ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Brána sběrnice CAN do počítačové sítě pro simulátor automobilu
Student:	Tomáš Čermák
Vedoucí:	Ing. Miroslav Skrbek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Proveďte rešerši existujících řešení bran sběrnice CAN do počítačové sítě pro přenos CAN rámců přes TCP/IP nebo UDP protokoly. Vyberte vhodné, případně navrhněte nové, zapouzdření CAN rámců do standardních síťových protokolů. Dále prozkoumejte existující simulátory automobilů s možností doplnění programových modulů pro vzdálené ovládání přes síťové rozhraní. S využitím existujících modulů a knihoven implementujte síťovou bránu pro sběrnici CAN a modul do simulátoru tak, aby byl simulátor ovládán rámci sběrnice CAN. Uvažujte možnost, že více instancí synchonizovaně řízených simulátorů poběží na více počítačích s různě nastavenými pohledy do scény. Bránu i modul implementujte pro OS Linux. Rozsah implementace upřesněte po dohodě s vedoucím práce.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 6. prosince 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Bakalářská práce

Brána sběrnice CAN do počítačové sítě pro simulátor automobilu

Tomáš Čermák

Vedoucí práce: Ing. Miroslav Skrbek, Ph.D.

15. května 2017

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Ing. Miroslavu Skrbkovi, Ph.D. za trpělivost a ochotu mi pomoci při zpracování této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Čermák, Tomáš. *Brána sběrnice CAN do počítačové sítě pro simulátor automobilu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se zabývá vytvořením brány sběrnice CAN do počítačové sítě. Dále úpravou automobilového simulátoru Speed Dreams umožňující ovládání ze sběrnice CAN pomocí vytvořené brány a běhu několika instancí s různými pohledy do scény synchronizovaných po síti. K vytvoření brány je využito převodníku USB2CAN od firmy CANLAB s. r. o. a vývojové desky Digilent ZYBO. Ovládání simulátoru přes sběrnici CAN je řešeno pomocí naprogramovaného řidiče v simulátoru, který reaguje na příchozí zprávy z brány. Pro běh s různými pohledy je potřeba upravit již existující grafický modul simulátoru. Výsledkem návrhu a implementace je aplikace brány v jazyce C pro operační systém Linux a patch pro simulátor umožňující zmíněné možnosti. V příloze práce lze nalézt zdrojové kódy aplikace, simulátor a patch pro simulátor.

Klíčová slova sběrnice CAN, brána sběrnice CAN do počítačové sítě, počítačová síť, Speed Dreams, simulátor automobilu, různé pohledy do scény, synchronizace po síti, aplikace, OS Linux

Abstract

This bachelor thesis deals with the creation of a CAN bus to computer network gateway. Also, changes of car simulator Speed Dreams to make it able to be controlled from CAN bus using the created gateway and multiple running instances with different views into the scene synchronized over the network. To create the gate, the converter USB2CAN from the company CANLAB s. r. o and development board Digilent ZYBO were used. Simulator's control using CAN bus is realized with programmed driver in the simulator which respond to incoming messages from the converter. For the run with multiple views, changes to the already existing graphic module are needed. The result of a design and implementation is the application of the gateway in C programming language for operating system Linux and patch for the simulator enabling mentioned possibilities. Source codes, simulator and patch for the simulator can be found in the attachment.

Keywords CAN bus, CAN bus to computer network gateway, computer network, Speed Dreams, car simulator, different views into the scene, synchronization over the network, application, OS Linux

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Užitím sběrnice CAN je v současnosti komunikace řídících jednotek zařízení (v našem případě automobilu) s monitorovacími senzory. Tato sběrnice se stala součástí simulátorů, aby byly schopné vymodelovat reálnější situace, které mohou řidiče, případně piloty potkat při výkonu jejich profese.

Práce se zabývá problematikou vytvoření brány sběrnice CAN do počítačové sítě pomocí převodníku USB2CAN a následné propojení existujícího simulátoru automobilu se sběrnicí CAN tak, aby bylo možné simulované vozidlo řídit pomocí zpráv generovaných na této sběrnici. Dále vytvoření patche pro simulátor, aby bylo možné synchronizovaně ovládat více instancí simulátoru s různými pohledy do scény, běžících na jednom nebo více počítačích. Tato práce bude poskytovat podklady pro výuku na ČVUT FIT.

Práce se zabývá rešerší existujících simulátorů a bran. Dále se zabývá úpravami simulátoru a vytvořením samotné brány.

Nejprve se v první části věnuji rešerší simulátorů, sběrnice CAN, počítačové sítě, převodníkem USB2CAN a bran. Ve druhé části uvedu návrh úprav pro simulátor a návrh realizace brány. Ve třetí části představím realizaci změn pro simulátor a implementaci brány. Čtvrtá část se věnuje testování pomocí herního volantu a vývojové desky Digilent ZYBO.

Tato práce navazuje na bakalářskou práci studenta Pavla Zajíce, která se zabývá propojením herního volantu se sběrnicí CAN.

Obrázek 0.1: Monitor vpravo je připojen k laptopu, zatímco ostatní monitory jsou zapojeny k stolnímu počítači

Kapitola 1

Cíl práce

Cílem rešeršní části této práce je seznámení se s již existujícími simulátory. Dále pak zpřehlednění funkce sběrnice CAN a počítačové sítě. Dalším cílem je ukázat si již vytvořené řešení bran sběrnice CAN do sítě.

V praktické části se pak věnuji navržení a uskutečnění brány, která vhodně zapouzdří jednotlivou zprávu sběrnice CAN do síťového protokolu a pak ji vyšle do počítačové sítě. Brána by měla tento proces dokázat provést i obráceně tj. zprávu počítačové sítě vypouzdřit a vyslat na sběrnici CAN.

Jako součást praktického řešení jsem si vytkl navrhnout a implementovat úpravy pro simulátor, které umožní simulované vozidlo ovládat pomocí zpráv příchozích z brány.

Dále pak navrhuji koncepci a realizaci úprav pro simulátor, které přidají možnost běhu více instancí s různými pohledy do scény synchronizovaně řízených na jednom nebo více počítačích.

Součástí je rovněž popis testu praktického fungování celku.

Kapitola 2

Analýza

Předpokladem pro vlastní zpracování zadání je provedení analýzy již známých řešení problematiky. Současně pak provedení výběru komponent s přihlédnutím na jejich technickou úroveň a dostupnost na trhu. Mým požadavkům na základě níže uvedených hodnocení nejlépe vyhovují: simulátor Speed Dreams a převodník USB2CAN.

