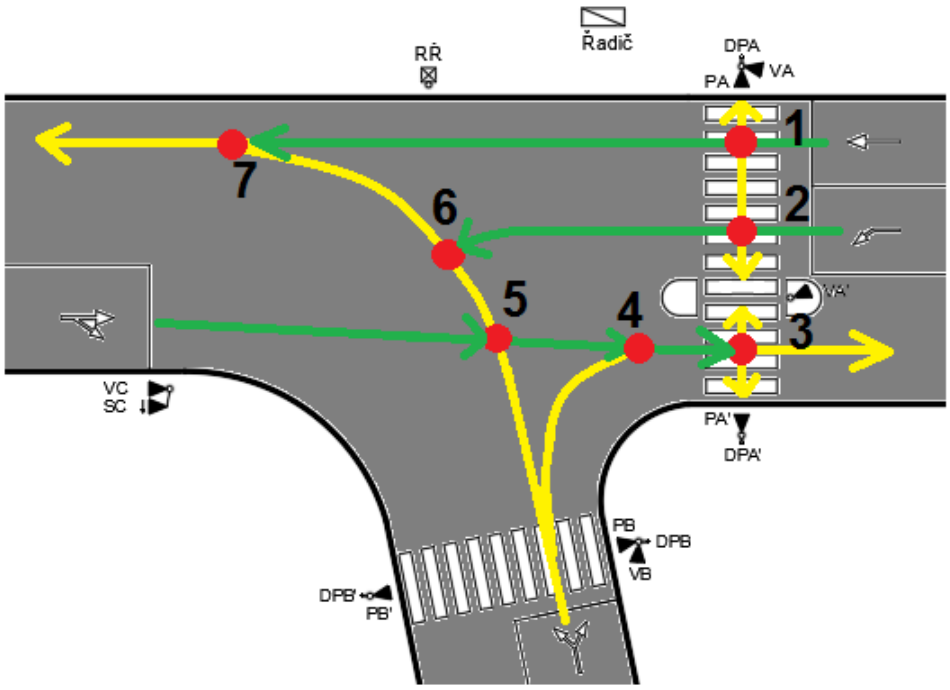
Vypočítané hodnoty jsem opět dosadil do finálního vzorce pro mezičas. Za parametr bezpečnostní doby t_b jsem dosadil 2 s, jelikož opět může dojít k najetí vozidla do křižovatky i při signálu „Pozor!“.

Výpočet finální hodnoty mezičasu pro kolizní bod č. 5:

$$t_m = t_v - t_n + t_b = 2{,}78 - 2{,}71 + 2 = 2{,}07\ s \qquad (2.3.2 - 15)$$

2.3.3. Výpočet mezičasů pro hlavní směr

V této kapitole se věnuji výpočtu doby mezičasu pro stav cyklu, kdy končí fáze ve vedlejším směru („svislém“) a začíná fáze v hlavním („vodorovném“) směru.



Obrázek 11 – Kolizní body pro hlavní směr

Na tomto obrázku (viz obrázek 11) jsem zelenou barvou zvýraznil možné dráhy jízdy vozidla v hlavním směru, které se protínají se žlutě zvýrazněnými dráhami jízdy vozidel vyklízejících křižovatku z minulé fáze. Červeně jsem označil všechny kolizní body a očísloval jsem je. Následně k všem těmto bodů stanovým hodnoty mezičasu, ze kterých pak určím finální mezičas pro tento směr.

Výpočet mezičasu kolizního bodu číslo 1 (přechod pro chodce)

První kolizní bod najíždějícího vozidla je na přechodu pro chodce. Výpočet mezičasu pro bezpečné vyklizení křižovatky chodcem jsem spočítal následovně. Do vzorce pro vyklizovací dobu jsem dosadil délku přechodu 6 m od kraje ke střednímu ostrůvku a za vyklizovací rychlost jsem dosadil 1,4 m/s, což je dle tabulky pro standardní výpočet mezičasů rychlost chodce. Za parametr délky vozidla se v případě chodce dosazuje 0 m.

Výpočet vyklizovací doby chodce:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{6+0}{1{,}4} = 4{,}29\ s \qquad (2.3.3-1)$$

Najížděcí doba vozidla:

$$t_n \frac{L_n}{v_n} = \frac{1{,}5}{9{,}7} = 0{,}16\ s \qquad (2.3.3-2)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 1:

$$t_m = t_v - t_n + t_b = 4{,}29 - 0{,}16 + 0 = 4{,}13\ s \qquad (2.3.3-3)$$

Výpočet mezičasu kolizního bodu číslo 2 (přechod pro chodce)

Druhý kolizní bod se opět nachází na přechodu pro chodce, tentokrát však v pruhu pro odbočení vozidel doleva. Výpočet tak zde bude stejný jako pro kolizní bod číslo 1, jen při výpočtu najížděcí doby dosadím za rychlost vozidla 7 m/s z důvodu zohlednění, že vozidlo jede do oblouku.

Stanovení vyklizovací doby chodce:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{6+0}{1{,}4} = 4{,}29\ s \qquad (2.3.3-4)$$

Stanovení najíždějící doby vozidla:

$$t_n \frac{L_n}{v_n} = \frac{1{,}5}{7} = 0{,}21\ s \qquad (2.3.3-5)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 2:

$$t_m = t_v - t_n + t_b = 4{,}29 - 0{,}21 + 0 = 4{,}07\ s \qquad (2.3.3-6)$$

Výpočet mezičasu kolizního bodu číslo 3 (přechod pro chodce)

Třetí kolizní bod se taktéž nachází na přechodu pro chodce. Zde však pro vozidla v hlavním směru najíždějící z druhé strany křižovatky. Délka přechodu je zde 3 m, najížděcí dráha vozidla 25 m, uvažovaná rychlost vozidla 9,7 m/s.

Stanovení vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{3+0}{1{,}4} = 2{,}14\ s \qquad (2.3.3-7)$$

Stanovení najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{25}{9,7} = 2,58\ s \qquad (2.3.3-8)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 3:

$$t_m = t_v - t_n + t_b = 2,58 - 2,14 + 0 = 0,44\ s \qquad (2.3.3-9)$$

Výpočet mezičasu kolizního bodu číslo 4

Čtvrtý kolizní bod představuje křížení jízdní dráhy vozidla jedoucího z vedlejšího směru odbočujícího doprava a vozidla v hlavním směru najíždějícího do křižovatky zleva a pokračujícího rovně. Délka dráhy vyklízejícího vozidla je 13 m a uvažovaná rychlost tohoto vozidla 7 m/s, jelikož jede do oblouku. Dráha najíždějícího vozidla je pak 19 m a uvažovaná rychlost tohoto vozidla je 9,7 m/s. V tomto případě budu za bezpečnostní dobu t_b dosazovat 2 s, pro zohlednění případné jízdy vyklizujícího vozidla během signálu „Pozor!“.

Stanovení vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{13+5}{7} = 2,57\ s \qquad (2.3.3-10)$$

Stanovení najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{19}{9,7} = 1,96\ s \qquad (2.3.3-11)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 4:

$$t_m = t_v - t_n + t_b = 2,57 - 1,96 + 2 = 2,61\ s \qquad (2.3.3-12)$$

Výpočet mezičasu kolizního bodu číslo 5

Pátý kolizní bod představuje křížení jízdní dráhy vozidla jedoucího z vedlejšího směru, které tentokrát odbočuje doleva a opět vozidla v hlavním směru najíždějícího do křižovatky zleva a pokračujícího rovně. Vyklízející dráha vozidla je v tomto směru 12 m a uvažovaná rychlost vozidla 7 m/s, z důvodu jízdy do oblouku. Najíždějící vozidlo má dráhu ke koliznímu bodu dlouhou 13 m. Rychlost pro toto vozidlo je 9,7 m/s, jelikož jeho dráha vede rovně. i v tomto případě budu za bezpečnostní dobu t_b dosazovat 2 s, pro zohlednění případné jízdy vyklizujícího vozidla během signálu „Pozor!“.

Stanovení vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{12+5}{7} = 2{,}43\ s \qquad (2.3.3 - 13)$$

Stanovení najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{13}{9{,}7} = 1{,}34\ s \qquad (2.3.3 - 14)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 5:

$$t_m = t_v - t_n + t_b = 2{,}43 - 1{,}34 + 2 = 3{,}10\ s \qquad (2.3.3 - 15)$$

Výpočet mezičasu kolizního bodu číslo 6

Tento kolizní bod představuje křížení drah vozidel, která najela do křižovatky z vedlejšího směru a odbočují doleva s vozidly, která odbočují z pravé strany hlavního směru taktéž doleva. Jejich dráha pro vyklizení křižovatky je dle situačního schématu křižovatky 17 m, vyklizovací rychlost pro jízdu do oblouku pak 7 m/s. Vozidla najíždějící z hlavního směru mají vzdálenost od stopčáry ke koliznímu bodu 15 m, uvažovaná rychlost je opět 7m/s, z důvodu, že dráha těchto vozidel vede do oblouku. Při výpočtu finální hodnoty mezičasu budu za parametr bezpečnostní doby t_b opět dosazovat hodnotu 2 s, z důvodu zahrnutí možného projetí vyklizujícího vozidla signál „Pozor!“.

Stanovení vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{17+5}{7} = 3{,}14\ s \qquad (2.3.3 - 16)$$

Stanovení najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{15}{7} = 2{,}14\ s \qquad (2.3.3 - 17)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 6:

$$t_m = t_v - t_n + t_b = 3{,}14 - 2{,}14 + 2 = 3{,}00\ s \qquad (2.3.3 - 18)$$

Výpočet mezičasu kolizního bodu číslo 7

Teno poslední kolizní bod představuje křížení jízdní dráhy vozidel z vedlejšího směru odbočujících doleva a vozidel najíždějících z pravé strany hlavního směru jedoucích rovně. Vyklizovací dráha odbočujících vozidel je v tomto případě 23 m, uvažovaná rychlost 7 m/s z důvodu jízdy do oblouku. Pro najíždějící vozidla je dráha od stopčáry ke koliznímu bodu 19 m a rychlost pro tyto vozidla 9,7 m/s z důvodu přímé jízdní dráhy. Pro finální výpočet mezičasu

pro tento kolizní bod opět dosadím za parametr bezpečnostní doby t_b 2 s pro zohlednění možného projíždění signálu „Pozor!“ vyklízejícími vozidly.

Stanovení vyklizovací doby:

$$t_v = \frac{L_v + l_{voz}}{v_v} = \frac{23+5}{7} = 4{,}00\ s \qquad (2.3.3 – 19)$$

Stanovení najížděcí doby:

$$t_n \frac{L_n}{v_n} = \frac{19}{9{,}7} = 1{,}96\ s \qquad (2.3.3 – 20)$$

Výpočet finální hodnoty mezičasu pro kolizní bod č. 7:

$$t_m = t_v - t_n + t_b = 4{,}00 - 1{,}96 + 2 = 4{,}04\ s \qquad (2.3.3 – 21)$$

2.3.4. Stanovení finální hodnoty mezičasu pro danou křižovatku

V přechozích dvou podkapitolách (2.3.2 a 2.3.3) jsou stanoveny hodnoty mezičasů pro všechny kolizní body pro vedlejší a hlavní směr křižovatky. Z vypočítaných mezičasů jsem sestavil tabulky, ze kterých plynou finální hodnotu mezičasů pro přechod zelené fáze z hlavního do vedlejšího směru a obráceně pro přechod z vedlejšího do hlavního směru (viz tabulka 3 a tabulka 4).

Hodnoty zapisované do tabulek jsem vypočítával s přesností na setiny sekundy a výsledné hodnoty jsem zaokrouhloval na celé sekundy, jak je uváděno v TP pro navrhování SSZ pro řízení silničního provozu (TP 81, 2015, str. 163). Dle těchto podmínek se vypočítané hodnoty mezičasů zaokrouhlují asymetricky, tzn. že hodnoty do 0,30 s se zaokrouhlují dolu a nad 0,30 s zase nahoru. Tyto podmínky se využívají při přesném digitálním zaměřování situace a měření délek v konstrukčním programu, kde lze měřit s přesností 0,1 m. V mém případě však pracuji pouze s hypotetickou křižovatkou a z toho důvodu jsem zvolil zaokrouhlování všech naměřených hodnot mezičasů na celé sekundy vždy nahoru, což doporučují TP pro navrhování SSZ v případech méně přesných měření (TP 81, 2015, str. 163).

Stanovení mezičasu pro přechod fáze z hlavního směru do směru vedlejšího

Tabulka 3 – Mezičasy pro vedlejší směr

Kolizní bod	Vypočtené mezičasy	Zaokrouhlené mezičasy
Číslo 1	4,48 s	5 s
Číslo 2	3,15 s	4 s
Číslo 3	2,74 s	3 s
Číslo 4	7,29 s	**8 s**
Číslo 5	2,07 s	3 s

Dle této tabulky jsem vybral nejdelší dobu vypočítaných mezičasů (viz tabulka 3). V tomto případě to byl mezičas pro kolizní bod číslo 4 s mezičasem 8 s. Z toho vyplývá jeden z požadavků na emulaci chování křižovatky. Při přechodu fáze z hlavního směru na vedlejší musí být tento časový interval mezičasu dodržen z důvodu bezpečnosti provozu na křižovatce.

Stanovení mezičasu pro přechod fáze z vedlejšího směru do směru hlavního

Tabulka 4 – Mezičas pro hlavní směr

Kolizní bod	Vypočtené mezičasy	Zaokrouhlené mezičasy
Číslo 1	4,13 s	**5 s**
Číslo 2	4,07 s	**5 s**
Číslo 3	0,44 s	1 s
Číslo 4	2,61 s	3 s
Číslo 5	3,09 s	4 s
Číslo 6	3,00 s	3 s
Číslo 7	4,04 s	**5 s**

Dle této tabulky s vypočítanými mezičasy pro změnu fáze z vedlejšího směru na hlavní jsem vybral nejdelší hodnotu mezičasu (viz tabulka 4). V tomto případě jsou zde hned tři nejdelší mezičasy, a to pro kolizní body číslo 1, 2 a 7. U všech těchto bodů jsem vypočítal minimální dobu mezičasu na 5 s. Z toho tedy vyplývá další z požadavku na emulaci chování křižovatky a to ten, že při změně fáze z vedlejšího směru do hlavního musí být dodržen minimální stanovený interval 5 s z důvodu zajištění bezpečnosti provozu na křižovatce.

Pro mezičasy reálných SSZ nasazených v provozu platí: „Po uvedení SSZ do provozu je vhodné nastavené mezičasy opakovaným pozorováním přezkoumat, přičemž je zapotřebí dbát především na situace, kdy vozidla odbočují doleva podmíněně kolizním pohybem, a na prostředky MHD“ (TP 81, 2015, str. 28).

2.3.5. Stanovení délky signálů volno

Jedním z hlavních parametrů při návrhu signálního plánu jsou délky zelených fází. Zelné fáze či jednoduše fáze představují stav, kdy na určité signální skupině svítí signál volno, umožňující pohyb vozidlům či chodcům. Délky fází se obvykle posuzují dle dopravních požadavků jednotlivých jízdních směrů dané křižovatky, tak aby došlo k optimalizaci kapacity křižovatky. Jedním z hlavních podkladů pro stanovení délky zelených fází je dopravní průzkum, kterým se stanovuje intenzita pohybu vozidel a chodců v jednotlivých směrech. Dále se také dopravními průzkumy stanovuje skladba dopravních proudů, kterou se určují typy vozidel v jednotlivých dopravních proudech, například zda-li se v dopravních proudech vyskytují často osobní vozidla, nákladní vozidla či vozidla městské hromadné dopravy (autobusy, tramvaje). Následně se stanoví hodnota délky signálů volno pomocí jedné ze tří možností výpočtů.

- Metoda saturovaného toku
- Metoda spotřeby času
- Metoda postupného přibližování (iterace)

Tyto metody se využívají při výpočtech signálů volno pro konkrétní případy křižovatek z praxe, u kterých je možné stanovit intenzity vozidel z jednotlivých směrů. V případě zvolené hypotetické křižovatky tyto parametry křižovatky nejsou samozřejmě k dispozici a proto jsem se při stanovení dob signálů volno řídil minimálními a doporučenými hodnotami.

Dle TP pro navrhování SSZ jsou minimální hodnoty pro signály volno pro vozidla, chodce, cyklisty i tramvaje stanoveny na hodnotu 5 s. Doporučené hodnoty pro pevné signální plány pro motorová vozidla jsou pak v hlavním přímém směru alespoň 12 s a pro vedlejší směry 8 s (Smělý, 2007, str. 24–25). Pro stanovení signálního plánu předmětné emulace byly tedy použity tyto doporučené hodnoty.

2.3.6. Stanovení délky cyklu

Délka cyklu reprezentují celkovou dobu plynoucí z opakující se sekvence jednotlivých fází. Délka cyklu se stanovuje jako součet dob zelených signálů volno a součtu hodnot nejdelších mezičasů pro příslušné signály volno. Výpočet se provádí dle rovnice 2.3.6 – 1 (TP 81, 2015, str. 28).

$$C = \sum t_z + \sum t_m \; [s] \qquad (2.3.6 - 1)$$

kde:

C je požadovaná délka cyklu v sekundách

t_z je doba zelených fází [s]

t_m je doba nejdelších mezičasů po sobě jdoucích fází [s]

V případě předmětné emulace jsem tedy uplatnil doporučené hodnoty pro délky zelených fází, kdy pro hlavní směr je doporučená délka fáze 12 s a pro vedlejší směr 8 s. Pro hodnotu mezičasů platí výsledky výpočtů dle pododdílu 2.3.4, tj. 5 s a 8 s.

Výpočet délky cyklu:

$$C = (12 + 8) + (5 + 8) = 33\ s \qquad (2.3.6 - 2)$$

Dle tohoto výpočtu jsem tedy stanovil požadovanou délku cyklu signálního plánu pro danou křižovatku.

V TP pro navrhování SSZ jsou stanovené minimální, doporučené i maximální délky cyklů pevných signálních plánů. Zde uvádím jejich orientační hodnoty (TP 81, 2015, str. 28).

- Minimální délka cyklu – 30 s
- Optimální délka cyklu – 50 s až 80 s
- Maximální délka cyklu – 100 s až 120 s

V případě předmětné emulace vyšla délka cyklu pevného signálního plánu 33 s, což splňuje minimální stanovené hodnoty dle TP 81.

2.3.7. Preference chodců

Signální plán pro danou křižovatku bude nastaven tak, aby docházelo k preferenci chodců na obou přechodech. To znamená, že v případě detekce chodce při stisku jednoho z tlačítek chodeckých detektoru DPA nebo DPB dojde ke zkrácení zelené fáze vozidel. V případě stisku tlačítka DPA (DPA´) dojde ke zkrácení zelené fáze v hlavním směru. Naopak při stisku tlačítka DPB (DPB´) se bude zkracovat délka zelené fáze ve směru vedlejším. Stisknutí tlačítka bude vždy signalizováno červenou indikační LED, která se rozsvítí u obou tlačítek najedou po stisku jednoho z tlačítek. Při rozsvícení zeleného signálu na semaforu pro chodce pak obě tyto červené LED zhasnou. Takto budou fungovat obě dvojice tlačítek.

Délku o kterou se budou časové intervaly zelených fází zkracovat jsem stanovil na 3 s. Při stanovení této hodnoty jsem vycházel z doporučené délky zelené fáze pro vedlejší směr, který je 8 s a z nejkratšího možného časového intervalu pro zelené fáze, který je dle TP 81 5 s. Z toho vyšlo, že největší možný časový interval, o který mohu zelenou fázi zkrátit jsou 3 s.

V případě stisku jednoho z tlačítek DPA (DPA´) se zkrátí zelená fáze hlavního směru z 12 s na 9 s. Pokud dojde ke stisku v průběhu této zelené fáze, musí být zachován stanovený 9 s interval pro tuto fázi. Když však už bude interval delší jak 9 s dojde rovnou k ukončení této zelené fáze hlavního směru. Při stisku jednoho z tlačítek DPB (DPB´) dojde ke zkrácení zelené fáze ve vedlejším směru z 8 s na 5 s. Opět bude platit, že musí být zachován 5 s interval pro zelenou fázi. V případě stisku tlačítka v době, kdy už poběží 6 s intervalu, dojde rovnou k ukončení zelené fáze ve vedlejším směru.

V případech, kdy dojde ke stisku tlačítka během doby svícení červeného nebo žlutého signálu na semaforech pro vozidla, nedojde k žádnému zásahu do průběhu signálního plánu. Až během zelených fází dojde k jejich případnému zkrácení. Pokud bude tlačítko stisknuto během stavu, kdy je přecházení umožněno, tj. kdy svítí zelený signál na odpovídajícím semaforu pro chodce, nedojde k žádné časové úpravě signálního plánu. Pokud nebude stisknuto žádné tlačítko pro chodce, bude signální plán pracovat s pevně danými časovými intervaly pro zelené signály. a to již zmíněných 12 s pro hlavní směr a 8 s pro směr vedlejší (viz kapitola 2.3.5).

2.4. Zabezpečení kolizních směrů křižovatky

SSZ se umisťují na křižovatky z důvodu optimalizace jízdních proudů a především zvýšení bezpečnosti pohybu vozidel a chodců v křižovatce. SSZ naopak nemá vlivem svého nasazení a provozu zvyšovat četnost hazardních situací vzniklých na dané křižovatce a s tím

spojenou pravděpodobnost vzniku škody. Z tohoto důvodu musí být zajištěna správná funkce takového zařízení a při jeho případné poruše musí být tato porucha včas detekována.

Jedním z požadavků na program emulace je tak detekování případného výskytu nežádoucích zelených signálů v kolizních směrech, což představuje potenciální selhání s jednoznačnými hazardními následky. V případě že by došlo k detekci takovéto události, musí křižovatka okamžitě přejít do neřízeného stavu, ve kterém na SSZ blikají pouze žluté signály. Provoz na křižovatce v neřízeném stavu pak závisí na svislém dopravní značení, podle kterého se musí účastnící dopravního provozu řídit.