2.1 Simulátor

Internet nabízí hromadu různých řešení simulátoru automobilu. Většina z nich jsou závodní hry ve kterých se ukazuje propracovaná fyzika. Na simulátorech nás zajímají poskytlé informace o vozidle a použité knihovny. Dále se podíváme na možnosti rozšíření kvůli úpravám umožňujícím synchronizaci instancí. Nakonec se seznámíme se systémovými požadavky pro OS Linux, aby jsme se ujistili, že dnešní počítače budou schopné běhu více instancí najednou. Informace o vozidle zohledňuji z důvodu možnosti pro budoucí práce, které se mohou zabývat připojením k již existujícímu modelu palubní desky vytvořené v bakalářské práci studenta Petra Sochy.

2.1.1 TORCS

„TORCS je vysoce portabilní multi platformní závodní simulátor. Obyčejně používaný jako závodní hra, jako AI zavodní hra a jako rešeršní platforma. Běží na Linuxu (všechny architektury, 32 i 64 bit, little i big endian), FreeBSD, OpenSolaris, MacOSX a Windows (32 i 64 bit).“[1] Simulátor poskytuje informace o rychlosti, otáčkách motoru, zařazené rychlosti, stavu paliva, poškození vozidla, stavu kol a stavu analogového vstupu ovládání.

2.1.1.1 Použité knihovny:

- FreeGLUT
  - Sada nástrojů pro OpenGL
- Plib
  - Portability Libraries: Run/time package
- OpenAL
  - Sada nástrojů OpenAL

2.1.1.2 Možnosti rozšíření:

TORCS umožňuje úpravou XML souborů měnit či přidávat tratě a vozidla. Dovoluje programování vlastních robotů (AI i člověk se zde bere jako robot). Dále umožňuje vytvoření vlastních modulů, které mají k dispozici hlavičkové soubory pro práci např. s vozidlem a kamerou. Simulátor již obsahuje moduly pro vykreslení grafiky, výpočet fyziky...

Listing 2.1: Ukázka nastavení motoru v příslušném XML souboru

```
1  <section name="Engine">
2    <attnum name="inertia" min="0.1" max="0.5" unit="kg.m2"
         val="0.2423"/>
3    <attnum name="revs limiter" min="3000" max="10000" unit="
       rpm" val="8500"/>
4    <attnum name="revs maxi" min="3000" max="10000" unit="rpm"
        val="10000"/>
5    <attnum name="tickover" min="500" max="5000" unit="rpm"
         val="1000"/>
6    <attnum name="fuel cons factor" min="0.5" max="5.0" val="
       1.2"/>
7    <section name="data points">
8      <section name="1">
9        <attnum name="rpm" min="100" max="10000" unit="rpm"
             val="1000"/>
10       <attnum name="Tq" min="10" max="600" unit="N.m" val="
             357"/>
11     </section>
12
13     <section name="2">
14       <attnum name="rpm" min="100" max="10000" unit="rpm"
             val="2000"/>
15       <attnum name="Tq" min="10" max="600" unit="N.m" val="
             357"/>
16     </section>
17   </section>
18 </section>
```

Obrázek 2.1: Simulátor TORCS

2.1.1.3 Systémové požadavky:

Tabulka 2.1: Systémové požadavky simulátoru TORCS pro OS Linux[2]

	Minimum	Doporučené
CPU	550 MHZ	800 MHZ
Paměť	128 MB	256 MB
Harddisk	200 MB	200 MB
Grafická karta	OpenGL 1.3 kompatibilní s 32 MB RAM	OpenGL 1.3 kompatibilní s 64 MB RAM

2.1.2 Speed Dreams

„Původně vychází z TORCS a dnes již obsahuje mnohem reálnější vzhled a simulaci fyziky, díky aktivnímu vývojářskému týmu a rostoucí komunitě.“[3] Speed Dreams poskytuje stejné informace jako simulátor TORCS.

2.1.2.1 Použité knihovny navíc oproti TORCS:

- openscenegraph
  - Výkonný real-time grafický toolkit

- sdl2
  - Portabilní low-level přístup k video framebufferu, audio výstupu, myši a klávesnice.
  - Možnost práce s vlákny a sdílenými prostředky
- enet
  - Relativně tenká, jednoduchá a robustní síťová komunikační vrstva nad UDP

2.1.2.2 Možnosti rozšíření:

Možnosti rozšíření simulátoru jsou stejné jako u simulátoru TORCS.

2.1.2.3 Systémové požadavky:

Tabulka 2.2: Systémové požadavky simulátoru Speed Dreams pro OS Linux[4]

	Minimum	Doporučené
CPU	1,5 GHZ	2 GHZ
Paměť	1,5 GB	2GB
Harddisk	1GB	1GB
Grafická karta	OpenGL 1.3 kompatibilní s 128 MB RAM	OpenGL 1.5 kompatibilní s 256 MB RAM

Obrázek 2.2: Simulátor Speed Dreams

2.1.3 VDrift

„VDrift je cross-platformní, open source řídící simulátor vytvořen s myšlenkou driftování.“[5] Simulátor nabízí otáčkoměr, rychloměr a informaci o zařazené rychlosti.

2.1.3.1 Použité knihovny:

- Bullet
  - Knihovna pro 3D kolizní detekci a tuhou tělesnou dynamiku pro hry a animace
- sdl2
  - Portabilní low-level přístup k video framebufferu, audio výstupu, myši a klávesnice.
  - Možnost práce s vlákny a sdílenými prostředky
- glew
  - The OpenGL Extension Wrangler Library

2.1.3.2 Možnosti rozšíření:

Simulátor umožňuje úpravou konfiguračních souborů měnit či přidávat tratě a vozidla.

Listing 2.2: Ukázka nastavení motoru v příslušném konfiguračním souboru

```
1 [engine]
2 position = 0.0, 1.4, -0.1
3 mass = 200.0
4 displacement = 2.3E-3
5 max-power = 119360
6 peak-engine-rpm = 6500
7 rpm-limit = 8000.0
8 inertia = 0.2
9 idle = 0.025
10 start-rpm = 1000
11 stall-rpm = 500
12 fuel-consumption = 0.0001
13 torque-curve-00 = 1000, 149
14 torque-curve-01 = 1500, 158.5
15 torque-curve-02 = 2000, 168
```

2.1.3.3 Systémové požadavky:

Tabulka 2.3: Systémové požadavky simulátoru VDrift pro OS Linux[6]

	Minimum	Doporučené
CPU	1 GHZ	2 GHZ
Paměť	512 MB	1GB
Harddisk	700 MB	700 MB
Grafická karta	OpenGL 2 kompatibilní	OpenGL 3.3 kompatibilní

Obrázek 2.3: Simulátor VDrift

2.1.4 T1 Car Racing Simulator, CarWorld, GRacer, SuperTuxKart, Vamos

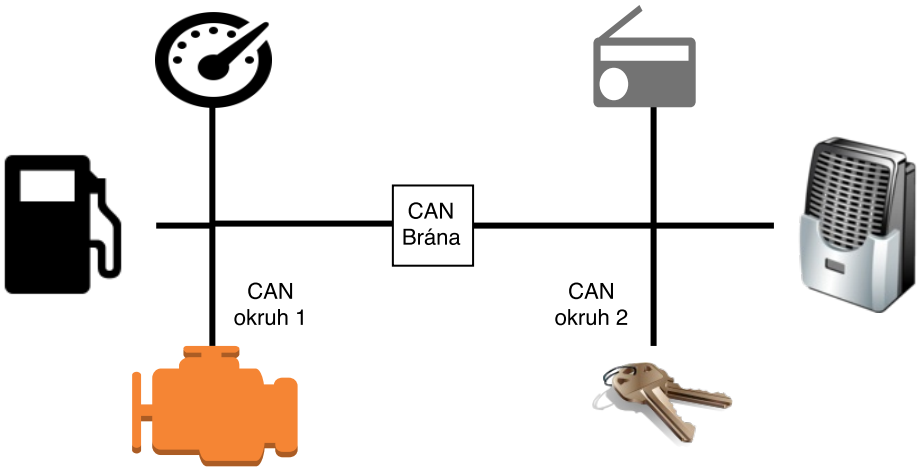
Tyto projekty jsou buď již zastaralé nebo se k vypracování práce nedají použít kvůli vzhledu nebo své dokumentaci.

2.1.5 Závěr

Pro naší práci jsem si vybral simulátor Speed Dreams. Důvodem je široká možnost rozšíření a také použití síťové knihovny. Systémové požadavky jsou sice vyšší, ale předpokládám běh maximálně tří instancí souběžně na jednom počítači, proto se domnívám, že nepůjde o problém. Zapouzdření řidičů do naprogramovaných robotů, které dává větší přehled o funkcích robota minimalizuje pravděpodobnost vytvoření chyb v celkovém programu.

2.2 Sběrnice CAN

CAN sběrnice je používaná nejčastěji pro komunikaci senzorů a řídících jednotek v automobilu. Jedná se o sériovou datovou sběrnici.



Obrázek 2.4: Ukázka možného zapojení CAN sběrnice ve vozidle

2.2.1 Síťový protokol

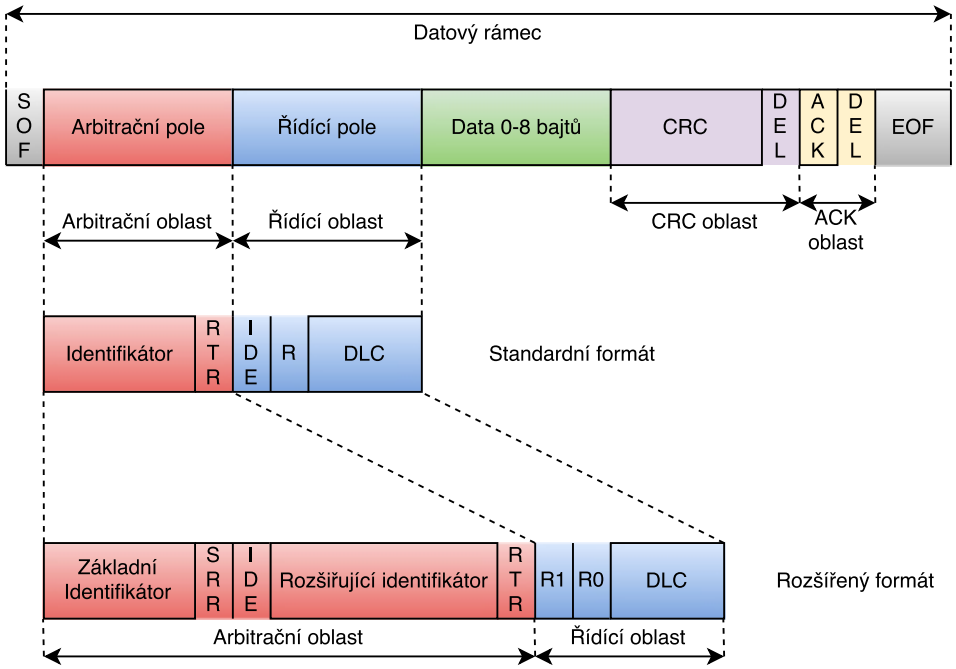
Protokol umožňuje odesílání až osmi datových bajtů. CAN rámec obsahuje identifikátor, který definuje obsah zprávy.

2.2.2 Přenos dat

CAN sběrnice využívá arbitrační metodu, která požaduje, aby všechny zařízení byly synchronizovány a připraveny ke vzorkování ve stejnou dobu. Komununikace mezi zařízeními je řešena asynchronním přenosem, tudíž zařízení musí znát rychlost datové linky. Sběrnice má definovanou dominantní (binární 0) a recesivní úroveň (binární 1).

2.2.3 Arbitrace

Každé zařízení obsahuje unikátní identifikátor v rámci sběrnice. „CAN protokol podporuje dva rámcové formáty zpráv. Podstatný rozdíl je v délce označení (ID identifier). Ve standardním formátu je délka ID 11 bitů a v prodlouženém formátu (extended formát) je délka označení zprávy 29 bitů. Rámec zprávy pro přenos na sběrnici se skládá ze sedmi hlavních polí.“[7] Pokud zařízení začnou vysílat ve stejnou dobu, tak zpráva s nižší hodnotou arbitračního pole vítězí, protože ostatní zařízení vidí na sběrnici dominantní úroveň i když vysílají recesivní.