Nefunkčností SSZ pak dochází ke zvýšení sekundárního ohrožení tím, že veškerá bezpečnost na křižovatce závisí na lidském faktoru. Z tohoto důvodu se v praxi provádí pravidelné kontroly funkčnosti SSZ a jeho součástí vyškolenými servisními techniky, aby k těmto hazardním situacím nedocházelo.

3. STANOVENÍ ARCHITEKTURY A STRUKTURY SW VČETNĚ INTERAKCÍ S OKOLÍM A VZÁJEMNÉ INTERAKCE DÍLČÍCH ČÁSTÍ SW

V této kapitole jsou popsány jednotlivé funkce ve vytvořeném programu. Ke každé funkci je též vytvořen vývojový diagram popisující jednotlivé části činnosti dané funkce. Tyto diagramy jsou v příloze 4 (viz str. 78).

3.1. Stanovení architektury programu

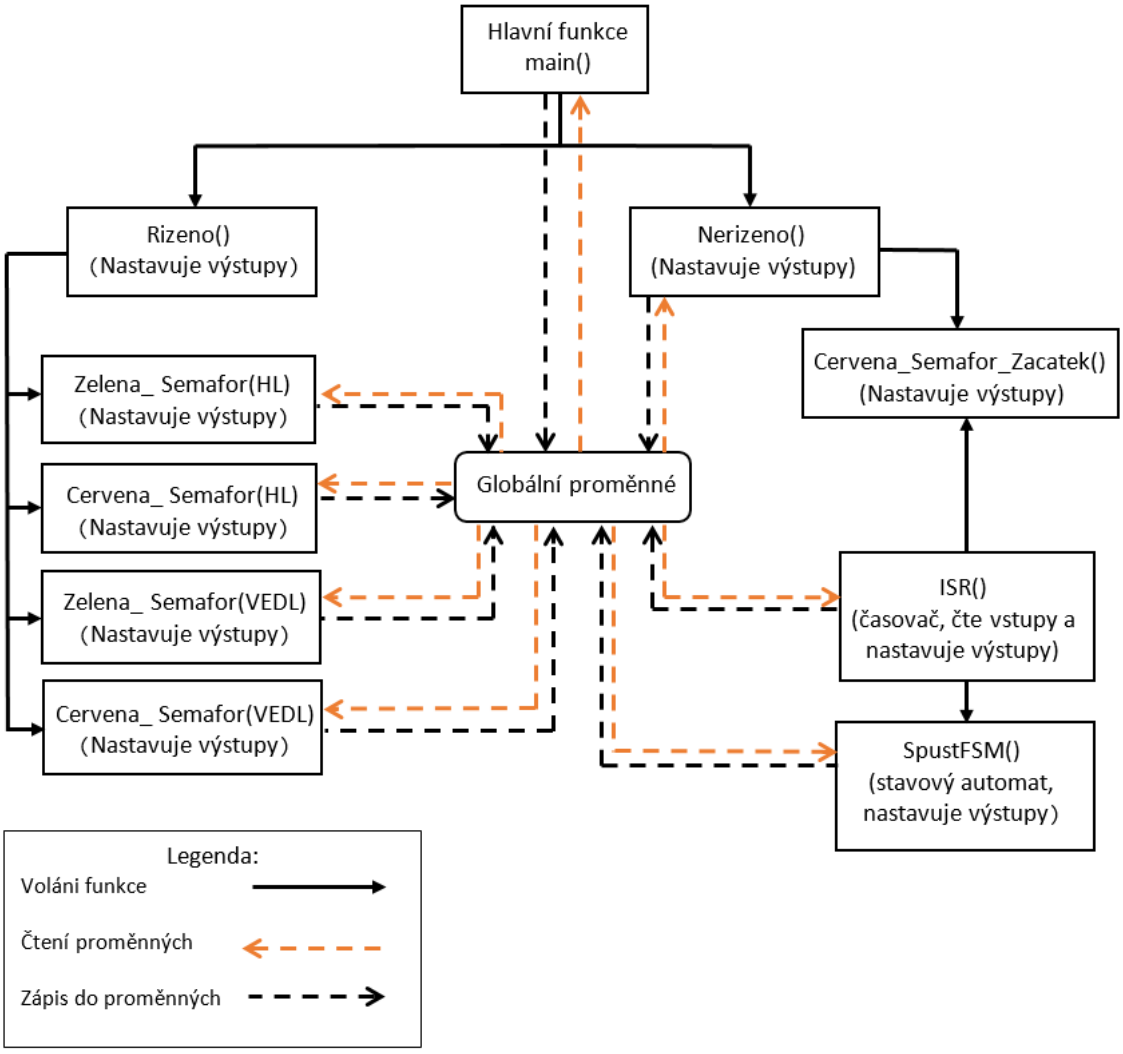
Jedná se o procedurální program, s přesně daným algoritmem pro řešení dané úlohy. Program se skládá z několika funkcí a každá má svou specifickou činnost. Činnosti jednotlivých funkcí jsem si ověřoval a zkoušel již během mé semestrální práce z předmětu Mikroprocesorová řídící technika. Na tuto semestrální práci jsem navázal a rozšířil pro zde stanovené podmínky. Program začíná vložením potřebných knihoven využívaných v programu. Jedná se o následující knihovny. (První 3 uvedené knihovny jsou běžně dostupné v rámci daného vývojového prostředí.)

- `#include <avr/io.h>` – knihovna pro nastavení registru vstupně výstupních portů mikrokontroleru
- `#include <avr/interrupt.h>` – knihovna s příkazy pro obsluhu přetečení časovače
- `#include <util/delay.h>` – knihovna umožňující použití funkce *_delay_ms()*, pro vytváření časových zpoždění
- `#include "tlacitka.h"` – knihovna zpřístupňující funkci *cti_tlacitka()*, které umožňuje čtení stavu tlačítek na modulu s tlačítky (viz 1.1.3). (Mašek, 2020)

Dále jsou na začátku programu definovány globální konstanty, které slouží pro nastavování vlastností křižovatky, jako jsou délka žlutých signálů „Pozor!“ či délka zelených fází. Následuje definice globálních proměnných využívaných v rámci programu, které slouží například k ukládání stavu stisku tlačítek nebo pro uložení indikace přítomnosti chodce či detekce poruchy. Nakonec jsou zde deklarace jednotlivých funkcí.

Následuje spuštění základní funkce celého programu *main()*. Dochází v ní k nastavení všech potřebných součástí, které program využívá, jak jsou definice připojení různých periférií

(tlačítka, semafory) a také nastavení časovače. Po těchto nastaveních se na konci funkce *main()* dostane program do uzavřené smyčky *while(1)*. Ta představuje základní řídící strukturu celého programu. Z této smyčky jsou dále volány další funkce dle následujícího diagramu (viz obrázek 12).



Obrázek 12 – Struktura programu

Podrobnější popisy jednotlivých částí programu jsou součástí vlastního v kódu programu. Níže uvádím rámcový přehledový popis. Výpis celého kódu programu je pak uveden v příloze 5 (viz str. 89).

3.2. Popis hlavní funkce *main()*

Celý program se skládá z několika hlavních funkcí, kde každá funkce je pro určitou činnost. Základní funkcí celého programu, jak již bylo uvedeno, je funkce *main(),* na jejímž začátku dochází k nastavení všech používaných periférií. Následně se v této funkci spustí uzavřená řídící smyčka *while()*, ze které se následně volají 2 funkce zajišťující chod světelného signalizačního řízení křižovatky. Na začátku uzavřeného cyklu program kontroluje podmínkou *if,* stav proměnných *prvni* a *porucha*. V případě že se hodnota proměnné *prvni* rovná 0 a zároveň se proměnná *porucha* nerovná hodnotě 1, je volána funkce *Rizeno()*, která zajišťuje řízení křižovatky dle signálního plánu. V případě že jedna z těchto podmínek není splněna, je volána funkce *Nerizeno()*, která představuje neřízený stav křižovatky, kdy na semaforech blikají žlutá světla.

3.3. Popis funkce *Nerizeno()*

Funkce *Nerizeno()* zajišťuje během neřízeného stavu křižovatky blikání žlutých signálů na všech semaforech. Pomocí zpoždění, které se provádí funkcí *delay(),* se v této funkci nastavuje perioda blikání žlutých signálů. Při uplynutí zadaného zpoždění dochází vždy ke změně stavu žlutých signálů na křižovatce. Tato funkce se periodicky volá. Během každého průchodu touto funkcí se po změně stavu žlutých signálů také sníží hodnota pomocné proměnné *prvni* o mínus jedna. V základním stavu je této proměnné přiřazena hodnota 9. Každým průchodem funkcí *Nerizeno()* se tedy tato hodnota sníží o 1. Zároveň je při průchodu touto funkcí hodnota proměnné *prvni* vždy porovnávána pomocí podmínky *if*. Pokud dojde ke snížení proměnné *prvni* na hodnotu 1, bude podmínka *if* splněna a dojde k vykonání další části této funkce. Tou je zajištění přechodu do řízeného stavu křižovatky. To se provádí voláním speciální funkce, která se jmenuje *Cervena_semafor_zacatek()* a slouží pro rozsvícení červených signálů na celé křižovatce. Po jejím vykonání se v této funkci ještě uskuteční časové zpoždění 3 s, opět pomocí funkce *delay(),* což zajišťuje dodržení stanoveného intervalu mezičasu. Jinými slovy, jedná se o časový úsek, kdy budou na celé křižovatce svítit pouze červené signály a bude docházet k vyklízení křižovatky od vozidel, které do ní najely během neřízeného stavu. Následně se ještě sníží hodnota proměnné první o mínus 1, čímž se tato proměnná dostane na nulu. Jako poslední částí, která se v této funkci vykoná je povolení globálního přerušení (viz. kapitola 3.8).

3.4. Popis funkce *Cervena_semafor_zacatek()*

Funkce *Cervena_semafor_zacatek()* je velmi jednoduchá. Její činností je rozsvícení žlutých signálů na požadovaný čas stanovený pro signál „Pozor!“ na 3 s. Následné dojde ke zhasnutí žlutých signálů a rozsvícení signálů červených na všech semaforech jak pro vozidla, tak pro chodce. Toto je celá činnost této funkce. Speciální je ale v tom, že rozsvěcí světelné signály naráz na více signálních skupinách. Z tohoto důvodu nelze pro rozsvícení červených signálů použít funkce *Cervena_semafor()*, protože ta pracuje vždy s jednou signální skupinou a ne s více, resp. oběma naráz.

3.5. Popis funkce *Zelena_semafor()*

Jedná se o jednu z hlavních funkcí pro řízení signálů na semaforech. Její činnost zajišťuje přechod požadované signální skupiny a příslušného přechodu pro chodce z červených signálů do signálů zelených umožňující pohyb vozidel a chodců. Přechod z červených signálů na semaforech pro vozidla začíná rozsvícením červených i žlutých signálů na 2 s, jejich následným zhasnutím a až potom rozsvícením zelených signálů.

Při volání této funkce se předává do funkce i číselná hodnota 1 nebo 2, která se dosazuje za proměnou *smer* a stanovuje funkci jaký ze směrů hlavního či vedlejšího má přejít do zelených signálů. Hodnota 1 značí požadavek nastavení zelených signálů pro hlavní směr a hodnota 2 pak pro směr vedlejší. Ve funkci se nachází dvě podmínky *if*, které proměnou *smer* porovnávají a dle její hodnoty pak spouští příslušný kód pro rozsvícení zelených signálů hlavního nebo vedlejšího směru.

Po splnění dané podmínky *if* a nastavení zelených signálů pro příslušnou signální skupinu se vynuluje proměnná *cyklus_HL* nebo *cyklus_VEDL* dle toho, o kterou signální skupinu se jedná. Následuje ještě nastavení globální proměnné *stav*, do které se ukládá, v jakém stavu se křižovatka nachází. V tomto případě se tedy do ní uloží buď ZELENA_HLAVNI nebo ZELENA_VEDLEJSI. Více o využití této globální proměnné je v kapitole 3.9. Dále se program dostane do části, která zajišťuje odměření požadovaného časového intervalu zelené fáze. Tuto činnost jsem zajistil pomocí cyklů *do-while*, které by se daly do češtiny přeložit jako „pracuj, doku je splněna podmínka“. Princip těchto cyklů spočívá v průchod kódu, který je v těle cyklu , přičemž po jeho průchodu se kontroluje podmínka *while*, pokud je podmínka splněna, kód v tomto cyklu se začne vykonávat znovu. Tento cyklus se opakuje do té doby, dokud je podmínka *while* splněna.

V podmínce každého z cyklu *while* (viz 2 podmínky if zmíněné výše) se pracuje vždy s jednou z globálních proměnných *cyklus_HL* nebo *cyklus_VEDL*, dle směru o který se jedná. Tyto proměnné jsou každých 0,1 s pomocí časovače inkrementovány o +1 (viz 3.8.1). Následně se jejich hodnota porovnává s jednou z globálních konstant ZELENA_HLAVNI nebo ZELENA_VEDLEJSI, které jsou definovány na začátku programu tak, aby splnily požadované časové intervaly příslušných zelených fází. Pokud inkrementovaná hodnota je stejná nebo větší než globálních konstanta ZELENA_HLAVNI nebo ZELENA_VEDLEJSI cyklus *do-while* se ukončí, neboli se ukončí interval zelené fáze.

Poslední částí kódu funkce *Zelena_semafor()* je vynulování pomocné proměnné *ochrana*. o funkci této proměnné se dočtete v kapitole 3.9.

3.6. Popis funkce *Cervena_semafor()*

Tato funkce má obdobný princip jako funkce *Zelena_semafor()*. V tomto případě však funkce slouží pro přechod ze zelených do červených signálů. Stejný principem se pomocí proměnné *smer* při volání této funkce stanovuje, který ze směrů hlavního či vedlejšího má přejít na červený signál. Pomocí podmínky *if* se pak hodnota proměnné *smer* porovnává a v případě hodnoty 1, bude přecházet na červený signál hlavní směr a při hodnotě 2 směr vedlejší.

Po vyhodnocení podmínky *if* se do globální proměnné *stav* uloží hodnota MEZI_STAV. Využití této globální proměnné je popsáno v kapitole 3.9. Následně dojde ke zhasnutí zelených signálů a rozsvícení žlutých signálů v požadovaném směru. Žluté signály svítí pod dobu definovanou pomocí globálních konstanty POZOR, která je defaultně nastavená tak, aby žluté signály svítily 3 s. Po uplynutí tohoto intervalu dojde ke zhasnutí žlutých signálů a rozsvícení pouze těch červených. Tím daná funkce končí.

3.7. Popis funkce *Rizeno()*

Toto je hlavní funkce pro realizaci signálního plánu. Funkce je volána z uzavřené smyčky programu po splnění podmínky *if*, která kontroluje, zda-li se pomocná proměnná *prvni* rovná 0. Pomocná proměnná *prvni* se rovná 0 v případě, že skončila celá činnost funkce *Nerizeno()*.

Funkce *Rizeno()* představuje základní stav křižovatky a slouží pro volání funkcí pro změny světelných signálů. Jinými slovy, realizuje změny světelných signálů dle signálního plánu. Jako první volá funkci *Zelena_semafor* pro rozsvícení zelených signálů v hlavním

směru. Pro určení směru, který má přejít na zelené signály se při volání této funkce v závorce zadává hodnota 1 nebo 2, čímž se volí, který směr (hlavní pro hodnotu 1 nebo vedlejší pro hodnotu 2) má přejít do zelených signálů. Další volanou funkcí je *Cervena_semafor*. Ta zajistí přechod na červené signály, i při volání této funkce je nutné nastavit hodnotu proměnné *smer* pro volbu směru, který má přejít do červených signálů obdobně jako u předchozí funkce. Následuje časový interval, který je nastaven pomocí globální konstanty CELO_CERVENA_HL_VEDL pro hlavní směr na 3 s a zajišťuje dodržení požadované hodnoty mezičasu, kdy na semaforech svítí červené signály. Po skončení intervalu pro dodržení mezičasu se opět volá funkce *Zelena_semafor,* s hodnotou pro volbu směru, který má přejít do zelených signálů (v tomto případě se jedná o vedlejší směr). Následuje volání funkce *Cervena_semafor* pro rozsvícení červených signálů, opět s hodnotou pro volbu směru, který má přejít do červených signálů. Nakonec je ve funkci ještě nastaven zpožďující časový interval, nastavený v globální konstantě CELO_CERVENA_VEDL_HL, pro dosažení požadovaného mezičasu. Během tohoto intervalu svítí na semaforech červené signály. Tento časový interval má však definovaný čas 0 s. To je z důvodu, že pomocí výpočtu byla hodnota celkového mezičasu pro daný směr stanovena na 5 s. Tento mezičas je tak dodržen pomocí žlutého signálů „Pozor!“, který trvá 3 s a navazujícího žlutého signálu (současně s červeným signálem) v druhém směru, upozorňující řidiče, aby se připravili k jízdě, který trvá 2 s. Dohromady tak oba tyto žluté signály dávají 5 s, což je požadovaná hodnota mezičasu a není tedy potřeba zpožďujícího intervalu při kterém by svítily pouze červené signály. Tento zpožďující časový interval je zde pouze pro případné pozdější úpravy mezičasů, kdy lze snadno jeho prodloužením dosáhnout požadovaného mezičasu.

3.8. Popis funkce *ISR()*

Jedná se o speciální funkci, která se spustí vždy při přetečení časovače. V mém programu využívám 16-bitový časovač Timer/Counter 1, kterým je mikrokontroler ATmega32 vybaven. Časovač pracuje na principu zvyšování hodnoty v čítacím registru při každém tiku taktovacích hodin mikrokontroleru. Pokud se naplní daný čítací registr, dojde k tzv. přetečení tohoto registru a to vyvolá tzv. přerušení. Při přerušení pak dojde ke spuštění funkce ISR (interrupt service routine), což by se do češtiny dalo přeložit jako obsluha přerušení. Čítání tohoto časovače je nastaveno tak, aby k již zmíněnému přetečení a volání funkce ISR došlo vždy po 100 ms (viz 3.8.1).

Na začátku této funkce se nastavuje hodnota čítacího registru pro přetečení tak, aby opět přetekl po 100 ms. Další částí je inkrementování dvou globálních proměnných cyklus_HL a cyklus_VEDL o +1. Tyto proměnné využívám při odměřování času zelených fází na semaforech. Jejich hodnota se vždy po 100 ms zvýší o +1. Následným porovnáním těchto proměnných lze pak zjistit, zdali už dosáhly například hodnoty 120, což představuje odměření časového intervalu 12 s, který je využíván u zelené fáze hlavního směru.

Další činností této funkce ISR je čtení stavu tlačítek chodců. Při čtení stavů tlačítek se vytvoří 8 bitové číslo, kde každý jeden bit odpovídá stavu jednoho tlačítka. Stisk tlačítka je v tomto čísle zaznamenán uložením hodnoty 1 na příslušný bit, ke kterému jsou tlačítka připojena. Pokud nedojde ke stisku tlačítka, je v daném bitu uložena hodnota 0. Každý bit pak představuje 1 z 8 tlačítek na modulu s tlačítky a lze tedy poznat, které konkrétní tlačítko bylo stisknuto. Následuje kontrola proměnné *tlacitka,* do které se uložilo číslo s aktuálním stavem tlačítek. První kontrola podmínkou *if* kontroluje bity pro tlačítko číslo 1 a 2, které představují chodecké detektory DPA a DPA´. Pakliže došlo k jejich stisku, je podmínka *if* splněna a pokračuje se ve vykonávání kódu v této podmínce. Nejdříve se nastavuje globální proměnná *chodec_HL* na hodnotu 1, což značí že se na hlavním přechodu objevil chodec. Následuje rozsvícení příslušných indikačních LED číslo 1 a 2 u tlačítek. Druhá kontrola podmínkou *if* je obdobná, v tomto případě se však jedná o kontrolu stisku tlačítek číslo 3 a 4, které představují chodecké detektory DPB a DPB´. Při jejich stisku dojde k nastavení globální proměnné *chodec_VEDL* na hodnotu 1, což znamená že se na vedlejším přechodě objevil chodec. Následně se rozsvítí i příslušné indikační LED u těchto tlačítek, které chodcům indikují, že tlačítko bylo již stisknuto. Nakonec této funkce je volána funkce stavového automatu, která řeší reakce na stisk tlačítek (přítomnost chodců).

V této funkci se dále nachází podmínka, která hlídá zelené signály v kolizních směrech. V případě, že je detekováno rozsvícení zelených signálů v kolizních směrech, dojde k okamžitému zhasnutí všech signálů na křižovatce. Následně je volána funkce, která zajistí blikání žlutých signálů, které značí neřízený stav křižovatky. Zároveň dochází k blikání indikační LED u tlačítka č. 7, což značí detekci poruchy. Pro možnou demonstraci této funkce emulace jsem vytvořil kód, který po stisku tlačítka č. 7 přímo (bez ohledu na další podmínky) rozsvítí zelený signál ve vedlejším směru. Pro zavedení umělé chyby (vytvoření kolizní zelené) tak musí dojít ke stisku daného tlačítka v případě, že svítí zelené signály v hlavním směru. Následuje již výše uvedený postup při detekci této poruchy. Pro viditelnou demonstraci rozsvícení kolizních zelených signálů jsem přidal ještě časové zpoždění 1,5 s před reakcí

programu na detekci této chyby tak, aby hazardní stav svícení zelených pro kolizní směry byl evidentní.