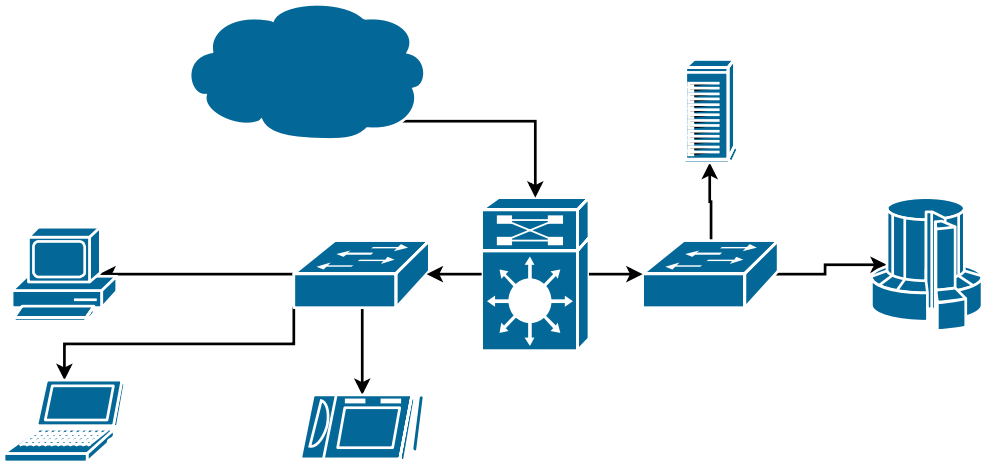


Obrázek 2.5: CAN rámec

- RTR bit značí tzv. Remote frame, který říká zařízení, že chce od něho data
- SRR bit musí být na recesivní úrovni, zachovává vzájemnou kompatibilitu obou formátů
- R bit je rezervovaný bit
- DLC pole značí počet datových bajtů v rámci

2.3 Počítačová síť

Počítačová síť slouží k výměně informací mezi počítači. Největší počítačovou sítí je celosvětová síť Internet, která využívá sadu protokolů TCP/IP. Sítě lze klasifikovat podle velikosti sítě, vlastnictví, podle přepojování nebo podle postavení jednotlivých uzlů. Počítačové sítě taktéž obsahují různé aktivní prvky. Toto téma je velmi rozsáhlé, proto se podíváme specificky na paketovou komunikaci mezi jednotlivými zařízeními. Paket je blokem dat posílaným po síti.



Obrázek 2.6: Ukázka počítačové sítě

Obrázek 2.7: RJ-45, nejčastěji používaná koncovka síťových kabelů

2.3.1 Postavení programů

Komunikaci mezi programy ovládá buď jeden (Server-Klient), nebo všichni (Peer-to-Peer).

2.3.1.1 Peer-to-peer

Neboli taktéž „rovný s rovným“. Toto postavení se používá například u sdílení prostředků, jako je třeba protokol bittorent.

2.3.1.2 Server-Klient

Server je nadřazen klientům. Server nabízí služby klientům. Poskytované služby jsou například: souborový server, email server, webový server...

2.3.2 Adresace protokolů TCP a UDP

Paket obsahuje zdrojovou a cílovou MAC i IP adresu. Díky těmto adresám paket doputuje do cílového zařízení, které díky zdrojové adrese ví s kým komunikuje. Tento model by nám stačil, pokud bychom chtěli současně pouze jednu komunikaci mezi danými zařízeními, proto paket obsahuje zdrojový a cílový port. Toto číslo umožňuje operačnímu systému adresovat zprávu aplikaci. To nám například dovoluje ve webovém prohlížeči otevíraní několika záložek najednou. Číslo je v rozmezí 0-65535, kde 0-1023 jsou vyhrazené porty pro nejběžnější služby a 1024-49151 jsou registrované porty. Jiné protokoly nejsou k bakalářské práci potřeba.

2.3.2.1 Známé porty služeb

Tabulka 2.4: Známé porty služeb

Port	Protokol	Popis
20	FTP	Přenos dat
22	SSH	Vzdálený textový terminál, šifrovaná komunikace
23	Telnet	Vzdálený textový terminál, nešifrovaná komunikace
25	SMTP	Přenos emailu
53	DNS	Překlad doménových jmen na IP adresy a zpět
80	HTTP	Přenos webových stránek a jiných dat nešifrovaně
110	POP3	Stahování emailu
143	IMAP	Vzdálená správa emailové schránky
161	SNMP	Správa sítě
443	HTTPS	Přenos webových stránek a jiných dat šifrovaně

2.3.2.2 TCP

Protokol garantuje spolehlivý přenos dat ve správném pořadí. Spolehlivý přenos je docílen pomocí potvrzování jednotlivých paketů. Nejprve je nutné navázat spojení. TCP se používá pro přenos dat, kde je důležité získat celá data. Spojení podporuje současný obousměrný přenos.

2.3.2.3 TCP paket

Tabulka 2.5: TCP paket[8]

Bity	0-3	4-9	10-15	16-23	24-31
0	Zdrojový port			Cílový port	
32	Číslo sekvence				
64	Potvrzený bajt				
96	Offset dat	Rezervováno	Příznaky	Okénko	
128	Kontrolní součet			Urgent Pointer	
160	Volby (volitelné)				
192	Volby (pokračování)			Výplň	
224	Data				

2.3.2.4 UDP

Programy mezi sebou nenavazují spojení, není garantováno přijetí dat koncovým zařízením, a proto se používá například u streamování videa nebo audia.

2.3.2.5 UDP paket

Tabulka 2.6: UDP paket[8]

	bity 0-15	16-31
0	Zdrojový port	Cílový port
32	Délka	Kontrolní součet
64	Data	

2.4 Převodník USB2CAN

Naše vytvořená brána bude číst a zapisovat do převodníku USB2CAN, a tak by bylo vhodné se podívat na strukturu komunikace mezi PC a CAN bus adaptérem USB2CAN.

Základem adaptéru je obvod firmy FTDI FT245BM (USB) a controller SJA 1000 (CAN sběrnice). Pro čtení z USB slouží knihovna libftdi.

Zdroj: `http://rs.canlab.cz/?q=cs/usb2can_interface`

Obrázek 2.8: Převodník USB2CAN

2.4.1 Struktura zpráv:

Tabulka 2.7: Struktura zprávy USB2CAN[9]

Bajt:	0	1	2	3-18
Popis:	Start byte	Command	Length	Data
Hodnota:	0x0F	X	0x00 - 0x10	X

„Prvním bajtem zprávy je takzvaný START_BYTE. Jeho hodnota je definována jako 0x0F. Tento bajt je určen pro synchronizaci přenosu mezi PC a adaptérem při jejím eventuálním přerušení. Následující bajt je označen jako COMMAND. Tímto bajtem sdělujeme PC nebo adaptéru o jaká data nebo příkaz se jedná. Od významu tohoto bajtu je pak odvozen význam dat v poli DATA. Pro jednotlivé příkazy COMMAND není z důvodu optimalizace pře-

nosu mezi obvody PIC a FTDI stanovena pevná délka. Proto je uveden bajt LENGTH, který udává počet datových bajtů. Optimalizace přenosu se vyplatí při přenosu velkého množství CAN zpráv, které se mohou lišit svojí délkou. Počet datových bajtů se může pohybovat od 3, pro zprávu se standardním identifikátorem a nulovým počtem datových bajtů v CAN zprávě až po 13, pro zprávu s rozšířeným identifikátorem a 8 datovými bajty v CAN zprávě.“[9] Podporované příkazy lze najít v citovaném manuálu.

Převodník nemá neomezenou paměť pro čekající CAN rámce k odeslání, proto má převodník definované dvě hranice: Critical Transmit Limit a Ready Transmit Limit. Pokud se přesáhne hranice Critical Transmit Limit, tak převodník začne generovat error zprávy do sítě s požadavkem o zpomalení posílaní dalších dat směrem z počítače. Pokud dále klesne počet čekajících CAN rámců na úroveň Ready Transmit Limit, tak převodník oznámí, že počítač může opět zrychlit. Převodník taktéž periodicky generuje zprávy o tom, že buffer je prázdný.

Struktura datové části při R/W odpovídá struktuře použité v obvodu SJA 1000.

2.4.2 SJA 1000 Controller

- Kompatibilní s CAN 2.0B protokolem
- Podporuje 11 bitové i 29 bitové identifikátory
- Bit rate až 1 Mbit/s
- Rozhraní pro různé mikroprocesory

2.4.2.1 Struktura zprávy použitá v komunikaci s řadičem:

Tabulka 2.8: První bajt zprávy SFF a EFF v komunikaci s SJA 1000

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FF	RTR	X	X	DLC.3	DLC.2	DLC.1	DLC.0

Tabulka 2.9: Druhý a třetí bajt zprávy SFF popisující ID v komunikaci s SJA 1000

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	X	X	X	X	X

Tabulka 2.10: Čtvrtý, pátý, šestý a sedmý bajt zprávy EFF popisující ID v komunikaci s SJA 1000

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ID.4	ID.3	ID.2	ID.1	ID.0	X	X	X

„X“ znamená don’t care. Dále následuje 0-8 bajtů obsahující data.

2.5 Brány sběrnice CAN do počítačové sítě

Na branách sběrnice CAN do počítačové sítě nás především zajímá způsob komunikace s koncovým zařízením z počítačové sítě a způsob zapouzdření CAN rámců do standardních síťových protokolů.

2.5.1 ETH2CAN firmy CANLAB s. r. o.

Převodník obsahuje rozhraní Ethernet 10 Mbit/s a konektor Molex, který slouží k napájení a připojení sběrnice CAN. „Zařízení má svou IP adresu a TCP port prostřednictvím kterých probíhá veškerá komunikace. Chová se jako server, tedy klient se připojuje k tomuto zařízení.“[10] K převodníku lze připojit až 4 klienty zároveň.

Převodník lze konfigurovat prostřednictvím nadefinované struktury paketů směrem klient→převodník.

Zdroj: `http://rs.canlab.cz/?q=cs/eth2can_ethernet_can_interface`

Obrázek 2.9: Brána ETH2CAN

2.5.1.1 Struktura pro datové pakety:

Listing 2.3: Struktura paketu dat převodníku USB2CAN

```
1 typedef struct _ETH_CAN_PACKET {
2  unsigned ___int32 magic;
3  unsigned char  packet_type;
4  unsigned ___int16 packet_size;
5  unsigned char  id;
6  unsigned char  number_of_items;
7  ETH_CAN_MESSAGE  item1;
8  ETH_CAN_MESSAGE  item2;
9  ..
10 ETH_CAN_MESSAGE  itemn;
11 } ETH_CAN_PACKET;
```

```
12 typedef struct _ETH_CAN_MESSAGE {
13   unsigned __int32 id;
14   unsigned char   data[8];
15   unsigned char   dataLen;
16   unsigned char   msgFlags;
17 } ETH_CAN_MESSAGE
```

`_ETH_CAN_PACKET` slouží pro obousměrnou komunikaci server/klient. Položka `magic` slouží pro ověření endianity. Údaj `number_of_items` označuje počet CAN zpráv v paketu. Maximální počet CAN zpráv v jednom paketu je 14. Toto řešení zvyšuje propustnost ethernetového rozhraní.

`_ETH_CAN_MESSAGE` je struktura pro samotnou CAN zprávu, kde položka `msgFlags` určuje zda jde o 11 bit, nebo 29 bit identifikátor a také určuje zda jde o přenos RTR zprávy.

2.5.2 CAN-Ethernet Gateway V2 firmy SYS TEC electronic GmbH

Tato brána obsahuje rozhraní Ethernet 10/100 Mbit/s, dvě rozhraní sběrnice CAN a rozhraní USB sloužící ke konfiguraci.

Zdroj: `http://www.systec-electronic.com/en/products/industrial-communication/interfaces-and-gateways/can-ethernet-gateway-v2`

Obrázek 2.10: Brána CAN-Ethernet Gateway V2

Tabulka 2.11: Možnosti nastavení různých interface brány CAN-Ethernet Gateway V2[11]

Možné rozhraní	Typ
CAN	`CAN-Bus`
UDP-Client	`BTP_UDP_CAN_CLIENT`
UDP-Server	`BTP_UDP_CAN_SRV`
TCP-Client	`BTP_TCP_CAN_CLIENT`
TCP-Server	`BTP_TCP_CAN_SRV`
Data logger	`DLOG`

Jak můžeme vidět v tabulce, tato brána umožňuje komunikaci pomocí protokolů UDP i TCP. Brána lze nakonfigurovat s více rozhraními najednou. Např. se může chovat jako UDP-server a posílat přijímané zprávy na CAN sběrnici a naopak. Zároveň se může chovat jako TCP-Client a posílat veškeré zprávy pomocí TCP jinému serveru.

2.5.2.1 Struktura pro datové pakety:

Listing 2.4: Struktura paketu dat brány CAN-Ethernet Gateway V2

```
1 typedef struct
2 {
3  DWORD m_dwID;
4  BYTE m_bMsgType;
5  BYTE m_bLen;
6  BYTE m_bData[8];
7 } tCANMsg;
```

Položka `m_bMsgType` opět slouží k zjištění velikosti identifikátoru a zda jde o zprávu typu RTR.

2.5.3 CAN@net II/Generic firmy HMS Industrial Networks

Tato brána obsahuje Ethernet 10/100 Mbit/s a přípojku CAN sběrnice pomocí konektoru RS-232. „CAN@net II/Generic uvádí jednoduchý a flexibilní přístup s CAN systémem přes Ethernet. Díky TCP/IP protokolu může být CAN@net II/Generic přístupný přes internet“[12] Brána posílá zprávy kódované v ASCII kódu.