3.8.1. Popis nastavení přetečení časovače

Pro přerušení časovače je zvolena hodnota 0,1 s. Hlavním důvodem, proč je zvolena přesně tato hodnota, je především dostatečná frekvence čtení stavu tlačítek a zároveň, aby se daly dobře nastavovat délky zelených fází, které jsou pomocí časovače odměřovány. Vyšší hodnota by znamenala problém u čtení stavu tlačítek, kdy by se mohlo stát, že nedojde k detekci stisku tlačítka a zároveň by nešlo tak přesně nastavovat délky zelených fází. Z těchto důvodů se mi jeví hodnota 0,1 s jako optimální.

Pro odměřování tohoto časového intervalu je zvolen 16-bitový časovač Timer/Counter 1, jelikož zbylé 2 časovače, kterými mikrokontroler disponuje jsou 8-bitové a ty nedokáží takto dlouhý časový úsek odměřit.

Následně je nutné vhodně nastavit daný časovač. První je určena, dle vzorce č. 3.8.1 – 1, předdělička časovače tak, aby bylo možno odměřit časový úsek 0,1 s.

$$N \geq perioda\ přetečení\ časovače * \frac{F_{cpu}}{2^{poč.bitů}} \qquad (3.8.1 - 1)$$

Zdroj: (Mašek, 2020, str. 9)

Kde:

N je požadovaná předdělička časovače

F_{CPU} je taktovací frekvence mikrokontroleru (v tomto případě je to 14 745 600 Hz)

poč. bitů představuje kolika bitový časovač byl zvolen

$$N \geq 0{,}1 * \frac{14\,745\,600}{2^{16}} = 22{,}5 \qquad (3.8.1 - 2)$$

Z dostupných předděliček (1, 8, 64, 256, 1024) časovače je následně vybrána tak, která je nejbližší vypočítané hodnotě, tedy předdělička 64. Nakonec je potřeba vypočítat hodnotu periody přetečení časovače tak, aby čítací registr přetekl vždy po 0,1 s. Tato hodnota se vypočítává pomocí vzorce č. 3.8.1 – 3.

$$Reload\_Timer1 = 2^{16} - požadovaný\ časový\ interval * \frac{F_{cpu}}{N} \quad (3.8.1 - 3)$$

Zdroj: (Mašek, 2020, str. 39)

Kde:

Reload_Timer1 je hodnota, která se musí zapisovat do čítacího registru, aby přetekl vždy po požadovaném časovém intervalu

N je zvolená předdělička časovače

F_{CPU} je taktovací frekvence mikrokontroleru (v mém případě je to 14 745 600 Hz)

$$Reload\_Timer1 = 2^{16} - 0{,}1 * \frac{14\,745\,600}{64} = 65\,536 - 23\,040 = 42\,496$$

(3.8.1 – 4)

Tato vypočítaná hodnota (42 496) je na začátku programu definována jako globální konstanta. Ta se na začátku programu a vždy po přetečení časovače zapisuje do čítacího registru časovače, čímž je zajištěno že k jeho naplnění (přetečení) dojde vždy po 0,1 s.

3.9. Popis funkce SpustFSM()

Jedná se o tzv. stavový automat neboli anglicky Finite-state machine (FSM). Ten na základě vstupních hodnot realizuje příslušné úkony. Jeho hlavním úkolem v tomto programu je reakce na stisk tlačítek chodců. Při volání stavového automatu se do něj předává informace o aktuálním stavu křižovatky, neboli v jakém stavu se křižovatka momentálně nachází. Jsou celkem 3 možnosti stavů křižovatky. První z nich je *ZELENA_VEDLEJSI*, což značí že se momentálně křižovatka nachází ve stavu zelené fáze ve vedlejším směru. Při tomto stavu kontroluje automat přítomnost chodců pomocí podmínek *if*, které kontrolují globální proměnné *chodec_HL* a *chodec_VEDL*. V případě že je v těchto proměnných uložena hodnota 1, značí to že byl detekován chodec, stiskem některého z tlačítek. Dle hodnot v těchto proměnných pak stavový automat vykonává jednotlivé příkazy. V případě, že došlo k detekci chodce ve vedlejším směru (*chodec_VEDL = 1*), je nutné zkrátit zelenou fázi vedlejšího směru, aby byla zajištěna preference chodců na křižovatce. To je zajištěno pomocí kontroly globální proměnné zajišťující časový interval zelené fáze. Pro vedlejší směr byla stanovena zkrácená doba zelené fáze na 5 s při detekci chodce. Stavový automat tedy zkontroluje, jaký časový úsek z nezkrácené zelené fáze, která je pro vedlejší směr stanovená na 8 s, již uběhl. Pakliže uběhl zatím kratší časový interval než 5 s, program upraví časový interval zelené fáze vedlejšího směru tak, aby trval přesně požadovaných 5 s. V případě že chodec stiskl tlačítko v době, když již byl časový interval zelené fáze vedlejšího směru za požadovanými 5 s, dojde k okamžitému

ukončení této zelené fáze. Při zkracování těchto fází se ještě nastavuje pomocná globální proměnná *ochrana* do hodnoty 1, což značí, že už došlo ke změně časového intervalu a při příštím volání stavového automatu se tedy už nemá časový interval nijak zkracovat. Tím je ošetřeno například opakované stisknutí tlačítek pro chodce. Tato globální proměnná se pak vynuluje po skončení zelené fáze vedlejšího směru, což umožní při dalším stisku tlačítka chodcem opět upravovat časový interval zelené fáze.

Další částí je kontrola globální proměnné pro detekci chodce, tentokrát ale ve směru hlavním. V případě, že se globální proměnná *chodec_HL* rovná 1, znamená to, že bylo stisknuto tlačítko pro chodce v hlavním směru a že svítí i příslušné indikační LED u těchto tlačítek. Když je křižovatka ve stavu *ZELENA_VEDLEJSI*, znamená to však, že svítí i zelená na semaforu pro chodce v hlavním směru a chodcům je tak umožněno přecházení. Proto zde dochází ke zhasnutí indikačních LED a vynulování globální proměnné *chodec_HL*.

Druhým stavem křižovatky je *ZELENA_HLAVNI*, to představuje stav, kdy je na křižovatce zelená fáze v hlavním směru. Princip činnosti stavového automatu je zde obdobný jako pro zelenou ve vedlejším směru. Zde však dochází ke zkracování zelené fáze hlavního směru při detekci chodce stiskem tlačítka v hlavním směru. V základním stavu je zelená fáze v hlavním směru stanovena na 12 s, v případě detekce chodce se zkracuje na 9 s. Pomocí porovnání, stavový automat zjišťuje, jaký časový úsek již uplynul ze zelené fáze hlavního směru a z toho se vyvozuje další činnost. Buď dochází k upravení časového intervalu zelené fáze tak, aby trvala přesně 9 s nebo v případě, že už 9 s ze zelené fáze uběhlo, dochází k okamžitému ukončení zelené fáze. Tím je zajištěna preference chodců v hlavním směru. Následuje ještě podmínka, která v případě že se globální proměnná *chodec_VEDL* rovná 1, tak zhasíná indikační LED u tlačítek na přechodu ve vedlejším směru. To je z důvodu, že při zelené fázi v hlavním směru svítí zelená na semaforu pro chodce ve směru vedlejším a chodcům je tak umožněno přecházet. Nemusí tak již svítí indikační LED stisknutých tlačítek.

Poslední stavem, ve kterém se křižovatka může nacházet, je stav, který jsem označil jako *MEZI_STAV*. Tento stav se objevuje vždy, když na semaforech již nesvítí zelená signály. Jedná se tedy o stav, kdy na semaforech svítí buď žluté či červené signály nebo oba tyto signály najednou. V tomto případě neprobíhá ve stavovém automatu žádná činnost a běh kódu stavového automatu je tak ukončen.

4. DEFINICE TESTOVÝCH PŘÍPADŮ PRO OVĚŘENÍ POŽADOVANÉHO CHOVÁNÍ A VLASTNOSTÍ EMULACE

4.1. Testování signálních obrazů

Jednou ze zásadních věcí pro funkčnost této emulace řízení křižovatky je správné rozsvěcení signálů na jednotlivých semaforech pro vozidla i chodce. Především, aby nedošlo k rozsvícení například zeleného a červeného signálu současně nebo vynechání některého signálu, jako například výstražného žlutého signálu upozorňujícího řidiče, že bude následovat signál stůj. Rozsvícením špatného signálu může totiž dojít ke vzniku hazardní situace, a proto je nutná kontrola rozsvícení všech signálů.

Správné pořadí signálních obrazů vypadá následovně. Po spuštění programu pro řízení křižovatky nastane tzv. neřízený stav (hovoříme také o „náběhu do neřízeného stavu“). Během toho stavu musí na všech semaforech pro vozidla blikat žluté signály, na chodeckých semaforech v tomto stavu nic nesvítí. Po uplynutí určitého intervalu dojde k trvalému rozsvícení žlutého signálu na všech semaforech na dobu 3 s. Během této doby již na semaforech pro chodce musí svítit červené signály. Po uplynutí těchto 3 s bude následovat celočervený stav na křižovatce, při kterém bude na všech semaforech pro auta i chodce svítit červený signál taktéž 3 s, pro dodržení délky mezičasu. V tomto stavu se už jedná o křižovatku řízenou světelnými signály. Následuje rozsvícení žlutého signálu společně se stále svítícím červeným signálem v hlavním směru na 2 s. Součet délek těchto signálů pak dává dohromady požadovaných 8 s, které respektují vypočtenou délku mezičasu (pro tento stav byla zvolena nejdelší hodnota mezičasu, pro dodržení bezpečnosti provozu, tj. dostatečného času pro vyklizení křižovatky najetými vozidly). Po těchto 2 s se již rozsvítí zelený signál pro vozidla v hlavním směru a pro chodce ve směru vedlejším. Po skončení této fáze se rozsvítí červená na semaforu pro chodce a vozidlům se na semaforu rozsvítí samostatný žlutý signál na 3 s. Následovat bude opět celočervený signál, na ten naváže zelená fáze ve vedlejším směru. Nejdříve se opět rozsvítí žlutý signál se stále svítícím červeným signálem na 2 s. Dále se už rozsvítí pouze zelený signál pro vozidla ve vedlejším směru a pro chodce ve směru hlavním. Po uplynutí příslušného intervalu pro tuto fázi se rozsvítí na semaforech pro vozidla žlutý signál na 3 s a na semaforech pro chodce signál červený. Následně se již rozsvítí červený signál také pro vozidla Zároveň s rozsvícením červeného signálu pro vozidla ve vedlejším směru se již

rozsvítí žlutý signál pro vozidla v hlavním směru, značící začínající zelenou fázi v hlavním směru.

Takto proběhl jeden celý cyklus signálního plánu. Při testování je nutné zajistit kontrolu svícení všech požadovaných signálů ve správném čase, dle tohoto popisu a zejména ověřit, zda-li nedochází k rozsvícení nesmyslných signálů, jako například současně svítící zelený a červený signál na jednom semaforu. Kontrola má probíhat vždy na jednom semaforu po dobu celého cyklu. Po konci jednoho cyklu se přejde k dalšímu semaforu. Testování musí být provedeno jak během řízeného, tak během neřízeného stavu křižovatky. Po testování se vyhodnotí všechny testované prvky. V případě kdy nedojde k detekci žádného problému, je možno přikročit k dalšímu testování.

4.2. Testování dodržení stanovených intervalů mezičasů

V kapitole 2.2, jsou vypočítány požadavky na časové intervaly mezičasů pro danou křižovatku. Jak bylo již zmíněno, je interval mezičasu z hlediska bezpečnosti na řízené křižovatce velmi důležitý. Nedodržení vypočítané minimální hodnoty mezičasu zvyšuje riziko výskytu hazardní situace, při které vyklízející vozidlo nestihne včas opustit křižovatku, před příjezdem najíždějícího vozidla v kolizním směru. Z tohoto důvodu je nutné důsledně otestovat časové intervaly mezičasů.

Pro předmětnou křižovatku byly stanoveny celkem 2 hodnoty mezičasů. Jeden mezičas pro případ přechodu fáze z hlavního směru na vedlejší, kdy je vypočítána minimální hodnota mezičasu 8 s. V případě přechodu fáze z vedlejšího směru na hlavní směr je vypočítána požadovaná doba mezičasu 5 s. Obě tyto hodnoty musí být otestovány měřením pro ověření dodržení zadaných parametrů.

4.2.1. Testování mezičasu po náběhu křižovatky

První ověřující měření bude nutno provést pro stav po náběhu křižovatky z neřízeného stavu (blikajících žlutých signálů) do řízeného stavu. Postup měření bude následující. Po určitém intervalu přejdou blikající žluté signály na svítící žluté signály a následně se rozsvítí samostatně červené signály na celé křižovatce. Při rozsvícení žlutých signálů se začne odměřovat časový interval až do zahájení první zelené fáze. Tento naměřený časový interval se pak musí shodovat se zadanými požadavky. Jelikož při neřízeném stavu se vozidla řídí pouze svislým dopravním značením, mohlo dojít k zastavení některého z najíždějících vozidel v prostoru křižovatky. Například při odbočování doleva, když dávalo vozidlo přednost

protijedoucím vozidlům v hlavním směru. Z tohoto důvodu bude pro uvažovaný mezičas pro přechod z neřízeného stavu křižovatky do zelené fáze pro libovolný směr nutné dodržet minimální stanovenou hodnotu mezičasu 8 s.

4.2.2. Testování mezičasů pro řízený stav křižovatky

Další dvě měření pro ověření mezičasů se už budou týkat pouze řízeného stavu křižovatky. Jak bylo zmíněno, bude se jednat o přechod zelené fáze z hlavního do vedlejšího směru a následně z vedlejšího směru do směru hlavního.

Prvním testovaným bude mezičas při přechodu zelené fáze z hlavního směru na vedlejší. Tento mezičas je pomocí výpočtů a zohlednění odbočujících vozidel stanoven na 8 s. Znamená to tedy, že na konci zelené fáze v hlavním směru, když dojde k zhasnutí zeleného signálu a rozsvícení žlutého signálu „Pozor!“ se začne odměřovat stopkami časový interval. Tento interval bude odměřován až do doby než se rozsvítí zelený signál ve vedlejším směru. Pokud se bude naměřený časový interval shodovat s požadovanými 8 s, resp. nebude kratší, bude výsledek testu úspěšný.

Druhý testovaný mezičas bude pro přechod zelené fáze z vedlejšího do hlavního směru. Mezičas pro tento případ jsem vypočítal 5 s. Čas se začne odměřovat opět po zhasnutí zeleného signálu ve vedlejším směru a rozsvícení žlutého signálu „Pozor!“, až do doby rozsvícení zeleného signálu v hlavním směru. Když se bude naměřený časový interval shodovat s požadovanými 5 s, resp. nebude kratší, bude výsledek testu úspěšný.

V případě, že některé z naměřených hodnot mezičasů budou více rozdílné od požadovaných hodnot, provede se ještě jedno měření. Pokud bude i při tomto měření naměřený časový interval kratší než požadovaný, bude nutné upravit kód, aby splnil zadané požadavky. Měření se pak bude stejným principem opakovat.

4.3. Testování délky signálů volno

Nedílnou částí signálního plánu, kterou bude nutné otestovat jsou přímo jednotlivé zelené fáze. Přestože jejich délka není přímo bezpečnostně kritická, je důležité dodržet stanovené časové intervaly zelených fází, zejména z hlediska dosažení potřebné kapacity křižovatky.

První testovanou fází bude zelená fáze v hlavnímu směru a s tím spojená zelená fáze chodců na vedlejším přechodu (PB). Pro tuto fázi byl stanoven časový interval 12 s. Měřením se tedy musí ověřit, že je na semaforech v hlavnímu směru (VA a VC) a chodeckém semaforu

(PB) dodržen tento stanovený interval. Délka signálů volno nemá přímý vliv na bezpečnost provozu na křižovatce, přesto je důležité ověřit splnění zadaných parametrů pro optimalizaci kapacity křižovatky. Druhou obdobně testovanou fází bude fáze ve směru vedlejším se kterou je spojená zelená fáze chodců na hlavním přechodu (PA). Pro tuto zelenou fázi byl stanoven časový interval 8 s. Měřením se tedy tento časový interval ověří na vedlejším semaforu (VB), odbočovací šipce (SC) a chodeckém semaforu (PA).

Důležitou podmínkou při tomto testování bude vynechání reakce signálního plánu při stisku chodeckých tlačítek. Při jejich stisku totiž může dojít ke zkrácení některé ze zelených fází, což by pak vedlo na špatné odměření časového intervalu. Proto během tohoto testování nesmí dojít ke stisku ani jednoho z chodeckých tlačítek. Pokud dojde k jeho stisku, bude nutné začít měření znovu.

4.4. Testování funkčnosti chodeckých tlačítek

Další testovanou částí budou tlačítka chodců. Na křižovatce se nacházejí dvě dvojice chodeckých detektorů (tlačítek) označených DPA a DPB. Jako základní testování bude ověření, že při stisku tlačítka na jedné straně přechodu dojde k rozsvícení indikační LED, a to na obou stranách křižovatky. Testovaní se provede na obou dvojících tlačítek pro chodce. Pří správné funkci by se měly indikační LED rozsvítit najednou na obou stranách a svítit do té doby, než bude chodcům umožněno přejít. Ve chvíli, kdy se rozsvítí zelená pro chodce, dojde k zhasnutí obou indikačních LED.

Další částí testování je nutné ověřit, že během opětovného stisku tlačítka nedojde například ke zhasnutí indikačních LED. Na opakovaný stisk tlačítka nesmí program nijak reagovat. Testováním se musí také ověřit, že ve stavu, kdy je na přechodu pro chodce zelená a dojde ke stisku tlačítka v daném směru, nebude na tento stisk program nijak reagovat a nedojde například k rozsvícení indikačních LED. Tyto vlastnosti je nutné otestovat na obou přechodech u všech 4 tlačítek.

4.5. Testování preference chodců

Po ověření funkčnosti chodeckých tlačítek je také potřeba ověřit reakci programu na jejich stisk. Jak už bylo řečeno v kapitole 2.3.7, bude docházet k časové preferenci chodců díky zkracování délky zelených fází. V případě chodců v hlavním směru se zkrátí zelená fáze hlavního směru z 12 s na 9 s a v případě chodců ve vedlejším směru se zkrátí zelená fáze ve

vedlejším směru z 8 s na 5 s. Testováním bude nutné ověřit zkracování těchto intervalů na požadované hodnoty. Při zkracování zelených fází zaleží také na době, kdy bylo tlačítko stisknuto. Například pokud chodec v hlavním směru stiskne chodecké tlačítko DPA až v době, kdy už běží například 10 s zelené fáze v hlavním směru, tak samozřejmě se už tato fáze nedá zkrátit na požadovaných 9 s. V tomto případě ale dojde k okamžitému ukončení zelené fáze v hlavním směru. Obdobně pak pro fázi ve vedlejším směru, když dojde ke stisku chodeckého tlačítka DPB až v době, kdy běží například 6 s této zelené fáze, nemůže dojít k dodržení požadovaných 5 s. Ale zelená fáze se v tomto případě ukončí rovnou, jelikož už 5 s interval uběhl.

Bude nutné tedy otestovat všechny tři možnosti, při kterých může dojít ke stisku tlačítka. První bude stav, kdy dojde ke stisku tlačítka, když nebude probíhat zelená fáze. Druhá možnost kdy může dojít ke stisku tlačítka bude v případě, že už zelená fáze probíhá, ale kratší dobu, než je stanovený zkrácený interval. V tomto stavu se zelená fáze ukonči v čase, kdy dosáhne požadovaného zkráceného intervalu. Třetí možností je, stisk tlačítka ve stavu, kdy už zelená fáze běží delší dobu, než je požadovaná zkrácená doba. V tomto případě musí dojít k okamžitému ukončení této zelené fáze. Všechny tyto sepsané možnosti musí být otestovány u obou dvojic tlačítek přechodů.

4.6. Testování zabezpečení zelených kolizních signálů

Nezbytnou částí emulace, kterou je potřeba otestovat je zabezpečení kolizních směrů křižovatky. Neboli, aby nedošlo chybou programu nebo jinou z hlediska provedení emulace blíže nespecifikovatelnou poruchou k rozsvícení zelených signálů v kolizních směrech.

Abych mohl tuto funkci programu otestovat, vytvořil jsem umělou chybu, která při stisku tlačítka č. 7 rozsvítí zelený signál na semaforu VB. Tuto funkci lze tedy testovat v případě, kdy svítí zelené signály v hlavním směru na semaforech VA a VC. Stiskem tlačítka se zavede umělá chyba v podobě rozsvícení zeleného signálu v kolizním směru na semaforu VB. Program následně tuto chybu detekuje a musí na ni správně zareagovat. Před reakcí programu na tuto událost jsem ještě vložil časovou prodlevu 1,5 s, aby bylo možné vidět, že opravdu došlo k rozsvícení zeleného signálu v kolizním směru. Reakce programu je totiž rychlá a k rozsvícení zelených signálů v kolizním směru dojde maximálně na dobu 0,1 s, což nemusí být okem vždy postřehnutelné. Po skončení této prodlevy už program pokračuje dále. První dochází ke zhasnutí všech signálů na celé křižovatce, následuje rozblikání indikační LED č. 7,

která svým blikáním indikuje poruchový stav. Současně je volána funkce *Nerizeno()*, která zajišťuje blikání žlutých signálů na celé křižovatce.

Testování této funkce tedy může proběhnout v případě, kdy svítí zelené signály na semaforech VA a VC v hlavním směru a dojde ke stisku tlačítka č. 7, kterým se zavede umělá chyba.

5. IMPLEMENTACE SW, JEHO ODLADĚNÍ A SPUŠTĚNÍ NA ZVOLENÉM HW

Implementaci a spouštění SW na daném HW jsem prováděl ve dvou hlavních částech. Jako první část jsem připojoval semafory k mikrokontroleru. Následně jsem jim vytvářel kód, který jsem postupně odlaďoval. Druhá hlavní část bylo připojení a zprovoznění modulu s 8 tlačítky a 8 LED.