Zdroj: `https://www.ixxat.com/products/products-industrial/gateways-and-bridges/can@net-ii-generic`

Obrázek 2.11: Brána CAN@net II/Generic

2.5.3.1 Struktura pro datové pakety:

Tabulka 2.12: Formát zprávy brány CAN@net II/Generic[12]

M	FTD	ID	XX	XX	XX	XX	XX	XX	XX	XX	`\r\n`

Tabulka 2.13: Detail jednotlivých znaků zprávy CAN@net II/Generic[12]

M	Typ zprávy „CAN zpráva“
FTD	CAN detaily zprávy: F - Frame formát (S-standardní; E-rozšířený) T - Frame formát (D-data; R-RTR) D - Frame formát (D-Délka dat; 0 až 8)
ID	ID zprávy CAN
XX	Data CAN zprávy; 0 až 8 bajtů dlouhé
`\r\n`	Oddělovač zpráv

Zpráva „M SD7 200 1 2 3 4 5 6 7 \r\n“ bude odeslána v TCP paketu takto: „4D, 20, 53, 44, 37, 20, 32, 30, 30, 20, 31, 20, 32, 20, 33, 20, 34, 20, 35, 20, 36, 20, 37, 20, 0A, 0D“.

2.5.4 PCAN-Ethernet Gateway firmy PEAK-System Technik GmbH

Opět tato brána obsahuje Ethernet 10/100 Mbit/s a RS-232 konektor pro připojení CAN sběrnice. Umožňuje využívání protokolů TCP i UDP.

Zdroj: `http://www.peak-system.com/PCAN-Ethernet-Gateway-DR.330.0.html?&L=1`

Obrázek 2.12: Brána PCAN-Ethernet Gateway

2.5.4.1 Struktura pro datové pakety:

Tabulka 2.14: Formát zprávy brány PCAN-Ethernet Gateway

Délka	Název	Popis
2 bajty	Length	Fixní hodnota 0x24.
2 bajty	Message type	Fixní hodnota 0x80 (CAN rámec).
8 bajtů	Tag	Nepoužitý.
4 bajty 4 bajty	Timestamp Low Timestamp High	Slouží pouze jako informace.
1 bajt	Channel	Nepoužitý.
1 bajt	DLC	Délka dat CAN rámce v bajtech.
2 bajty	Flags	Nepoužitý.
4 bajty	CAN ID	Bit 0-28 ID Bit 29 fixně 0 Bit 30 RTR Bit 31 1-rozšířený ID Bit 31 0-standardní ID
8 bajtů	CAN Data	Vždy 8 bajtů.

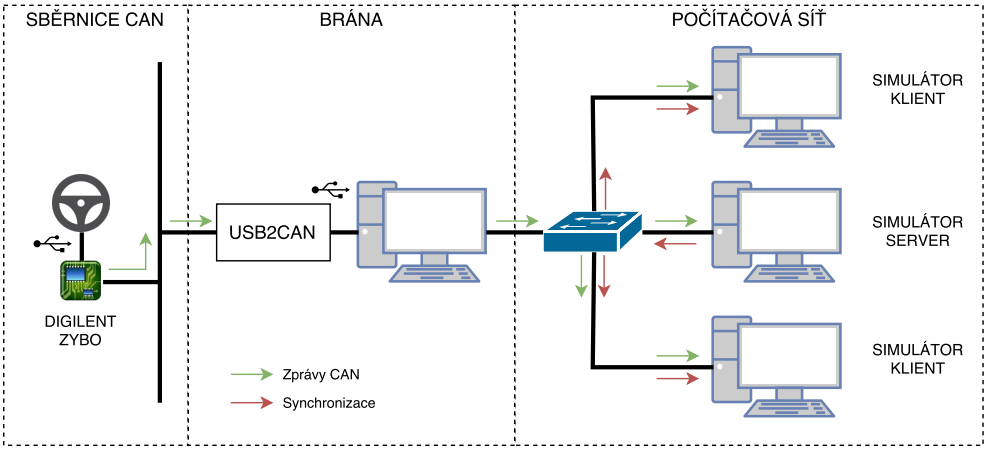
2.5.5 Závěr

Většina bran podporuje protokoly TCP i UDP. Některé z nich obalují zprávy do rozsáhlých struktur, které zatěžují síťové rozhraní. Tento přístup snižuje celkovou propustnost dat. Naopak některé se snaží co nejvíce minimalizovat tyto struktury a tak umožnují vetší propustnost dat síťového rozhraní.

Kapitola 3

Návrh

Zadáním práce bylo navrhnout a upravit vybraný simulátor tak, aby bylo možné simulované vozidlo ovládat přes sběrnici CAN pomocí mnou vytvořené brány. Další úpravou mělo být přidání možnosti běhu více instancí s různými pohledy do scény synchronizovaně. V první části se vrhnu na zmíněné návrhy pro simulátor a v druhé části se budu zabývat návrhem brány.



Obrázek 3.1: Celkový návrh spojený s přechozí bakalářskou prací

3.1 Návrh úprav pro simulátor

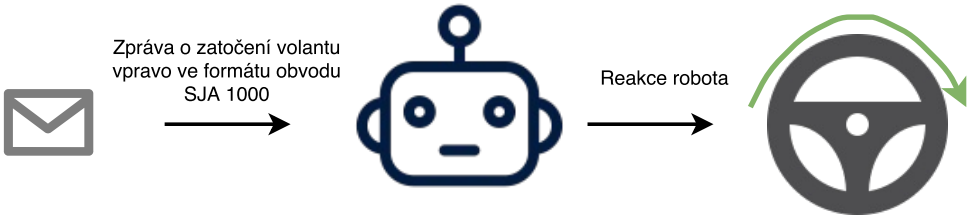
3.1.1 Ovládaní pomocí CAN sběrnice

Návrhem je vytvořit robota, který bude očekávat zprávy od brány v definovaném formátu zpráv obvodu SJA 1000. Robot bude naslouchat IDčkům definovaným v bakalářské práci Pavla Zajíce. Pro naslouchání sítě bude potřeba vytvořit druhé vlákno, které bude předávat informace o vstupech hlavnímu

vláknu.

K síťové komunikaci navrhuji použití knihovny ENet, abychom pokud možno zachovali multiplatformnost simulátoru.

Pro vytvoření vlákna a zamykání předaných dat mezi vlákny navrhuji knihovnu SDL2.