5.1. Připojení a nastavení semaforů

Jako první věc, kterou jsem udělal, bylo sestavení HW. Semafory pro vozidla i chodce jsem umístil na nepájivé pole a následně jsem je pomocí propojovacích drátů spojil s portem CON2 na základové desce. Dále jsem v programu vytvořil definice připojení těchto semaforů, kdy jsem si určil, jaký světelný signál semaforu je k danému pinu připojen pro pozdější lepší orientaci při programování. Pro každý signál připojený k určitému pinu jsem si vytvářel globální konstantu, kterou jsem pojmenoval například *LED_PIN_NUM_VA_R*. První část názvu této konstanty říká, že se jedná o číslo pinu připojené LED, písmena VA označují, na jakém semaforu se daná LED nachází (např. VA jako semafor pro vozidla či PA jako semafor pro chodce). Poslední písmeno pak určuje barvu dané LED. Zde jsem přistoupil k použití anglických názvů barev, jelikož první písmeno pro zelený a žlutý signál by byly stejné (v kódu nejsou použita písmena s diakritikou). Je tady celkem na výběr ze 3 písmen, písmeno R označuje červený signál (angl. Red), písmeno Y signál žlutý (angl. Yellow) a zelený signál značí písmeno G (angl. Green). Číslo uložené v dané globální konstantě pak označuje na jaký pin je daný LED signál připojen v rozmezí od 0 do 7, jelikož je první pin označen číslem 0. Dále jsem na začátku funkce *main()* nastavil porty C a D, ke kterým jsou semafory připojeny, jako výstupní. a nakonec jsem vytvářel, spouštěl a odlaďoval jednotlivé funkce pro rozsvěcování a zhasínání LED semaforů.

5.2. Připojení a nastavení modulu s 8 LED a 8 tlačítky

Další částí, kterou jsem připojoval a nastavoval byl HW modul s 8 LED a 8 tlačítky. Tento modul jsem připojil k portu CON1 a obdobně jako pro LED semaforů jsem si zde vytvořil globální konstanty, pro jednodušší práci s tímto modulem. Konstanty slouží jak pro práci s tlačítky, tak i s indikačními LED na tomto modulu. Na začátku funkce *main()* jsem nastavil

port A, ke kterému jsou připojena tlačítka, jako port vstupní a port B, ke kterému jsou připojeny indikační LED, jako port výstupní.

U tohoto modulu jsou tlačítka využívána jako detektory chodců. Při stisku tlačítek dochází k rozsvěcení indikačních LED na modulu. To byl asi jeden z největších problémů při programování. Potřeboval jsem totiž, aby se při stisku tlačítka se rozsvítili dvě indikační LED pro daný směr a zároveň zůstal nezměněn stav ostatních LED. To se mi po několika pokusech nakonec podařilo splnit. Druhým problémem byla reakce programu na stisk tlačítek, kdy má docházet ke zkracování délky zelených fází v příslušných směrech. Tento problém jsem řešil přes funkci stavového automatu *SpustFSM()*. Ta sleduje globální proměnnou, která má v sobě uložený aktuální stav křižovatky, a tím pozná, jakou reakci má vyvodit. V případě zelených fází a po detekci chodce v daném směru pak zajišťuje zkracování zelených fází. Zde jsem měl největší starosti s vytvořením programu tak, aby správně reagoval na stisk tlačítka i v době běžící zelené fáze. V tomto případě jsem program vytvořil tak, že dochází ke kontrole intervalu běžící zelené fáze a program vypočítává, jak se musí změnit proměnná, která určuje délku zelené fáze, aby byly dodrženy stanovené časy pro preference chodců (viz 2.3.7).

5.3. Funkce pro kontrolu kolizních zelených signálů

Poslední část, kterou jsem vytvářel byl kód pro kontrolu zelených kolizních signálů. Jeho princip činnosti spočívá v kontrolování nastavení hodnot na pinech, ke kterým jsou připojeny zelené signály v kolizních směrech. V případě, že dojde k detekci nastavení daných pinů do logické úrovně 1, nastaví se globální proměnná *porucha* do hodnoty 1 a křižovatka přejde do neřízeného stavu (blikajících žlutých signálů). Zároveň bliká indikační LED u tlačítka č. 7, která indikuje detekci poruchy. Tímto způsobem je hlídán hazardní stav kolizních zelených na křižovatce. Po detekci daného hazardu se musí program restartovat stiskem tlačítka RESET na základové desce mikrokontroleru.

Pro kontrolu kolizních zelených signálů jsem vytvořil ve funkci časovače ISR (viz 3.8) podmínku, která hlídá nastavení pinů, ke kterým jsou připojeny zelené signály v kolizních směrech. V případě, že dojde k detekci stavu, že na pinech pro zelené signály v kolizních směrech, jsou nastavené hodnoty pro svícení těchto signálů, znamená to že došlo k detekci poruchy. Program na tento stav reaguje zhasnutím všech signálů na křižovatce a následným blikáním žlutých signálů.

6. PROVEDENÍ A VYHODNOCENÍ TESTŮ POŽADOVANÉHO CHOVÁNÍ A VLASTNOSTÍ EMULACE

Tato kapitola je zaměřena na testování chování finálního řešení emulace pro otestování její činnosti. Veškeré testy, které jsou v této kapitole uvedeny, jsou provedeny dle postupu sepsaných v definicích testových případů v kapitole 4. Každý test, který je nadefinován v kapitole 4, je zde dle těchto vytvořených definic proveden, aby se otestovalo chování emulace světelné křižovatky. Každý definovaný test tedy přesně odpovídá jednomu oddílu této kapitoly.

6.1. Testování signálních obrazů

Toto testování mělo za úkol ověřit správnost signálů svítících na jednotlivých semaforech. Testovány byly všechny semafory vozidlové i chodecké, a to vždy po dobu jednoho signálního cyklu. Semafory byly testovány jak během řízeného, tak neřízeného stavu. Potvrzení splnění zadaných požadavků daného semaforu v tabulce představuje hodnocení „OK“ (viz tabulka 5). V případě neshody signálních obrazů s požadovanými je v tabulce vyplněno „Nesplňuje“.

Tabulka 5 – Testování signálních obrazů

Testovaný semafor	Neřízený stav křižovatky	Řízený stav křižovatky	Vyhodnocení
VA	OK	OK	OK
VB	OK	OK	OK
VC	OK	OK	OK
SC	OK	OK	OK
PA	OK	OK	OK
PB	OK	OK	OK

Během tohoto testování byly zkontroloval signální obrazy všech semaforů na křižovatce. Při testu nebyly nezaznamenal žádné nežádané stavy, a test tak vyšel pozitivně, a mohl jsem tak přistoupit k dalšímu testování.

6.2. Testování dodržení stanovených mezičasů

6.2.1. Testování mezičasu po náběhu křižovatky

Toto testování bylo prováděno pomocí odměřování času. Dle požadavků bylo ověřován časový interval mezičasu po náběhu křižovatky. Definovaná hodnota pro tento mezičas byla stanovena na 8 s. Do tabulky (viz tabulka 6) byly zapisovány naměřené hodnoty, které byly naměřeny při testování. Vyhodnocení bylo provedeno na základě naměřených hodnot v porovnání s požadovanými. Při dodržení obou hodnot mezičasů je do vyhodnocení zapsáno „OK“. Při odchylkách od požadované hodnoty je ve sloupci vyhodnocení vyplněno „Nevyhovuje“.

Tabulka 6 – Testování mezičasu po náběhu křižovatky – 1

	Měření č1	**Měření č2**	**Vyhodnocení**
Hodnota mezičasu	9,29	8,77	Nevyhovuje

Při testování hodnot mezičasů při přechodu z neřízeného do řízeného stavu jsem zjistil, že se jejich hodnoty zásadně liší. Z toho důvodu jsem musel provést drobnou úpravu kódu pro dodržení stanovených hodnot. Problém jsem objevil u funkce *Nerizeno()*. Zde je za pomocí proměnné *prvni* nastavována délka (počet) blikání žlutých signálů. V případě, že je do proměnné *prvni* zadáno sudé číslo, bude funkce *Nerizeno()* končit rozsvíceným žlutým signálem. Navazující funkce *Cervena_Semafor_Zacatek()* pak naváže rozsvícením žlutých signálů na 3 s pro přechod do červených signálů. Zde tedy docházelo k sečtení časového intervalu blikající žluté (1,25 s) při neřízeném stavu křižovatky a intervalu (3 s), po který svítí žlutý signál při přechodu do červeného signálu. To zapříčinilo prodloužení celkového mezičasu o 1,25 s. Ve výsledku by však tato chyba neměla žádný zásadní dopad na bezpečnost provozu. Pouze by docházelo k prodlužování hodnoty mezičasu. Přesto jsem tuto chybu opravil a následně opět provedl příslušný test pro ověření splnění požadované hodnoty mezičasu. Výsledky druhého testu lze opět vidět v následující tabulce (viz tabulka 7).

Tabulka 7 – Testování mezičasu po náběhu křižovatky – 2

	Měření č1	**Měření č2**	**Vyhodnocení**
Hodnota mezičasu	8,18	7,99	Vyhovuje

6.2.2. Testování mezičasů pro řízený stav křižovatky

Při tomto testování byly ověřovány 2 mezičasy. Při přechodu zelené fáze z hlavního do vedlejšího směru a naopak z vedlejšího směru do hlavního. Pro mezičas přechodu z hlavního do vedlejšího směru byla stanovena hodnota 8 s (označen v tabulce jako Mezičas HL-VEDL, viz tabulka 8). Pro mezičas přechodu z vedlejšího do hlavního směru pak 5 s (označen v tabulce jako Mezičas VEDL-HL, viz tabulka 8). V tabulce jsou zapsány hodnoty naměřené při testování. Vyhodnocení jsem prováděl porovnáním naměřených hodnot s požadovanými. V případě že se obě naměřené hodnoty shodovaly s požadovanou hodnotou, je ve sloupci vyhodnocení zapsáno „Vyhovuje“. Při odchylkách od požadované hodnoty je ve sloupci vyhodnocení zapsáno „Nevyhovuje“.

Tabulka 8 – Testování mezičasů pro řízený stav křižovatky

Mezičas	**Měření č1**	**Měření č2**	**Vyhodnocení**
Mezičas HL-VEDL	7,99	8,14	Vyhovuje
Mezičas VEDL-HL	4,99	5,05	Vyhovuje

Test jsem provedl pro obě hodnoty mezičasů. Naměřené hodnoty se shodují s požadovanými, pouze z drobnými odchylkami, které jsou zanedbatelné. Test proto vyšel úspěšně.

6.3. Testování délky signálů volno

Tímto testováním jsem ověřoval délky zelených fází v jednotlivých směrech a porovnával je s požadovanými hodnotami. Testována byla zelená fáze v hlavním směru, která má stanovený interval 12 s a zelená fáze ve vedlejším směru, která má stanovený interval 8 s. Důležitou podmínkou při tomto testování bylo, že nesmělo dojít ke stisku chodeckých tlačítek během měření. Reakce programu na stisk tlačítek totiž spočívá ve zkracování délky zelených fází, což by vedlo ke špatnému výsledku testu.

Naměřené hodnoty jsem zapisoval do tabulky a následně provedl jejich vyhodnocení (viz tabulka 9). V případě, že se obě naměřené hodnoty shodovaly s požadovanou hodnotou, je ve sloupci vyhodnocení zapsáno „Vyhovuje“. Při odchylkách od požadované hodnoty je ve sloupci vyhodnocení zapsáno „Nevyhovuje“.

Tabulka 9 – Testování délky signálů volno

Zelená fáze	Měření č1	Měření č2	Vyhodnocení
Hlavní směr	12,05	12,00	Vyhovuje
Vedlejší směr	8,13	8,08	Vyhovuje

Měřením jsem testoval délky zelených fází při nestisknutých tlačítkách od chodců. Pro oba směry se naměřené hodnoty shodovaly s požadovanými a test tak proběhl úspěšně.

6.4. Testování funkčnosti chodeckých tlačítek

Tímto testem jsem ověřoval funkčnost chodeckých tlačítek. Testována byla tlačítka číslo 1, 2, 3 a 4. Ověřovanou činností bylo, zda se rozsvítí indikační LED při stisku tlačítek. Správná funkčnosti tlačítek spočívala v rozsvícení obou indikačních LED při stisku jednoho z dvojice tlačítek. Dalším testováním tlačítek se ověřovalo jejich chování při opakovaném stisku. Tlačítka nesmí nijak na tuto událost reagovat, například zhasnutím indikačních LED ani jinými způsoby. Poslední testování které bylo provedeno, spočívalo v ověření, že při stavu, kdy na přechodu svítí zelená a je tak chodcům umožněno přecházet, nesmí při stisku tlačítek dojít k rozsvícení indikačních LED.

V případě funkčnosti tlačítka je v tabulce zapsáno „OK“. V případě že došlo k jiné reakci, než byla od tlačítek požadována, je v tabulce zapsáno „Nevyhovuje“. (viz tabulka 10)

Tabulka 10 – Testování funkčnosti chodeckých tlačítek

Zelená fáze	Svit indikačních LED	Opakovaný stisk	Stisk při zelené	Vyhodnocení
Tlačítko č1	OK	OK	OK	OK
Tlačítko č2	OK	OK	OK	OK
Tlačítko č3	OK	OK	OK	OK
Tlačítko č4	OK	OK	OK	OK

Při testování se správně rozsvěcely indikační LED pro jednotlivé dvojice tlačítek. Zároveň nebyla při testování zaznamenána žádná reakce na opakovaný stisk, ani rozsvícení indikačních LED během zelené fáze při stisku tlačítka. Všechna tlačítka tak prošla testováním úspěšně.

6.5. Testování preference chodců

Po ověření funkčností tlačítek v minulém testu bylo možno přejít k testování reakce programu na stisk těchto tlačítek. Dle definice testových případů bylo nutné otestovat celkem 3 možnosti, které vedou ke zkracování zelených fází. Ty se liší podle doby stisku tlačítek.

První 2 možnosti bylo možné testovat odměřováním časových intervalů zelených fází. Rozdíl byl pouze v době, kdy došlo ke stisku tlačítka. První testování tedy probíhalo pro stisk tlačítka před začátkem zelené fáze. Ke zkrácení zelené fáze v hlavním směru dojde po stisku jednoho z tlačítek DPA (DPA´), které představují tlačítka č1 a č2. Následným měřením byly zjištěny délky zelené fáze a zapsány do tabulky pro příslušný směr (viz tabulka 11). Po určitém čase v řádu několika minut byl test opakován. Následně byla provedena vyhodnocení obou naměřených hodnot ve vztahu s hodnotami zadanými a určilo se, zda-li se shodují.

Stejným způsobem bylo postupováno při kontrole zkracování zelené fáze ve vedlejším směru. Ke zkracování zelené fáze vedlejšího směru dochází po stisku jednoho z tlačítek DPB (DPB´), které představují tlačítka č3 a č4. Následným měřením byly zjištěny délky zelené fáze po stisku tlačítka a zapsány do tabulky (viz tabulka 11). Druhý test byl prováděn po několika minutách, pro kontrolu správnosti funkčnosti i po delší době běhu zařízení. Následně bylo provedeno vyhodnocení obou naměřených hodnot ve vztahu k zadaným a určeno, zda-li se s nimi shodují.

Tabulka 11 – Testování preference chodců – 1

Zkracovaná fáze	**Test 1**	**Test 2**	**Vyhodnocení**
Zelená fáze hlavní směr	8,98	9,18	OK
Zelená fáze vedlejší směr	4,99	5,06	OK

Další částí testování preference chodců byl stav, kdy došlo ke stisku příslušného tlačítka již v průběhu dané zelené fáze. Přesněji pro stav, kdy běžící interval zelené fáze nepřekročil stanovenou hodnotu při zkrácení zelené fáze. Pro hlavní směr je tato hodnota 9 s a pro vedlejší směr 5 s. Testování bylo provedeno opět dvěma měřeními s časovým odstupem. Naměřené hodnoty byly zapsány do tabulky a následně byly vyhodnoceny (viz tabulka 12).

Tabulka 12 – Testování preference chodců – 2

Zkracovaná fáze	**Test 1**	**Test 2**	**Vyhodnocení**
Zelená fáze hlavní směr	9,07	8,92	OK
Zelená fáze vedlejší směr	5,11	4,98	OK

Poslední testovanou možností byl stisk tlačítka v době, když už interval zelené fáze překročil čas stanovený pro zkrácení zelených fází. V tomto případě má nastat okamžité ukončení příslušné zelené fáze. Testování bylo provedeno za pomocí odměřování času, pro zjištění, zda už se interval zelené fáze dostal za požadovaný zkrácený čas. Následně jsem stiskl jedno z příslušných tlačítek na přechodu a sledoval reakci programu. V případě, že došlo k okamžitému ukončení zelené fáze, je v tabulce zapsáno „OK“. Když nedošlo k okamžitém ukončení zelené fáze, je v tabulce vyplněno „Nevyhovuje“. (viz tabulka 13)

Tabulka 13 – Testování preference chodců – 3

Zkracovaná fáze	**Test 1**	**Test 2**	**Vyhodnocení**
Zelená fáze hlavní směr	OK	OK	OK
Zelená fáze vedlejší směr	OK	OK	OK

Všechny druhy pro preferenci chodců byly testovány celkem dvakrát. V prvních dvou testech byly ověřovány zkracované délky zelených fází. Všechny naměřené hodnoty se shodují s požadovanými a nedochází zde k žádným větším odchylkám. Třetí test, který měl otestovat ukončování zelených fází v momentu stisku tlačítka, kdy byl již časový interval zelených fází za požadovanými zkracovanými hodnotami, vyšel též úspěšně a zařízení se chovalo dle zadaných požadavků. Celkové testování preference chodců na přechodech tak vyšlo pozitivně.

6.6. Testování zabezpečení zelených kolizních signálů

Testování této funkce bylo provedeno přesně dle definice oddílu 4.6. Po spuštění křižovatky a jejího přechodu do řízeného stavu se počkalo, až dojde k rozsvícení zelených signálu v hlavním směru (semafory VA a VC). Stiskem tlačítka č. 7 došlo k zavedení umělé chyby, která rozsvítila zelený signál na semaforu VB. Po krátkém intervalu, který byl v programu zaveden z důvodu demonstrace funkčnosti, program správně zhasl všechny signály a následně rozblikal všechny žluté signály na křižovatce. Zároveň byly ověřeny činnosti indikační LED č. 7, která v případě detekce poruchy bliká.

Testování této funkce byly provedeno v různých časových úsecích probíhající zelené fáze v hlavním směru i při stiscích chodeckých tlačítek. Ve všech případech se program zachoval správně a křižovatka přešla do neřízeného stavu. Testováním tedy vedlo k úspěšnému ověření zabezpečení zelených signálů v kolizních směrech.

7. ZÁVĚR

Cílem bakalářské práce bylo zpracování vývojového (životního) cyklu programu pro emulaci řízení světelného signalizačního zařízení. Vytvářené programy procházejí zpravidla určitým vývojem (životním cyklem), ten se skládá z definice požadavků na program včetně způsobu ověření jejich naplnění, tvorby samotného programu a následného testování, resp. provedení ověření naplnění na začátku vývojového cyklu definovaných požadavků. Stejně tak se dá i tato práce jako celek rozdělit na 4 hlavní části. První z nich je definice požadovaného chování emulace řízení křižovatky a stanovení základních parametrů. Na ní navazuje část zabývající se specifikací testových případů pro ověření naplnění požadovaného chování. Další část je pak samostatná tvorba programu dle požadavků, odlaďování činnosti programu a následné spouštění na vybraném HW. Poslední část cyklu tvoří testování chování finální podoby emulace a ověřování splnění zadaných požadavků.

Můj pracovní postup se tak fakticky řídil takovými jednotlivými fázemi vývojového cyklu a jednotlivé části bakalářské práce tomu odpovídají a daný postup dokládají. Jako prvním jsem si definoval požadované chování prořízení zvolené křižovatky, resp. pro emulaci takového řízení, spolu s definicí testových případů pro ověření správného chování emulace po jejím dokončení. Průběžně jsem si připravil HW část se semafory, které jsem připojil k základové desce. Po jejich připojení jsem začal s tvorbou programu, kde jsem jako první nastavil připojení periférií a následně jsem začal s tvorbou programu, což byla asi nejnáročnější část celé práce. Kód, který jsem programoval by se dal rozdělit na dvě hlavní části. První části které jsem se věnoval byl signální plán s pevně danými časovými intervaly, který určuje chod řízení křižovatky ve smyslu časové posloupnosti jejich stavů. Druhá část, možná i složitější představovala zajištění dynamiky signálního plánu, neboli aby docházelo k dynamické změně intervalů délky zelených signálů v závislosti na stisku chodeckých tlačítek. Poslední částí, které jsem se věnoval bylo již zmíněné testování finálního programu na zvoleném HW. Výsledkem mé práce je SW, který dynamicky řídí signály na semaforech v závislosti na stisku tlačítek chodců.

Téma této bakalářské práce, které jsem si zvolil, ovlivnila především moje praxe, kterou jsem absolvoval u firmy Eltodo a která zajišťuje provoz a servis SSZ v Praze. Dále k výběru tohoto tématu přispěl i předmět Mikroprocesorová řídící technika, který probíhal v zimním semestru 2020/21 a který jsem během zpracovávání tématu BP úspěšně absolvoval. V rámci tohoto předmětu jsem získal potřebné znalosti v programování mikrokontrolerů. Zpracováním této bakalářské práce jsem si prošel určitým vývojem, kdy jsem si rámcově vyzkoušel co

představuje dodržení vývojového (životní) cyklu tvorby bezpečnostně kritické aplikace. Nové poznatky, které jsem zpracováním této BP získal představovaly především zkušenosti se specifikací požadavků a specifikací testů pro bezpečnostně kritickou aplikaci. Na začátku tvorby práce jsem s touto problematikou neměl žádné předchozí zkušenosti a bylo pro mě tak obtížné rozlišit pojmy specifikace požadavků/testů. Postupným zpracováním daného tématu jsem však získal určité základy znalostí této problematiky, což dokládají kapitoly věnované těmto tématům (viz kapitola 2 a 4).