Obrázek 3.2: Předpokládaná reakce robota na příchozí zprávu

3.1.1.1 Zdůvodnění použití knihovny ENet

Síťová knihovna ENet dovoluje posílat spolehlivé i nespolehlivé pakety pomocí protokolu UDP.

„ENet nepoužívá žádné velké množství globálních proměnných. Většina stavů je zapouzdřena v ENetHost struktuře. Dokud si hlídáte přístup k této struktuře, tak ENet by měl fungovat v více vláknovém prostředí.“[13]

„V perfektním světě by to znamenalo, že znovu objevujeme TCP, ale jak mnozí zjistili, tak použití TCP místo nebo s UDP může vést k široké škále nočních můr. TCP je dobrý, solidní protokol, jenomže jednoduše není vhodný k úlohám real-time her. Implementaci diktuje mnoho pravidel, které nejsou praktické pro hry. Pokud chcete použít TCP, tak čiňte - tato knihovna je pro lidi, které nechtějí buď použít TCP nebo ho zkusili a skončili odrazeni jeho výkonem.“[13]

3.1.1.2 Zdůvodnění použití knihovny SDL2

Knihovna SDL2 umožňuje použití multiplatformních vláken a zámků.

Listing 3.1: Ukázka použití knihovny SDL2

```
1    SDL_Thread *thread;
2    int         threadReturnValue;
3
4    /* Simply create a thread */
5    thread = SDL_CreateThread(TestThread, "TestThread", (void
         *)NULL);
6
7    if (NULL == thread) {
8        printf("\nSDL_CreateThread failed: %s\n", SDL_GetError
             ());
9    } else {
10       SDL_WaitThread(thread, &threadReturnValue);
11       printf("\nThread returned value: %d",
             threadReturnValue);
12   }
```

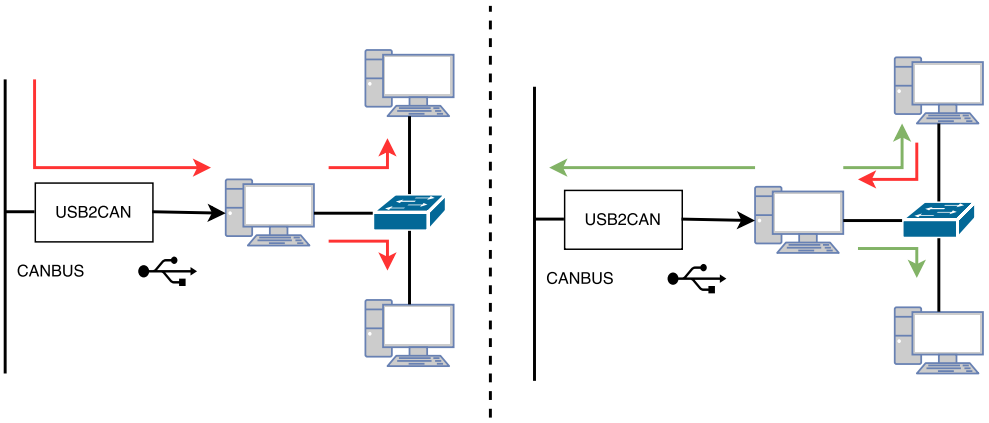
3.1.2 Řízení více instancí s různými pohledy do scény

Simulátor obsahuje nedodělaný modul pro hru více hráčů. Menu zprostředkující tento druh závodu bohužel často shazuje celý simulátor kvůli chybě způsobující „SEGMENTATION FAULT“, tudíž se z mého přesvědčení autoři rozhodli tento herní mód nevydat. Síťový modul je zakomponován do modulu k obsluze fyziky, kde provádí synchronizaci jednotlivých vozidel. Propojení instancí je formou peer-to-peer. Předpokládám možnost úpravy modulu pomocí vytvoření globální proměnné v modulu tak, aby bylo možné vytvořit spojení server-klient. Dalším předpokladem je, že půjde srozumitelně upravit grafický modul tak, že umožní různé pohledy do scény.

3.2 Návrh brány

Za předpokladu, že se podaří vytvořit robota naslouchajícího na zprávy v definovaném formátu obvodu SJA 1000 a že k síťové komunikaci použijeme knihovnu ENet, budeme taktéž tuto knihovnu k programování používat. Návrhem je vytvoření dvou vláken. Hlavní vlákno bude číst z převodníku pomocí knihovny libftdi. K použití této knihovny využiji dokumentace dostupné z `https://www.intra2net.com/en/developer/libftdi/documentation/`.

Pokud brána načte celou zprávu z převodníku do předem připraveného bufferu, tak zprávu pošle dál všem klientům v síti. Brána se bude chovat jako server, tudíž druhé vlákno naopak bude naslouchat síti a posílat zprávy všem klientům i převodníku.



Obrázek 3.3: Vlevo CAN zařízení posílá zprávu a vpravo PC

Kapitola 4

Realizace

Tato kapitola se zabývá samotnou realizací návrhů. Pomocí níže uvedeného softwaru a hardwaru jsem upravil simulátor a poté vytvořil bránu sběrnice CAN do počítačové sítě.

4.1 Použitý software a hardware

Jako samotný simulátor jsem použil Speed Dreams verze 2.1, který můžete nalézt na přiloženém CD. K editaci software jsem použil textový editor Xed. K vytvoření práce jsem pak využil sázecího systému LaTeXa k vytváření obrázků jsem použil nástroje z internetové stránky `https://www.draw.io/`. Pro testování jsem použil realizaci práce Pavla Zajíce, obsahující vývojovou desku Digilent ZYBO s moduly pro CAN komunikaci a potřebným softwarem. Dále obsahující herní volant, který je do vývojové desky připojen pomocí USB konektoru. Jako počítač jsem používal osobní laptop s procesorem Intel i5-6300HQ, grafickou kartou Nvidia GeForce GTX1060 6 GB a 8 GB RAM.

Popis použitého softwaru a hardwaru není předmětem práce, tudíž se jimi nebudu zabývat.

4.2 Úpravy pro simulátor

Velkou částí simulátoru jsou konfigurační XML soubory. Některé z nich se kopírují do domovské složky uživatele, tím umožňují různé nastavení pro různé uživatele na stejném počítači. Veškeré konfigurační soubory lze nalézt v datové složce simulátoru. Změny v XML souborech jsou popsané na přiloženém CD.

4.2.1 Realizace robota naslouchajícího CAN sběrnici

Modul robota se nazývá `CAN_OVER_IP`. K realizaci potřebujeme nejprve vytvořit do složky `/src/drivers/CAN_OVER_IP` zdrojový soubor `CAN_OVER_IP.cpp`, který slouží jako vstupní bod modulu. Tento soubor musí obsahovat definici

předem určených funkcí. Simulátor umožňuje spuštění závodu s více instancemi stejného robota.