Dosažené výsledky a poznatky dané práce poskytují další náměty na zpracování v rámci ročníkových projektů či diplomové práce v navazující magisterské etapě studia, ve kterém bych rád pokračoval.

Použití literatura:

[1] HRÁBEK, Lukáš 2015. *Automatické semafory řídí dopravu již 85 let. První z nich byl uprostřed Václaváku. Metro.cz* [online]. 21. ledna [cit. 2021-5-4]. Dostupné z: https://www.metro.cz/automaticke-semafory-ridi-dopravu-jiz-85-let-prvni-z-nich-byl-uprostred-vaclavaku-g14-/praha.aspx?c=A150121_114846_co-se-deje_rab

[2] MAŠEK, Zdeněk 2020. *Přednáška č. 5: Čítače/Časovače*. Pardubice: UPa, v Pardubicích, 23. listopad.

[3] PK Design, 2011.*Uživatelský manuál: MB-ATmega16/32 v4.0, Základová deska modulárního vývojového systému MVS*.

[4] PK Design, *2007a*. *Uživatelský manuál: UniProg-USB v1.0, Programovací kabel modulárního vývojového systému MVS.*

[5] PK Design, 2007b. *Uživatelský manuál: Modul 8 LED diod a 8 tlačítek v2.0, (verze dokumentace v1.0)*.

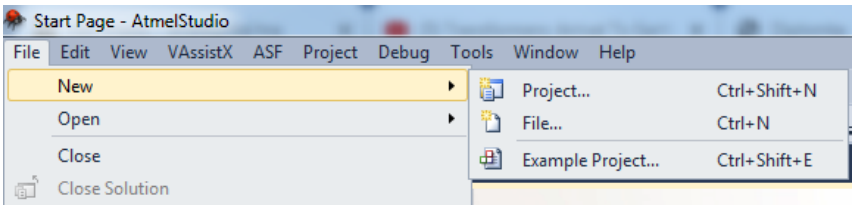
[6] RÁBEK, Vlastimil 2015. *Zákonitosti, typologie a metodika řešení dopravních nehod na křižovatkách řízených soustavou světelných signálů*. In: *Sborník příspěvků konference Expert Forensic Science Brno 2015* [online]. Vysoké učení technické v Brně, Ústav soudního inženýrství, s. 119-183 [cit. 2021-04-15]. ISBN 978-80-214-5100-1. Dostupné z: http://hdl.handle.net/11012/42737

[7] SMĚLY, Martin, 2007. *DOPRAVNÍ INŽENÝRSTVÍ: Řízení úrovňové křižovatky část 1, modul 4* [online]. Brno, Studijní opora. Vysoké učení technické v Brně. [cit. 2021-04-15]. Dostupné z: http://lences.cz/domains/lences.cz/skola/subory/Skripta/CM04-Dopravní%20inženýrství%20(DST)/M04-Řízené%20úrovňové%20křižovatky%20I.pdf.

[8] TP 81, 2015. Technické podmínky - *Navrhování světelných signalizačních zařízení pro řízení silničního provozu na pozemních komunikacích, schválených MD ČR ze dne 21.10. 2015 pod č.j. 122/2015-120-TN/2*. Dostupné z: http://www.pjpk.cz/search.asp?menu=2032&menuX=%2Fvyhledavani%2F&searchText=TP+81&doSearch.x=0&doSearch.y=0

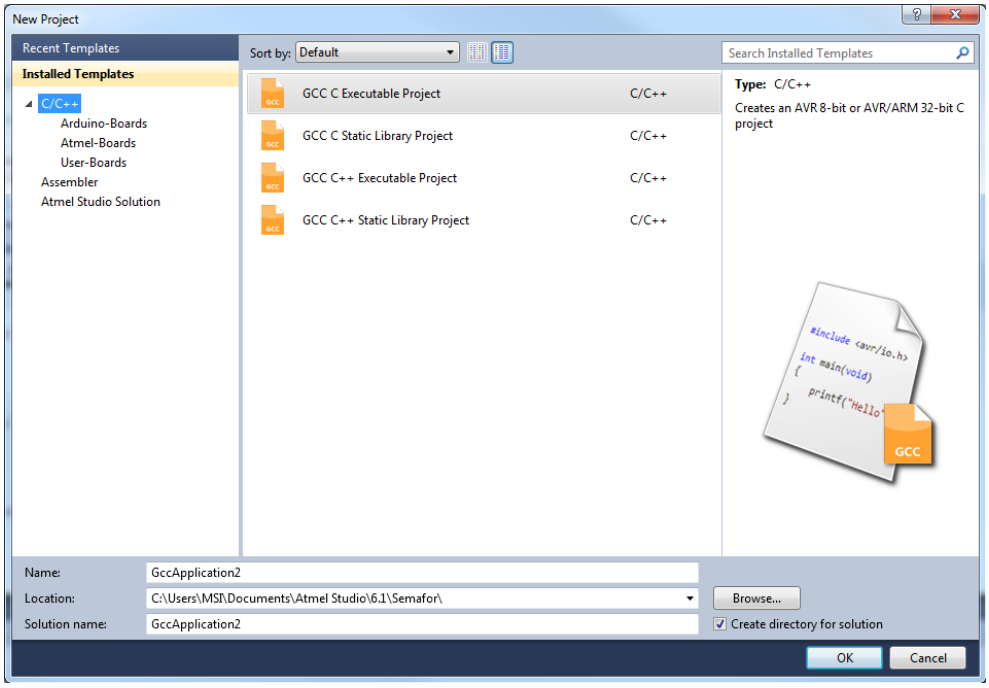
PŘÍLOHA 1 – PRÁCE S ATMEL STUDIEM

Po spuštění Atmel studia se zobrazí hlavní obrazovka programu. Po rozkliknutí záložky *File* v levé horní části se zobrazí nabídka, kde je na výběr možnost otevřít již vytvořený projekt nebo vytvořit nový projekt (viz obrázek 13).



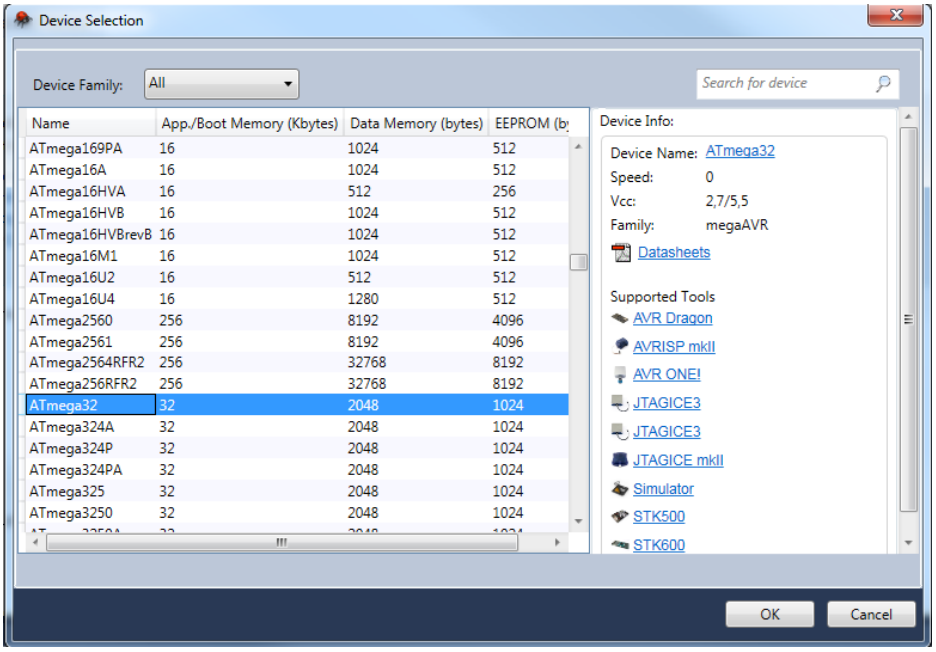
Obrázek 13 – Vývojové studio Atmel

Zvolením možnosti nového projektu se dostaneme do výběrového okna New Project (viz obrázek 14). Zde je na výběr z několika možných nových projektů v programovacím jazyce C a C++. Pro můj program jsem volil možnost „*GCC C Executable Project*“, která umožňuje tvorbu spustitelného programu v jazyce C pro mikrokontrolery AVR 8-bit nebo AVR 32-bit. V dolní částí okna lze nastavit jméno nově vytvářeného projektu a umístění, do kterého se bude projekt ukládat.



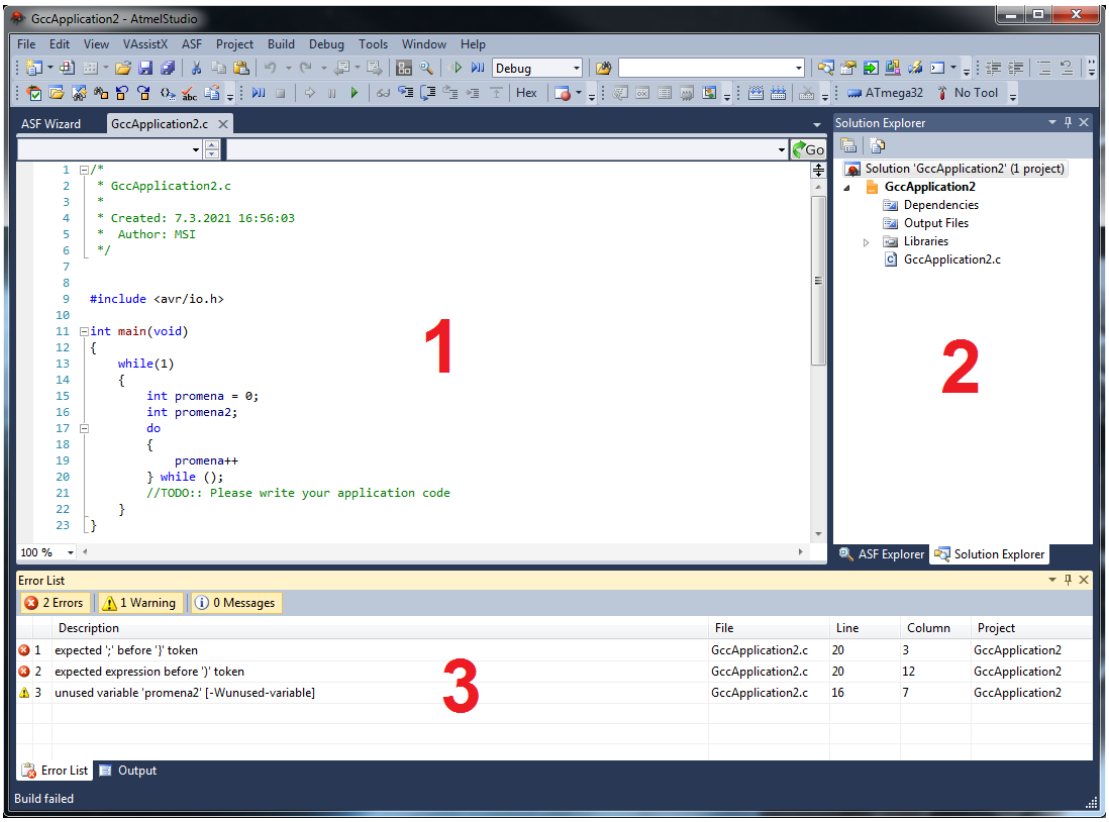
Obrázek 14 – Nový projekt v Atmel studiu

Po zvolení typu, uložiště a jména nového projektu se dostaneme do posledního výběrového okna, což je okno pro výběr zařízení, pro které daný projekt budeme vytvářet. V mém případě využívám mikrokontroler ATmega32.



Obrázek 15 – Volba zařízení v Atmel studiu

Na následujícím obrázku (viz obrázek 16) demonstruji vzorový náhled s programem projektu. Část označené číslem 1 představuje okno editoru, do kterého programátor zapisuje kód programu. Číslem 2 je označena záložka „*Solution Explorer*“, ve které jsou vidět veškeré soubory, které spadají do daného projektu jako například knihovny či přímo daný program. Poslední okno označené číslem 3 představuje informační okno a lze přepínat mezi oknem zobrazujícím informace o průběhu kompilace programu nebo oknem, ve kterém se vypisuje seznam chyb v kódu, tzv. „*Error List*“.

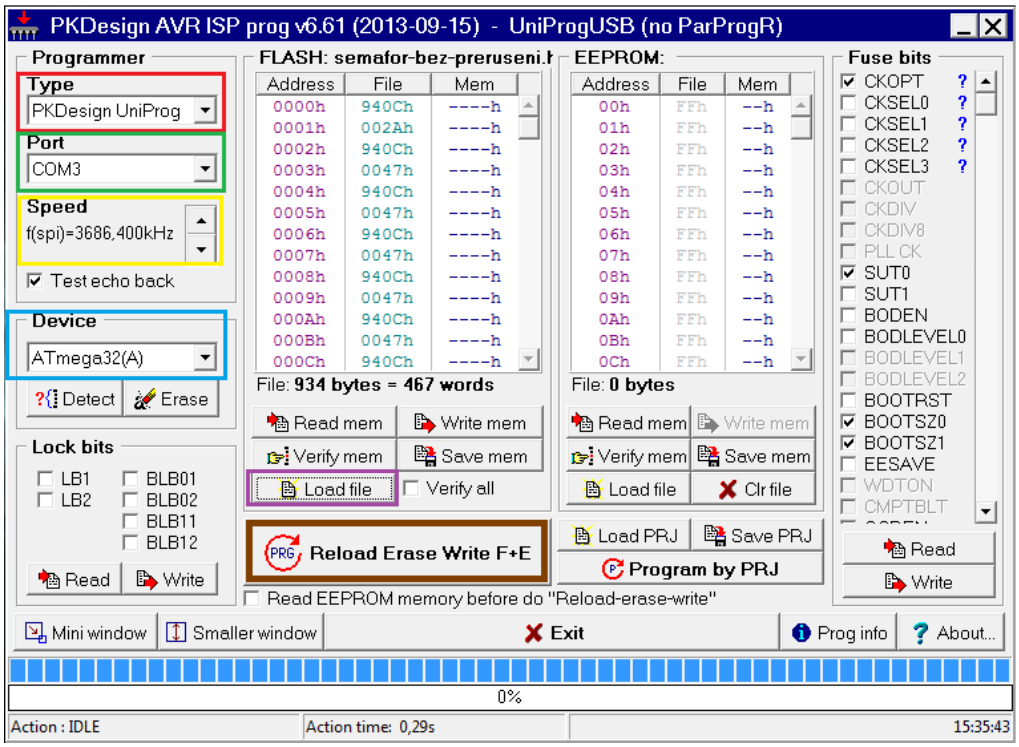


Obrázek 16 – Ukázkový program v Atmel studiu

Na tomto obrázku (viz obrázek 16) jsem chtěl demonstrovat funkci Debuggeru programu. Do programovacího okna jsem napsal pár řádků kódu s několika chybami. Následně jsem spustil Debugger, který se napsaný kód pokusil přeložit a simulovat. Z důvodu chyb však kompilace programu selhala, což v okně výstupu vypíše chybovou hlášku „Build FAILED“ a v okně se seznamem chyb se vypíší chyby, které zabránily kompilaci programu. Dvojklikem na chybovou hlášku program přeskočí na příslušný řádek, na kterém se chyba nachází, což tak zjednoduší detekci a opravu dané chyby programátorovi.

PŘÍLOHA 2 – PRÁCE S PROGRAMEM AVR ISP

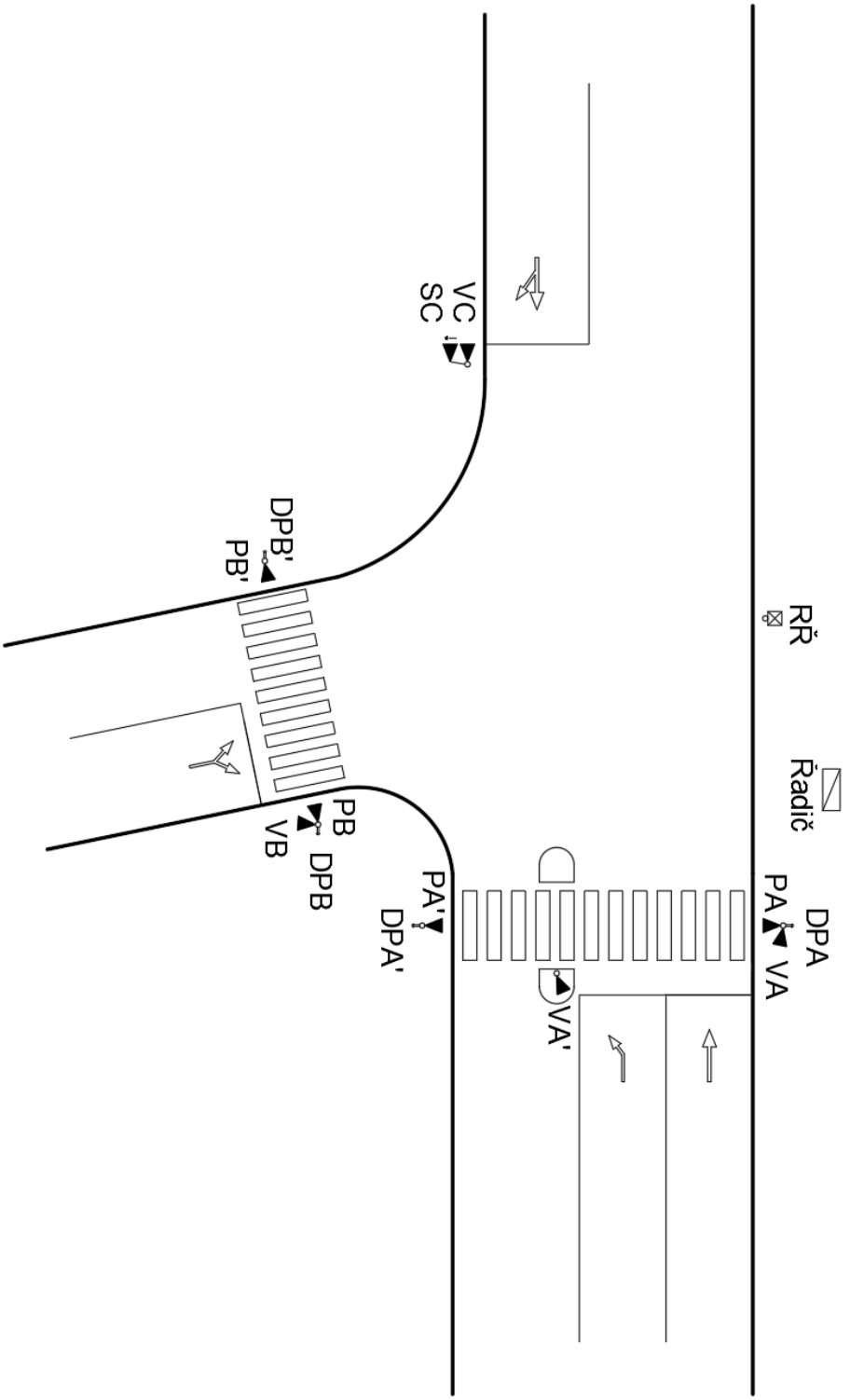
Po připojení mikrokontroleru k PC spustíme na počítači software AVR ISP Programmer a zobrazí se nám následující okno (viz obrázek 17).



Obrázek 17 – AVR ISP programmer

Tento obrázek (obrázek 17) zobrazuje software pro nahrávání programu do mikrokontroleru AVR. Barevnými rámečky jsou zvýrazněny důležité části tohoto programu. První rámeček červeně ohraničený slouží pro nastavení typu programátoru. V mém případě se jedná o programátor PK Design UniProg. Po volbě programátoru program následně sám nalezne připojený UniProg-USB na daném COM portu (znázorněno zeleným rámečkem) a pokusí se detekovat mikrokontroler, ke kterému je UniProg-USB připojený, což je znázorněno světle modrým rámečkem (PK Design, 2007a, str. 20). Po úspěšném připojení programátoru od základové desky mikrokontroleru do počítače můžeme nahrát soubor s vytvořeným kódem. To se dělá pomocí tlačítka Load file (označeného fialovým rámečkem na viz obrázek 17), které po rozkliknutí umožní vybrat požadovaný soubor s kódem z paměti počítače. Poslední krokem je tímto vybraným souborem mikrokontroler přeprogramovat. Přeprogramování mikrokontroleru se provádí stiskem tlačítka Reload Erase Write F+E, které je znázorněno hnědým rámečkem (viz obrázek 17).

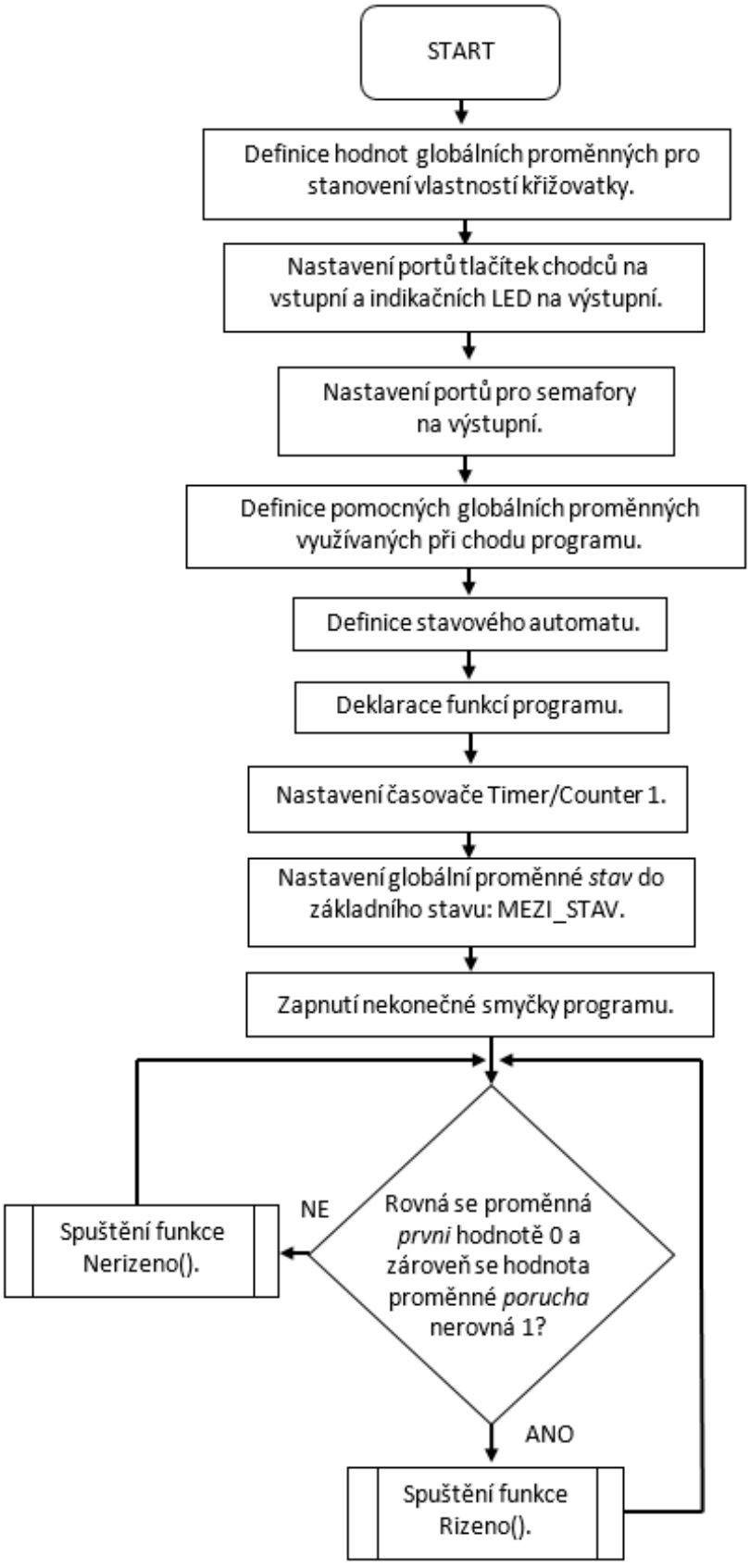
PŘÍLOHA 3 – SITUAČNÍ SCHÉMA KŘIŽOVATKY



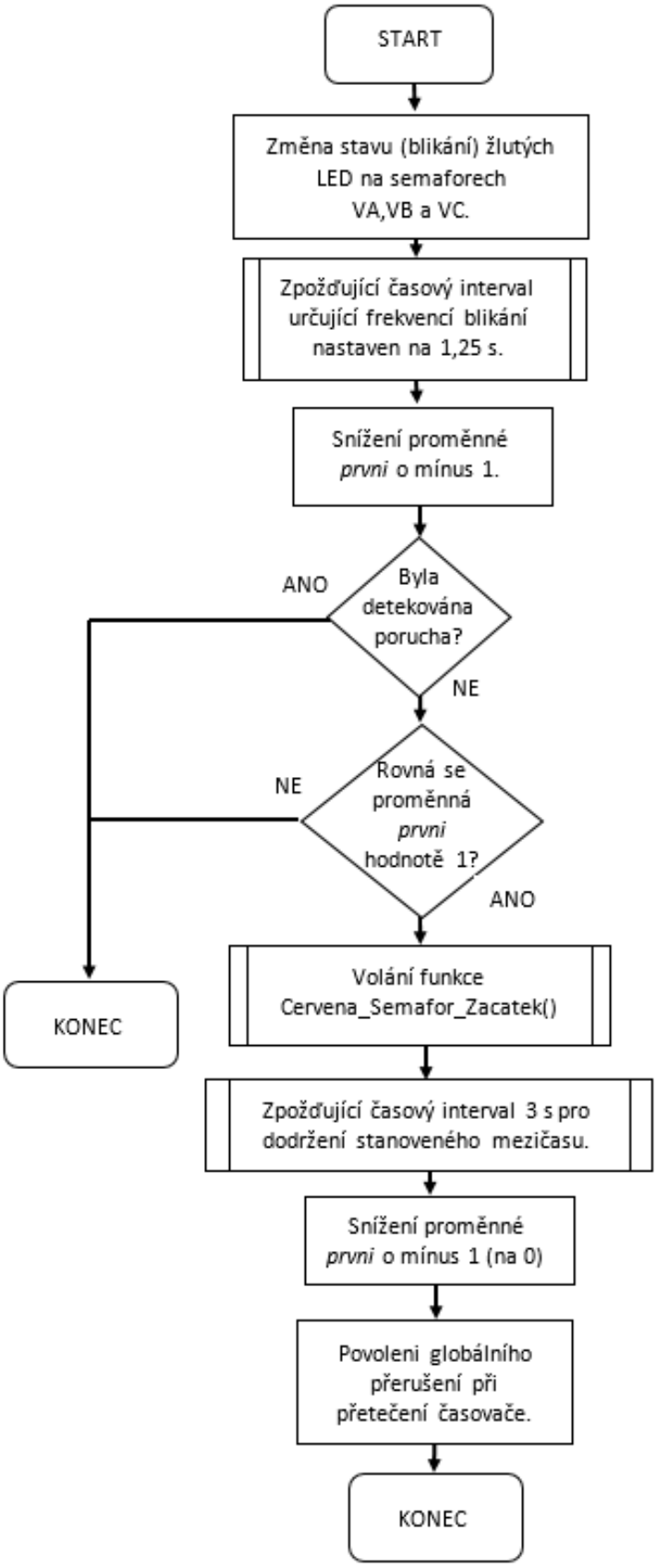
Obrázek 18 – Situační schéma křižovatky

Obrázek 19 – Legenda situačního schéma

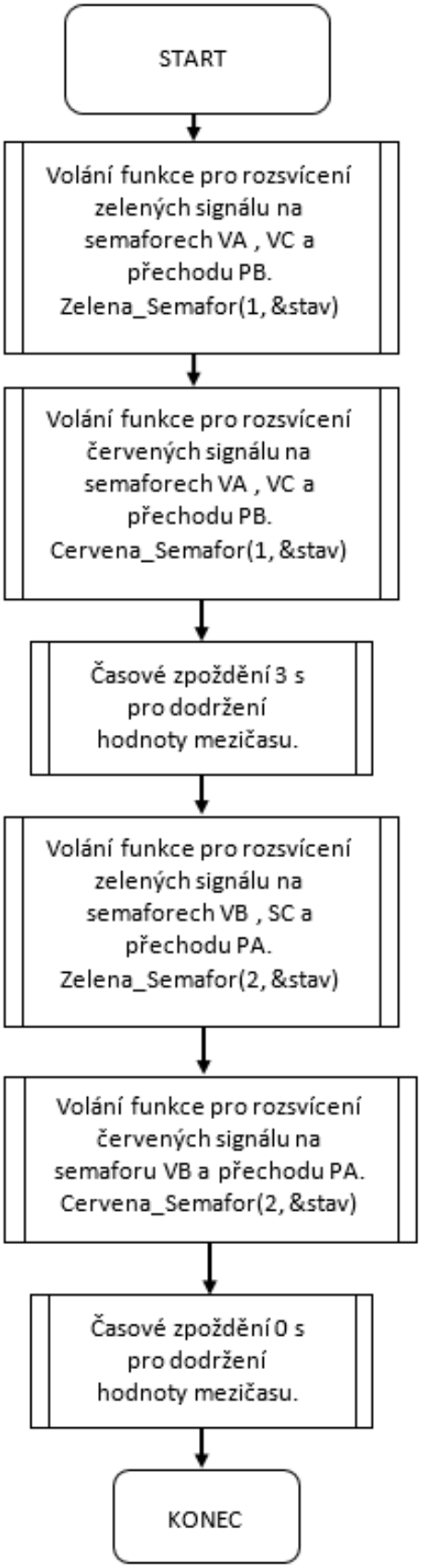
PŘÍLOHA 4 – DIAGRAMY FUNKCÍ



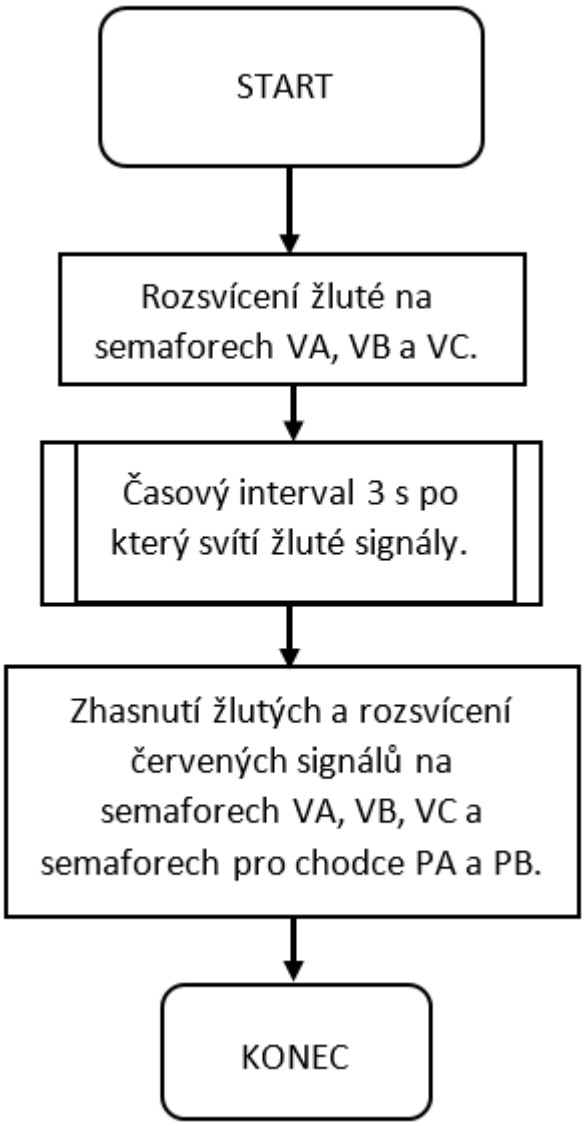
Obrázek 20 – Diagram funkce main()



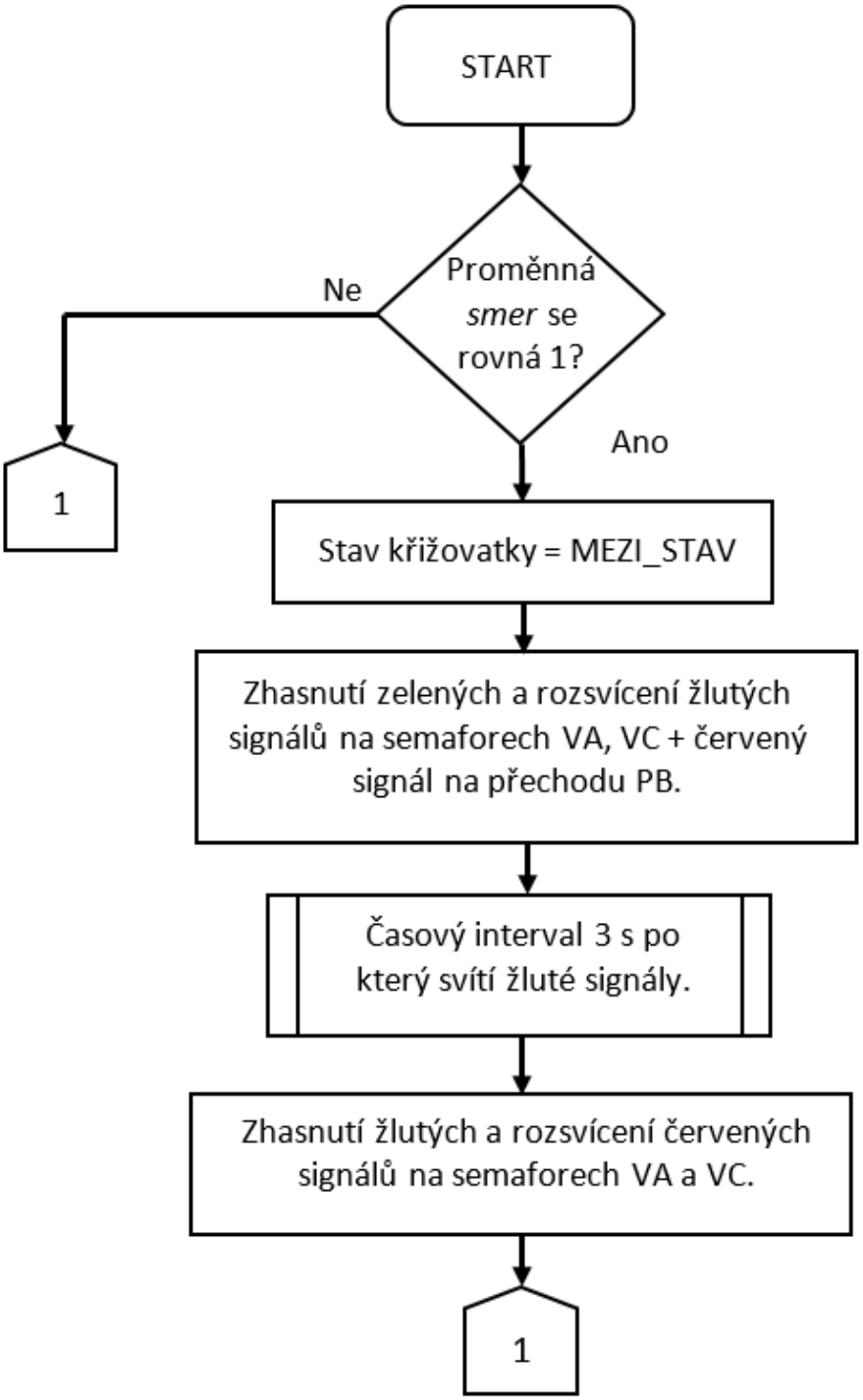
Obrázek 21 – Diagram funkce Nerizeno()



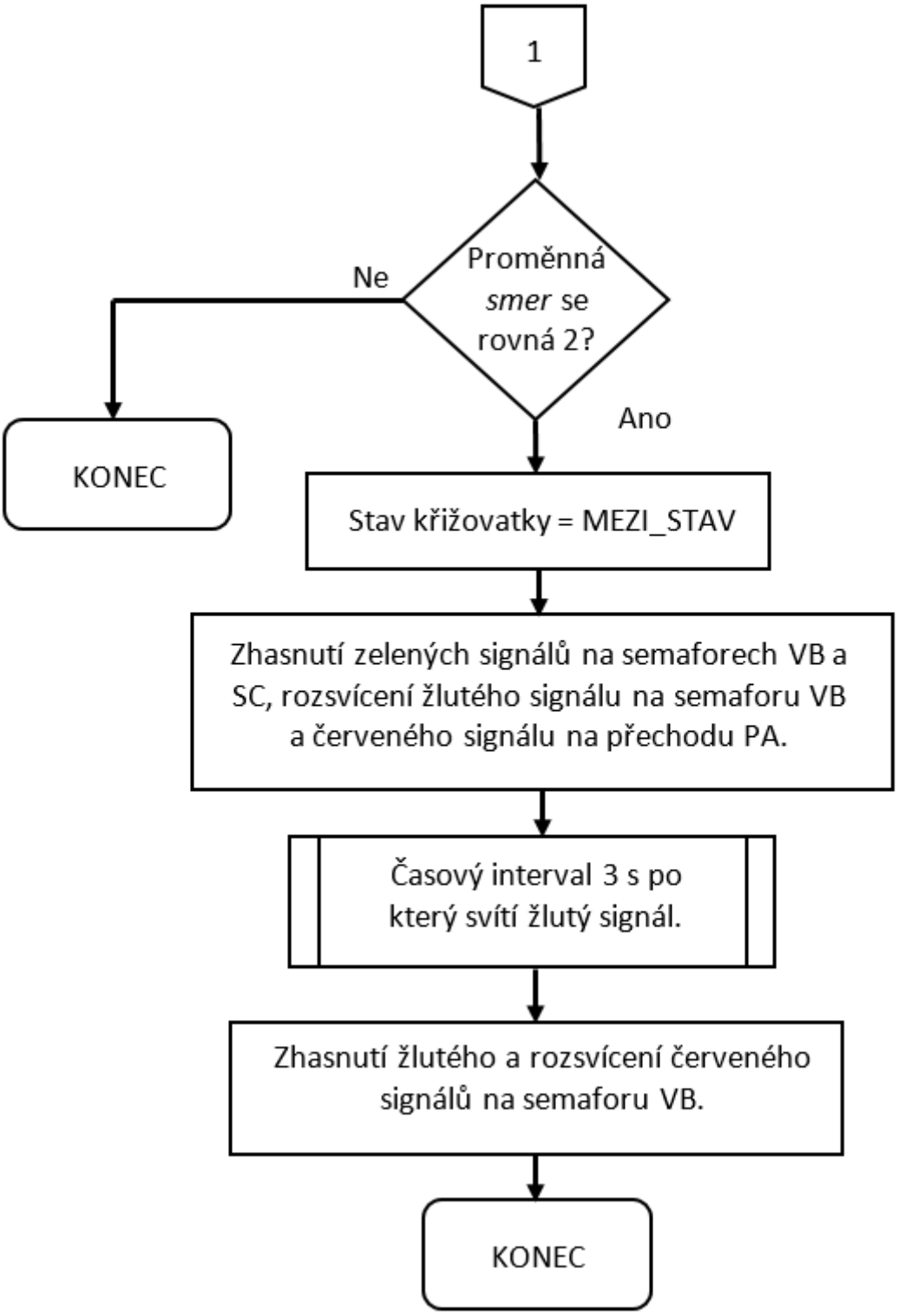
Obrázek 22 – Diagram funkce Rizeno()



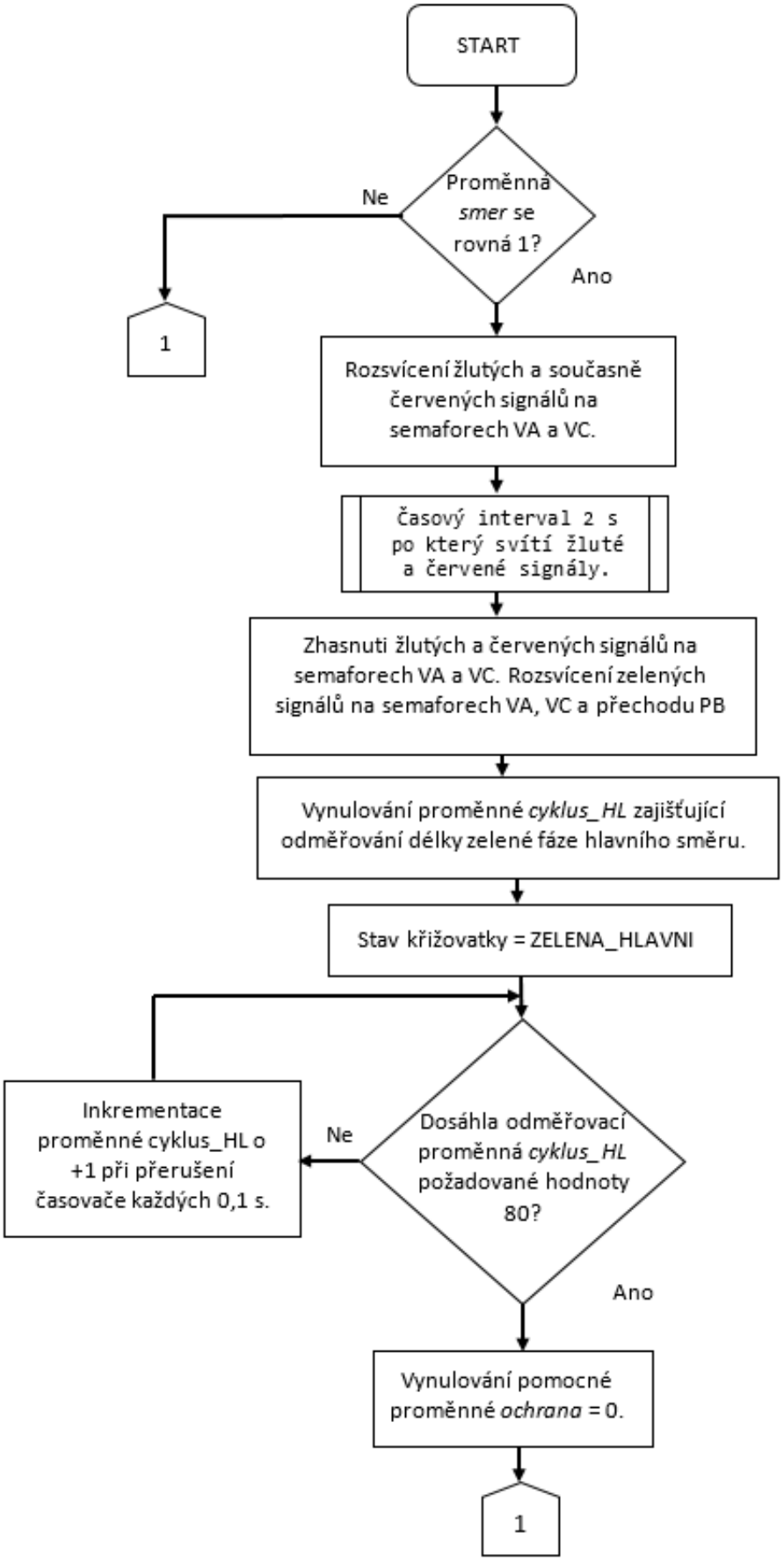
Obrázek 23 – Diagram funkce Cervena_Semafor_Zacatek()



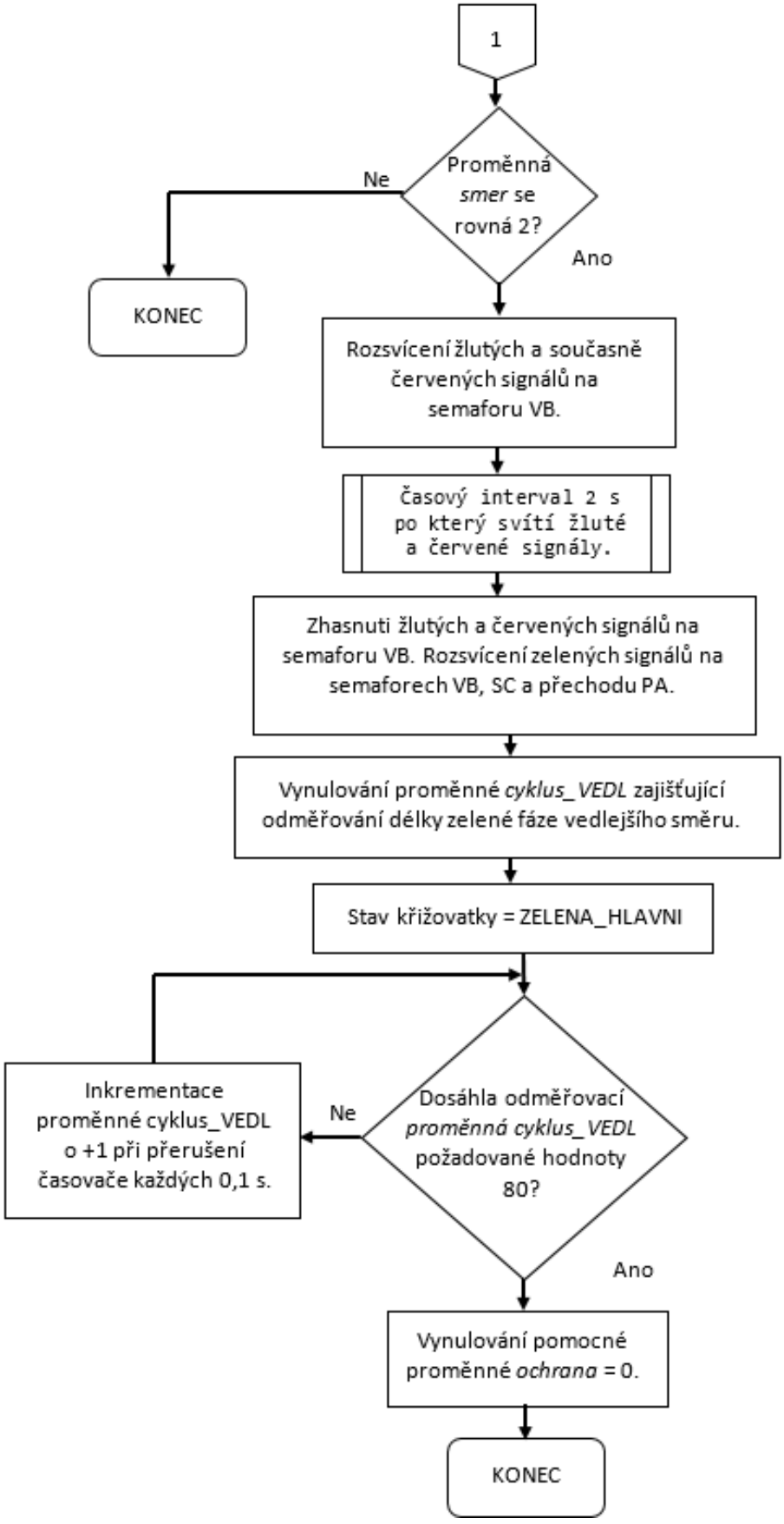
Obrázek 24 – Diagram funkce Cervena_Semafor() 1



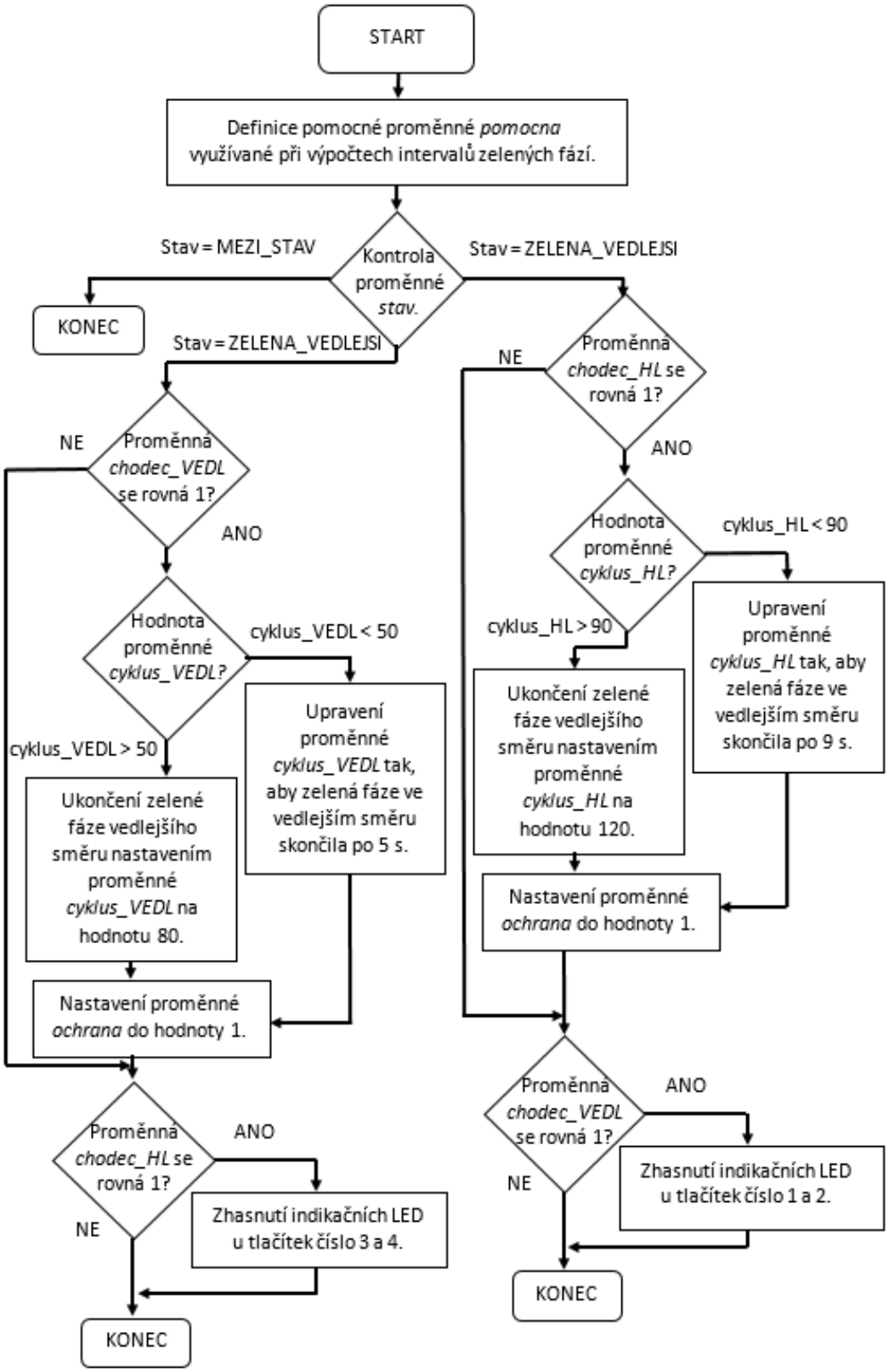
Obrázek 25 – Diagram funkce Cervena_Semafor() 2



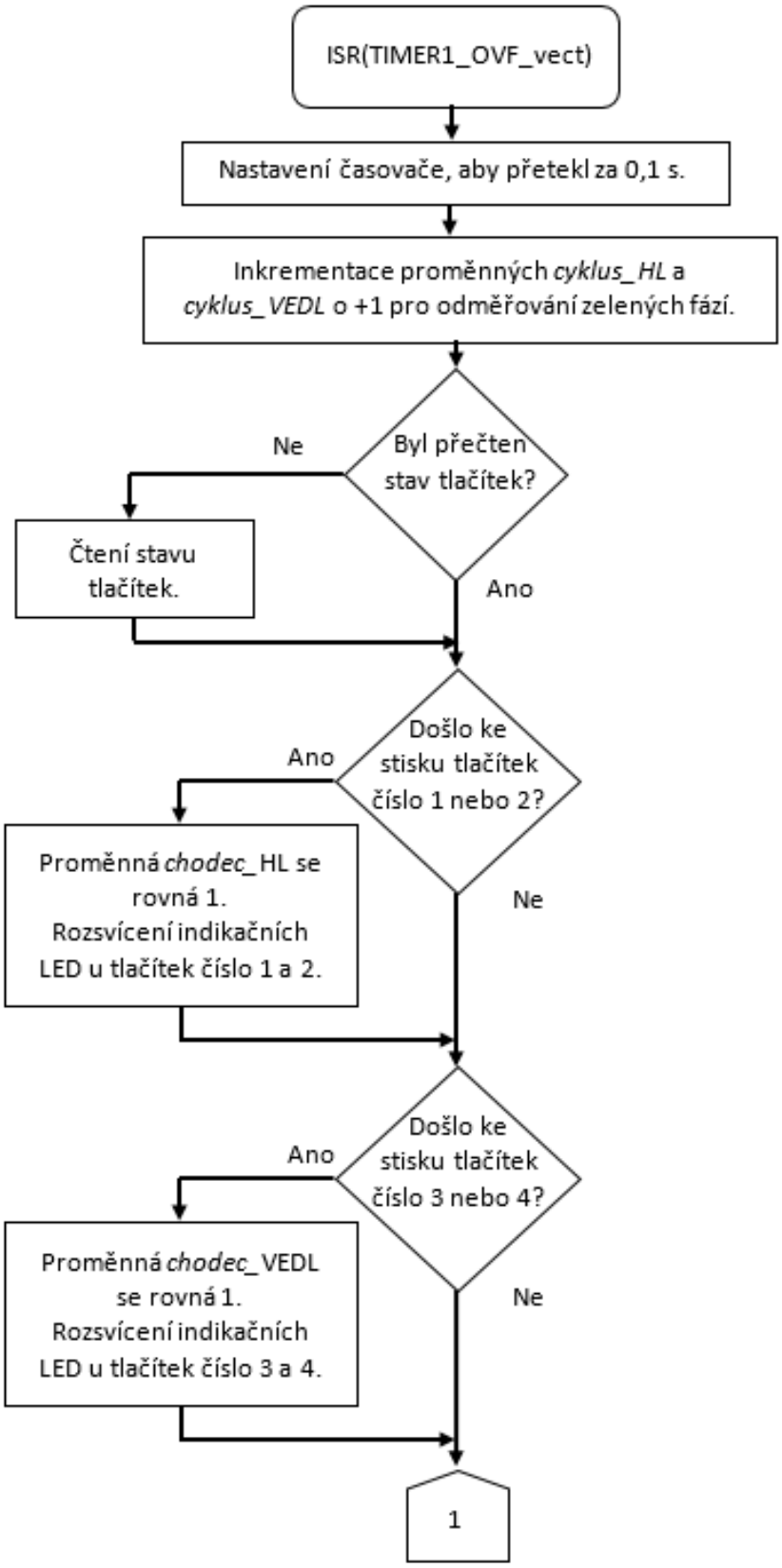
Obrázek 26 – Diagram funkce Zelena_Semafor() 1



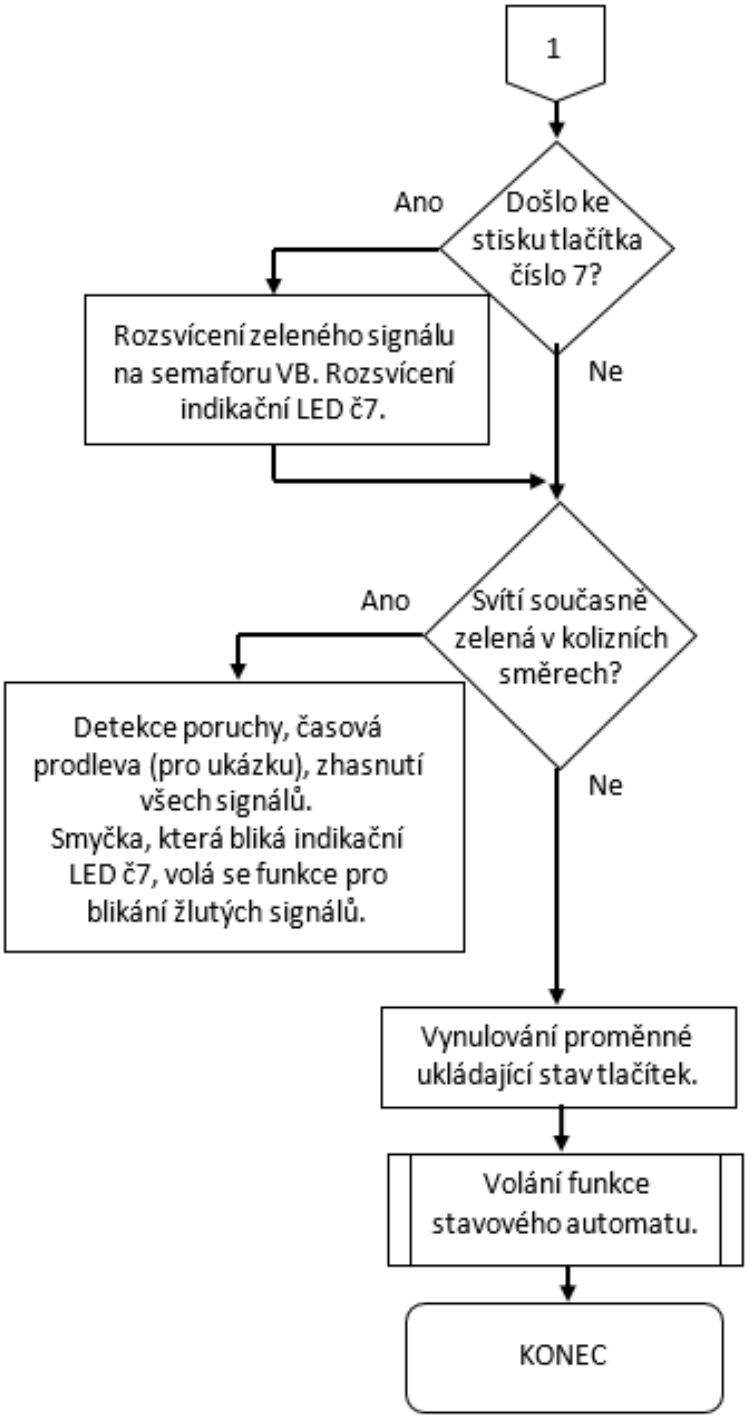
Obrázek 27 – Diagram funkce Zelena_Semafor() 2



Obrázek 28 – Diagram funkce SpustFSM()



Obrázek 29 – Diagram funkce ISR() 1



Obrázek 30 – Diagram funkce ISR() 2

PŘÍLOHA 5 – VYTVOŘENÝ PROGRAM

V této příloze je umístěn kód, který zajišťuje chod celé emulace. Zelenou barvou jsou u jednotlivých řádků přidány vysvětlující popisky.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "tlacitka.h"

//Globalní nastavitelné konstanty pro stanovení vlastností křižovatky. (Hodnoty jsou zadávány v ms)
#define POZOR 3000      //Definice časové konstanty pro signál: Pozor! (3 s)
#define PRIPRAVIT 2000  //Definice časové konstanty pro signál: Pozor připravte se k jízdě. (2 s)

#define CELO_CERVENA_HL_VEDL 3000   //Tato definice slouží pro nastavení délky signálu stůj pro dosažení požadované délky mezičasu při
přechodu fáze z hlavního do vedlejšího směru. (3 s)
#define CELO_CERVENA_VEDL_HL 0      //Tato definice slouží pro nastavení délky signálu stůj pro dosažení požadované délky mezičasu při
přechodu fáze z vedlejšího do hlavního směru. (0 s - U tohoto přechodu fáze není nutný celočervený signál, proto je hodnota nastavena
na 0.

//Nastavení globálních konstant pro zelené fáze. (Hodnoty zadávány v 0,1 s)
#define ZELENA_FAZE_HLAVNI 120    //Zelená fáze pro hlavní směr. (12 s)
#define ZELENA_FAZE_VEDLEJSI 80   //Zelená fáze pro vedlejší směr. (8 s)

#define RELOAD_TIMER1 42496    //Nastavení konstanty časovače, aby přetekl vždy po 100 ms.

//Definice připojení indikačních LED u tlačítek chodců k MCU.
#define TLED_PORT PORTA
#define TLED_DDR  DDRA
//Definice připojení tlačítek chodců na přechodu k MCU.
#define TLAC_PORT PORTB
#define TLAC_DDR  DDRB
#define TLAC_PIN  PINB
```

```
//Definice připojení tlačítek chodců na jednotlivé piny MCU
#define TLAC_PIN_NUM_1 0  //pin 0, TLAC 1   Definice připojení tlačítka č1.
#define TLAC_PIN_NUM_2 1  //pin 1, TLAC 2   Definice připojení tlačítka č2.
#define TLAC_PIN_NUM_3 2  //pin 2, TLAC 3   Definice připojení tlačítka č3.
#define TLAC_PIN_NUM_4 3  //pin 3, TLAC 4   Definice připojení tlačítka č4.
#define TLAC_PIN_NUM_7 6  //pin 6, TLAC 7   Definice připojení tlačítka č7.

//Definice připojení LED semaforů k portu MCU.
#define LED_DDRD    DDRD
#define LED_PORTD   PORTD
#define LED_DDRC    DDRC
#define LED_PORTC   PORTC

//Definice připojení signální skupiny č1 k pinům MCU:
//Vozidlové návěstidlo VA
#define LED_PIN_NUM_VA_R 7
#define LED_PIN_NUM_VA_Y 6
#define LED_PIN_NUM_VA_G 5
//Vozidlové návěstidlo VC
#define LED_PIN_NUM_VC_R 4
#define LED_PIN_NUM_VC_Y 3
#define LED_PIN_NUM_VC_G 2

//Definice připojení signální skupiny č2 k pinům MCU:
//Vozidlové návěstidlo VB
#define LED_PIN_NUM_VB_R 5
#define LED_PIN_NUM_VB_Y 4
#define LED_PIN_NUM_VB_G 3

//Šipka pro odbočení SC
#define LED_PIN_NUM_SC_G 2

//Chodecké návěstidlo PA
#define LED_PIN_NUM_PA_R 1
#define LED_PIN_NUM_PA_G 0
//Chodecké návěstidlo PB
#define LED_PIN_NUM_PB_R 1
#define LED_PIN_NUM_PB_G 0
```

```
//Definice pomocných globálních proměnných
int prvni = 9;      //Proměnná, která slouží na začátku programu k doměřování trvání neřízeného stavu křižovatky. Zadávejte pouze
liché hodnoty!
volatile int tlacitka;        //Proměnná, do které se ukládá stav tlačítek při jejich čtení.
volatile int cyklus_HL = 0;   //Proměnná, pro odměřování intervalu zelené fáze ve vedlejším směru.
volatile int cyklus_VEDL = 0; //Proměnná, pro odměřování intervalu zelené fáze v hlavním směru.
volatile int ochrana = 0;     //Proměnná pro zamezení opakované reakce na stisk tlačítek.
volatile int porucha = 0;     //Proměnná pro indikaci detekce poruchy současného svícení kolizních zelených.

volatile int chodec_HL = 0;   //Proměnná, do které se ukládá přítomnost chodce na přechodu PA, 1 = chodec, 0 = není chodec.
volatile int chodec_VEDL = 0; //Proměnná, do které se ukládá přítomnost chodce na přechodu PB, 1 = chodec, 0 = není chodec.

volatile int hodnota1 = 0xFF; //Proměnné do kterých se budou ukládat stavy indikačních LED tlačítek chodců č1 a č2. V základu jsou
definovány, aby LED nesvítily.
volatile int hodnota2 = 0xFF; //Proměnné do kterých se budou ukládat stavy indikačních LED tlačítek chodců č3 a č4. V základu jsou
definovány, aby LED nesvítily.


//Definice stavového automat - FSM (Finite-state machine)
//Definice stavů, ve kterých se křižovatka v řízeném režimu může nacházet.(Definováno jako nový datový typ Stav_krizovatky)
typedef enum
{
  ZELENA_HLAVNI,
  ZELENA_VEDLEJSI,
  MEZI_STAV
} Stav_krizovatky;

//Definice globální proměnné stavového automatu.
Stav_krizovatky stav;
```

```
//Deklarace funkcí využívaných v programu:

//Deklarace funkce stavového automatu. Do funkce se předává aktuální stav křižovatky.
//Tato funkce řeší reakce na stisk tlačítek chodců.
void SpustFSM(Stav_krizovatky stav);


//Funkce zajišťující blikání žlutými světly při neřízeném stavu křižovatky.
void Nerizeno(void);

//Funkce realizující průběh signálního plánu při řízeném stavu křižovatky.
void Rizeno(void);

//Funkce, která řídí přechod semaforů do zelené fáze.
//Proměnou směr se určuje, který směr má přejít do zelené fáze. Do proměnné stav se ukládá aktuální stav křižovatky.
void Zelena_Semafor(int smer, Stav_krizovatky *stav);

//Funkce, která řídí přechod semaforů do červených signálů.
//Proměnou směr se určuje, který směr má přejít do zelené fáze. Do proměnné stav se ukládá aktuální stav křižovatky.
void Cervena_Semafor(int smer,  Stav_krizovatky *stav);

//Funkce, která zařizuje přechod z blikajících žlutých světel do celočervené. Neboli přechod z neřízeného do řízeného stavu
křižovatky.
void Cervena_Semafor_Zacatek(void);
```

```c
int main(void)  //Hlavní funkce programu, která se spustí jako první po spuštění programu.
{
  LED_PORTC = 0x00;      //Nastavení hodnot pinů portu C s připojenými semafory VB, SC a PB na hodnotu 0, aby nedošlo k jejich
  náhodnému rozsvícení.
  LED_DDRC = (1<<LED_PIN_NUM_VB_R)|(1<<LED_PIN_NUM_VB_Y)|(1<<LED_PIN_NUM_VB_G)|(1<<LED_PIN_NUM_SC_G)|(1<<LED_PIN_NUM_PB_R)
  |(1<<LED_PIN_NUM_PB_G);  //Nastavení používaných pinů na portu C jako výstupních.
  LED_PORTD = 0x00;       //Nastavení hodnot pinů portu D s připojenými semafory VA, VC a PA na hodnotu 0, aby nedošlo k jejich
  náhodnému rozsvícení.
  LED_DDRD = (1<<LED_PIN_NUM_VA_R)|(1<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VA_G)|(1<<LED_PIN_NUM_VC_R)|(1<<LED_PIN_NUM_VC_Y)
  |(1<<LED_PIN_NUM_VC_G)|(1<<LED_PIN_NUM_PA_G)|(1<<LED_PIN_NUM_PA_R); //Nastavení používaných pinů na portu D jako výstupních.

  //Nastavení portů pro tlačítka a indikační LED chodců.
  TLAC_PORT = 0x00;   //Interní pull-upy tlačítek vypnuty. (Tlačítka nejsou stisklá)
  TLAC_DDR = 0x00;    //Nastavení portu tlačítek chodců jako vstupní.
  TLED_PORT = 0xFF;   //Zapnutí pull-upu indikačních LED, aby po startu nesvítily.
  TLED_DDR = 0xFF;    //Nastavení portu indikačních LED jako výstupní.

  //Nastavení časovače Timer/Counter 1
  TCNT1 = RELOAD_TIMER1;  //SW předvolba pro nastavení periody přetečení časovače na 100 ms.
  TCCR1A = 0;             //Režim časovače Timer/Counter 1 - NORMAL
  TIMSK = 1<<TOIE1;       //Povolení přerušení při přetečení časovače.
  TCCR1B = (1<<CS11)|(1<<CS10);  //Nastavení předděličky pro správný chod časovače - přetečení po 100 ms.

  //Nastavení stavu křižovatky do základního stavu.
  stav = MEZI_STAV;

  //Spuštění nekonečné smyčky programu, ve které se spouští signální plán po skončení neřízeného stavu křižovatky.
  while(1)
    {
      if ((prvni == 0) && (porucha != 1))  //Kontrola zda-li skončil neřízený stav křižovatky a nebyla detekována porucha.
      Rizeno();
      else
      Nerizeno();
    }
}
```

```c
//Tato funkce zajišťuje blikání žlutých signálů při neřízeném stavu křižovatky.
void Nerizeno(void)
{
   LED_PORTC ^= (1<<LED_PIN_NUM_VB_Y);
   LED_PORTD ^= (1<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VC_Y);
   _delay_ms(1250);    //Frekvence blikání žlutých LED udávající jak často se změní stav žlutých LED (ms).

   if (porucha != 1)   //Podmínka pro kontrolu detekce chyby.
   {
      prvni--;

      if (prvni == 1)         //Podmínka po jejímž splnění přechází křižovatka do řízeného stavu.
        {
          Cervena_Semafor_Zacatek();        //Volání funkce zajišťující rozsvícení červených signálů na celé křižovatce.
          _delay_ms(CELO_CERVENA_HL_VEDL);  //Časový interval pro dodržení vypočtené hodnoty mezičasu.
          prvni--;       //Pomocná proměnná prvni se sníží na hodnotu 0 -> zajištění spuštění signálního plánu později.
          sei();         //Globální povolení přerušení při přetečení časovače.
        }
   }
}

//Funkce která realizuje signální plán (střídání jednotlivých fází voláním příslušných funkcí).
void Rizeno()
{
   Zelena_Semafor(1,&stav);         //Volání funkce pro zelenou fázi v hlavním směru.
   Cervena_Semafor(1,&stav);        //Volání funkce pro přechod semaforu v hlavním směru do červených signálů.
   _delay_ms(CELO_CERVENA_HL_VEDL); //Zpožďující časový interval zajišťující dodržení hodnoty mezičasu. Během této doby svítí pouze
                                      červené signály. Jeho velikost se nastavuje na začátku programu.
   Zelena_Semafor(2,&stav);         //Volání funkce pro zelenou fázi ve vedlejším směru.
   Cervena_Semafor(2,&stav);        //Volání funkce pro přechod semaforu ve vedlejším směru do červených signálů.
   _delay_ms(CELO_CERVENA_VEDL_HL); //Zpožďující časový interval zajišťující dodržení hodnoty mezičasu. Během této doby svítí pouze
                                      červené signály. Jeho velikost se nastavuje na začátku programu.
}
```

```c
//Funkce zajišťující přechod křižovatky z neřízeného do řízeného stavu. (Přechod z blikajících žlutých do červených signálů na celé křižovatce.)
void Cervena_Semafor_Zacatek()
{
   LED_PORTC = (1<<LED_PIN_NUM_VB_Y);     //Rozsvícení žlutého signálu na semaforu VB.
   LED_PORTD = (1<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VC_Y); //Rozsvícení žlutého signálu na semaforech VA a VC.
   _delay_ms(POZOR);   //Časové zpoždění po které musí žluté signály svítí pro dostatečný čas na reakci řidičů. Jeho hodnota je na začátku definována na 3 s.
   LED_PORTC = (0<<LED_PIN_NUM_VB_Y)|(1<<LED_PIN_NUM_VB_R)|(1<<LED_PIN_NUM_PA_R);
   //Zhasnutí žlutého a rozsvícení červeného signálů na  semafor VB a červeného signálu na přechodu PA.
   LED_PORTD = (0<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VA_R)|(0<<LED_PIN_NUM_VC_Y)|(1<<LED_PIN_NUM_VC_R)|(1<<LED_PIN_NUM_PB_R);
   //Zhasnutí žlutých a rozsvícení červených signálů na semaforech VA a VC a červeného signálu na přechodu PB.
}
//Tato funkce zajišťuje přechod ze zelených do červených signálů v požadovaném směru.
//Hodnotou proměnné smer se určuje, který směr má přejít do červených signálů (1 pro hlavní směr, 2 pro směr vedlejší).
void Cervena_Semafor(int smer, Stav_krizovatky *stav)
{
   if (smer == 1)    //Podmínka pro hlavní směr.
   {
     *stav = MEZI_STAV;   //Nastavení aktuálního stavu křižovatky.
     LED_PORTD = (0<<LED_PIN_NUM_VA_G)|(0<<LED_PIN_NUM_VC_G)|(1<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VC_Y)|(0<<LED_PIN_NUM_PB_G)
     |(1<<LED_PIN_NUM_PB_R); //Zhasnutí zelených a rozsvícení žlutých signálů na semaforech VA a VC. Červena na přechodu PB.
     _delay_ms(POZOR);   //Časové zpoždění po které musí žluté signály svítí pro dostatečný čas na reakci řidičů. Jeho hodnota je na začátku definována na 3 s.
     LED_PORTD = (0<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VA_R)|(0<<LED_PIN_NUM_VC_Y)|(1<<LED_PIN_NUM_VC_R)|(1<<LED_PIN_NUM_PB_R);
     //hasnutí žlutých signálů a rozsvícení červených na semaforech VA,VC.
   }
   if (smer == 2)  //Podmínka pro vedlejší směr.
   {
     *stav = MEZI_STAV;
     LED_PORTC = (0<<LED_PIN_NUM_VB_G)|(0<<LED_PIN_NUM_SC_G)|(1<<LED_PIN_NUM_VB_Y)|(0<<LED_PIN_NUM_PA_G)|(1<<LED_PIN_NUM_PA_R);
     //Zhasnutí zelených signálů na semaforech VB a SC, rozsvícení žlutého signálů na semaforu VB. Červena na přechodu PA.
     _delay_ms(POZOR);   //Časové zpoždění po které musí žluté signály svítí pro dostatečný čas na reakci řidičů. Jeho hodnota je na začátku definována na 3 s.
     LED_PORTC = (0<<LED_PIN_NUM_VB_Y)|(1<<LED_PIN_NUM_VB_R)|(1<<LED_PIN_NUM_PA_R);  //Zhasnutí žlutého a rozsvícení červeného signálu na semaforu VB
   }
}
```

```c
//Tato funkce zajišťuje přechod z červených do zelených signálů v požadovaném směru.
//Hodnotou proměnné smer se určuje, který směr má přejít do zelených signálů (1 pro hlavní směr, 2 pro směr vedlejší).
void Zelena_Semafor(int smer, Stav_krizovatky *stav)
{
   if (smer == 1)  //Podmínka pro hlavní směr.
   {
      LED_PORTD = (1<<LED_PIN_NUM_VA_R)|(1<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VC_R)|(1<<LED_PIN_NUM_VC_Y)|(1<<LED_PIN_NUM_PB_R);
      //Rozsvícení červených a žlutých signálů na semaforu VA a VC.
      _delay_ms(PRIPRAVIT);  //Časové zpoždění po které musí žluté signály svítí pro dostatečný čas na reakci řidičů. Jeho hodnota je
      na začátku definována na 2 s.
      LED_PORTD = (0<<LED_PIN_NUM_VA_R)|(0<<LED_PIN_NUM_VA_Y)|(1<<LED_PIN_NUM_VA_G)|(0<<LED_PIN_NUM_VC_R)|(0<<LED_PIN_NUM_VC_Y)
      |(1<<LED_PIN_NUM_VC_G)|(0<<LED_PIN_NUM_PB_R)|(1<<LED_PIN_NUM_PB_G); //Zhasnutí žlutých a červených signálu, rozsvícení zelených
      na semaforech VA a VC a přechodu PB.
      cyklus_HL = 0;    //Proměnná využívaná pro odměřování délky zelené fáze hlavního směru.
      *stav = ZELENA_HLAVNI;  //Nastavení aktuálního stavu křižovatky.
      do        //Smyčka která probíhá pořád do kola, dokud neskončí interval zelené fáze v hlavním směru.
        {
        } while (cyklus_HL <= ZELENA_FAZE_HLAVNI);  //Kontrola, zda-li se už uběhlý interval rovná konečné délce zelené fáze.
      ochrana = 0;      //Vynulování pomocné proměnné, která zabraňuje reakci na opakovaný stisk tlačítka chodců.
   }

   if (smer == 2)  //Podmínka pro vedlejší směr.
   {
      LED_PORTC = (1<<LED_PIN_NUM_VB_R)|(1<<LED_PIN_NUM_VB_Y)|(1<<LED_PIN_NUM_PA_R); //Rozsvícení červených a žlutých signálů na
      semaforu VB.
      _delay_ms(PRIPRAVIT);    //Časové zpoždění po které musí žluté signály svítí pro dostatečný čas na reakci řidičů. Jeho hodnota
      je na začátku definována na 2 s.
      LED_PORTC = (0<<LED_PIN_NUM_VB_R)|(0<<LED_PIN_NUM_VB_Y)|(1<<LED_PIN_NUM_VB_G)|(0<<LED_PIN_NUM_PA_R)|(1<<LED_PIN_NUM_PA_G)
      |(1<<LED_PIN_NUM_SC_G); //Zhasnutí žlutých a červených signálu, rozsvícení zelených na semaforech VB a SC a přechodu PA.
      cyklus_VEDL = 0;         //Proměnná využívaná pro odměřování délky zelené fáze vedlejšího směru.
      *stav = ZELENA_VEDLEJSI; //Nastavení aktuálního stavu křižovatky.
      do      //Smyčka která probíhá pořád do kola, dokud neskončí interval zelené fáze ve vedlejším směru.
        {
        } while ((cyklus_VEDL <= ZELENA_FAZE_VEDLEJSI)); //Kontrola, zda-li se už uběhlý interval rovná konečné délce zelené fáze.
      ochrana = 0;    //Vynulování pomocné proměnné, která zabraňuje reakci na opakovaný stisk tlačítka chodců.
   }
}
```

```
//Funkce stavového automatu. Je volána z funkce časovače při přetečení (každých 100 ms).
//Dle vstupních hodnot zjišťuje, zda-li byla stisknuta tlačítka chodců a je případně nutné upravit časový interval zelených fází.
//Časové intervaly upravuje vždy dle aktuálních hodnot již běžících časových intervalů.
//Zelenou fázi v hlavním směru, v případě detekce chodce, zkracuje na 9 s nebo ji rovnou ukončuje, pakliže bylo tlačítko stisknuto až
po 9 s již běžící zelené fáze.
//Zelenou fázi ve vedlejším směru, v případě detekce chodce, zkracuje na 5 s nebo ji rovnou ukončuje, pakliže bylo tlačítko stisknuto
až po 5 s již běžící zelené fáze.
//Zároveň zajišťuje zhasnutí indikačních LED v případě, že je chodcům umožněno přecházet. (Svítí zelený signál na příslušném
přechodu).
//Rozhodnutí o tom, která z činností se má vykonat provádí na základě stavu, ve kterém se křižovatka nachází
(ZELENA_HLAVNI/ZELENA_VEDLEJSI/MEZI_STAV)
// a pomocí globálních proměnných, které indikují přítomnost chodců (chodec_HL a chodec_VEDL).
void SpustFSM(Stav_krizovatky stav)
 {
    int pomocna = 0;     //Definice pomocné proměnné využívané při výpočtu potřebného zbylého času zelených fází.
    switch (stav)
     {
     case ZELENA_HLAVNI:    //Případ kdy se křižovatka nachází ve stavu, kdy svítí zelená v hlavním směru.
        if (chodec_HL == 1)    //Kontrola proměnné, která indikuje přítomnost chodce v hlavním směru.
        {
         if ((cyklus_HL < 90) && (ochrana != 1))  //Kontrola zbylého času zelené fáze a vypočtení kolik času ještě musí trvat do
         stanovených 9 s.
           {
              pomocna = 90 - cyklus_HL;
              cyklus_HL = ZELENA_FAZE_HLAVNI - pomocna;
              ochrana = 1;
           }
         else if ((cyklus_HL >= 90) && (ochrana != 1)) //Případ, kdy už zbylý čas zelené fáze překročil požadovaných 9 s -> u
         končení zelené fáze.
           {
              cyklus_HL = ZELENA_FAZE_HLAVNI;
              ochrana = 1;
           }
        }
```

```
            if (chodec_VEDL == 1)     //Kontrola proměnné, která indikuje přítomnost chodce ve vedlejším směru.
            {
               TLED_PORT = hodnota1;  //Zhasnutí indikačních LED č. 3 a 4 ve vedlejším směru, jelikož je teď umožněno chodcům přecházet.
               hodnota2 = 0xFF;       //Nastavení pomocné proměnné indikačních LED č. 3 a 4 do základního stavu (LED nesvítí).
               chodec_VEDL = 0;       //Vynulování proměnné indukující přítomnost chodce.
            }
         break;

         case ZELENA_VEDLEJSI:     //Případ kdy se křižovatka nachází ve stavu, kdy svítí zelená ve vedlejším směru.
            if (chodec_VEDL == 1)  //Kontrola proměnné, která indikuje přítomnost chodce ve vedlejším směru.
            {
              if ((cyklus_VEDL < 50) && (ochrana != 1))   //Kontrola zbylého času zelené fáze a vypočtení kolik času ještě musí trvat do
              stanovených 5 s.
                {
                   pomocna = 50 - cyklus_VEDL;
                   cyklus_VEDL = ZELENA_FAZE_VEDLEJSI - pomocna;
                   ochrana = 1;
                }
              else if ((cyklus_VEDL >= 50) && (ochrana != 1))  //Případ, kdy už zbylý čas zelené fáze překročil požadovaných 5 s ->
              ukončení zelené fáze.
                {
                   cyklus_VEDL = ZELENA_FAZE_VEDLEJSI;
                   ochrana = 1;
                }
            }
            if (chodec_HL == 1)    //Kontrola proměnné, která indikuje přítomnost chodce v hlavním směru.
            {
               TLED_PORT = hodnota2;  //Zhasnutí indikačních LED č. 1 a 2 v hlavním směru, jelikož je teď umožněno chodcům přecházet.
               hodnota1 = 0xFF;       //Nastavení pomocné proměnné indikačních LED č. 1 a 2 do základního stavu (LED nesvítí).
               chodec_HL = 0;         //Vynulování proměnné indukující přítomnost chodce.
            }
         break;
         case MEZI_STAV:   //Případ kdy se křižovatka nachází ve stavu, kdy na semaforech svítí červené či žluté signály nebo kombinace
                           obojího. V tomto případě stavový automat nic nevykonává a rovnou se ukončuje.
         break;

   }
}
```

```
//Funkce, která se volá pokaždé při přetečení časovače Timer/Counter1, což se dle nastavení děje každých 100 ms.
//Funkce slouží pro odměřování času zelených fází, čtení stavu tlačítek chodců a hlídání zelených signálů kolizních směrů.
ISR(TIMER1_OVF_vect)
  {
     TCNT1 = RELOAD_TIMER1;  //Nastavení čítacího registru, aby přetekl za 100 ms.
     cyklus_HL++;      //Zvýšení proměnné pro odměřování délky zelené fáze v hlavním směru o +1.
     cyklus_VEDL++;    //Zvýšení proměnné pro odměřování délky zelené fáze ve vedlejším směru o +1.

     if (tlacitka == 0x00)   //Kontrola přečtení stavu tlačítek.
     {
        tlacitka = cti_tlacitka();  //Čtení stavu tlačítek.
     }

     if (((tlacitka & (1<<TLAC_PIN_NUM_1)) != 0)||((tlacitka & (1<<TLAC_PIN_NUM_2)) != 0)) //Kontrola, zda-li došlo ke stisku
     tlačítka č. 1 nebo 2.
     {
        chodec_HL = 1;    //Nastavení proměnné do hodnoty 1, značící přítomnost chodce v hlavním směru.
        if (stav != ZELENA_VEDLEJSI) //Podmínka před rozsvícením indikačních LED pro zamezení jejich rozsvícení, když už je chodcům
        umožněno přecházet.
        {
           hodnota1 = 0b11111100;
           TLED_PORT = hodnota2 & (~(1<<TLAC_PIN_NUM_2 | 1<<TLAC_PIN_NUM_1)); //Rozsvícení indikačních LED u tlačítek č. 1 a 2.
        }
     }

     if (((tlacitka & (1<<TLAC_PIN_NUM_3)) != 0)||((tlacitka & (1<<TLAC_PIN_NUM_4)) != 0)) //Kontrola, zda-li došlo ke stisku
     tlačítka č. 3 nebo 4.
     {
        chodec_VEDL = 1;    //Nastavení proměnné do hodnoty 1, značící přítomnost chodce ve vedlejším směru.
        if (stav != ZELENA_HLAVNI)  //Podmínka před rozsvícením indikačních LED pro zamezení jejich rozsvícení, když už je chodcům
        umožněno přecházet.
        {
           hodnota2 = 0b11110011;
           TLED_PORT = hodnota1 & (~(1<<TLAC_PIN_NUM_3 | 1<<TLAC_PIN_NUM_4)); //Rozsvícení indikačních LED u tlačítek č. 3 a 4.
        }
     }
```

```
    if ((tlacitka & (1<<TLAC_PIN_NUM_7)) != 0)  //Kontrola, zda-li došlo ke stisku tlačítka č. 7 sloužícího pro simulaci poruchy.
    {
        LED_PORTC = (1<<LED_PIN_NUM_VB_R)|(1<<LED_PIN_NUM_VB_G)|(1<<LED_PIN_NUM_PA_R);//Rozsvícení červené a zelené na semaforu VB +
        červená na přechodu PA
        TLED_PORT = (~(1<<TLAC_PIN_NUM_7));           //Rozsvícení indikační LED u tlačítka č. 7.
    }

    if (((LED_PORTC & (1<<LED_PIN_NUM_VB_G)) != 0) && ((LED_PORTD & (1<<LED_PIN_NUM_VA_G)) != 0)) //Kontrola pinů kolizních
    zelených. V případě že by měly svítí obě zelené naráz -> detekce chyby.
    {
        porucha = 1;   //Nastavení indikační proměnné pro chybu na hodnotu 1
        _delay_ms(1500);//Časové zpoždění pro ukázku, že opravdu svítí oba zelené signály naráz. Při normálním chodu by toto zpoždění
        zde nebylo.
        LED_PORTC = 0x00; //Zhasnutí všech signálů na semaforech pro vozidla i chodce připojených k portu C.
        LED_PORTD = 0x00; //Zhasnutí všech signálů na semaforech pro vozidla i chodce připojených k portu D.
        _delay_ms(1250);//Časové zpoždění, kdy nesvítí žluté signály.
        do   //Nekonečná smyčka která bliká indikační LED č. 7 indikující detekci poruchy. Zároveň volá funkci pro blikání žlutých
        signálů.
        {
            TLED_PORT ^= (1<<TLAC_PIN_NUM_7);
            Nerizeno();
        } while (porucha == 1);
    }

    tlacitka = 0x00;//Vynulování proměnné do které se uložil stav tlačítek.
    SpustFSM(stav); //Volání funkce stavového automatu.
}
```