Listing 4.1: Důležité funkce modulu robota

```
1 // Module entry point (new fixed name scheme).
2 // Extended for use with schismatic robots
3 extern "C" int moduleWelcome(const tModWelcomeIn* welcomeIn,
4                               tModWelcomeOut* welcomeOut);
5
6 //Module entry point
7 extern "C" int CAN_OVER_IP
8 (tModInfo *modInfo);
9
10 //Module exit
11 extern "C" int moduleTerminate();
12
13 static void
14 drive(int index, tCarElt* car, tSituation *s);
```

- `moduleWelcome`
  - Načítá instance robota definované v jehož konfiguračním souboru
  - Alokuje a vytvoří instanci třídy `NetworkInput`
- `CAN_OVER_IP`
  - Nastavuje informace o modulu a informuje o existenci instance
- `moduleTerminate`
  - Dealokuje prostředky modulu včetně instanci třídy `NetworkInput`
- `drive`
  - Přebírá informace o ovládání z příslušné instance `NetworkInput`

4.2.1.1 Soubor networkinput.cpp

Tento soubor definuje v hlavičce souboru ID vstupů herního volantu, strukturu `SCarSpecs` a třídu `NetworkInput`. Struktura `SCarSpecs` slouží k předání informací o ovládání a třída `NetworkInput` vytváří vlákno, které se připojí k bráně jako klient. Toto vlákno pak slouží poslechu příchozích zpráv, jejich filtraci a nakonec nastavení proměnné odpovídající parametru vozidla.

4.2.1.2 Konfigurační soubory

Pro robota jsou důležité tyto dva soubory `CAN_OVER_IP.xml` a `network.xml`. První ze zmíněných souborů definuje jednotlivé instance robota. Druhý pak definuje IP adresu a port brány. IP adresu a port lze nastavit prostřednictvím úvodního menu simulátoru.

Listing 4.2: Definice instance robota

```
<section name="1">
<attstr name="name" val="CAN_OVER_IP - sc-boxer-96"/>
<attstr name="short name" val="CAN_OVER_IP"/>
<attstr name="code name" val="COIP"/>
<attstr name="desc" val=""/>
<attstr name="team" val="CVUT FIT"/>
<attstr name="author" val="Tomas Cermak"/>
<attstr name="car name" val="sc-boxer-96"/>
<attnum name="race number" val="1"/>
<attnum name="red" val="0.9"/>
<attnum name="green" val="0.9"/>
<attnum name="blue" val="0.9"/>
<attstr name="features" val="wet track"/>
</section>
```

Listing 4.3: Nastavení adresy brány

```
<params type="param" name="network">
<section name="network">
<attstr name="IP address" val="127.0.0.1"/>
<attnum name="port" val="50001"/>
</section>
</params>
```

4.2.2 Realizace synchronizace více instancí s různými pohledy do scény

Druh závodu jsem pojmenoval instancesync. Tento mód je přístupný v menu závodů. Do modulu `networking` jsem si přidal globální proměnnou jako příznak, který značí, zda jde o normální hru po síti, nebo jde o mód instancesync. Mód instancesync filtruje nepotřebné volání části kódu uvnitř modulu. Odstraněním připojení klienta ke klientům vzniká z propojení peer-to-peer server-klient. Navíc je potřeba odstranit časovač, který každých 40 ms vyvolává čtení ze sítě a nahradit ho vláknem, které bude čekat na zprávy ze sítě. Důvodem je zbytečné opoždění příjmání synchronizačních paketů a reakce simulátoru na ně. Příznak také nastavuje kameru do scény včetně HUD a otočení kamery ve scéně. Nutností je vytvoření robota `NOTMOVINGROBOT`, který na rozdíl od robota `CAN_OVER_IP` nemá definovaný žádný vstup. Tento robot slouží pro klientské instance simulátoru jako obdoba robotů servrové instance, které jsou poté spojeny synchronizací. Komunikace probíhá na portu č. 28500.

4.2.2.1 Menu pro instancesync mód

Konfigurační soubory jednotlivých menu definují jejich vzhled a jejich dílčí komponenty. Menu pro závody taktéž nastavuje závody, proto menu instancesync definovanou v souboru `instancesyncmenu.cpp` lze rozdělit na klientskou a serverovou část. Serverová část načítá a upravuje konfigurační soubor závodu

`instancesyncrace.xml` a vytváří kopii obsahující místo jednotlivých robotů připojených k závodu roboty `NOTMOVINGROBOT`. Server také vytvoří soubor s definicí instancí robota `NOTMOVINGROBOT` a rozešle tyto dva soubory připojeným klientům. Klienti si oba soubory načtou a čekají na změnu nebo start závodu.

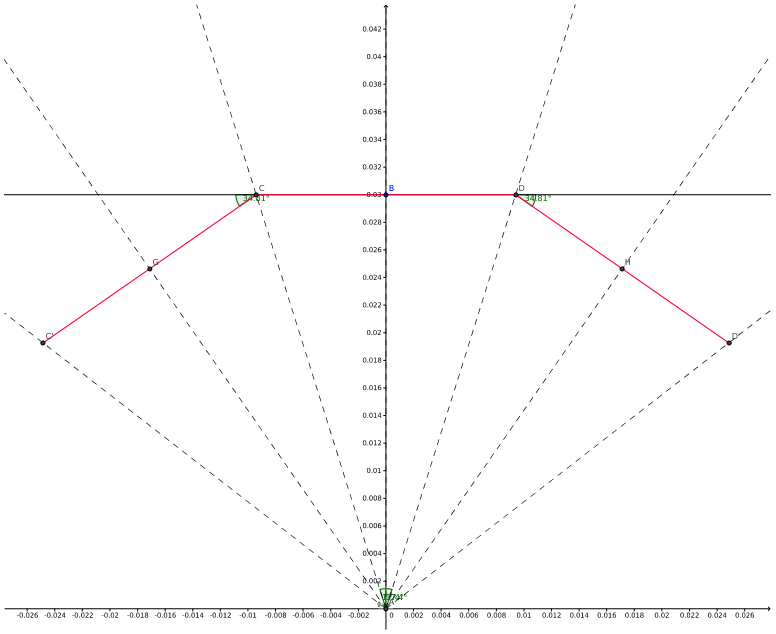
4.2.2.2 Nastavení kamery

Pro vytvoření kamery bylo potřeba upravit modul `ssggraph`. Simulátor bohužel neobsahuje žádnou třídu kamery, která by vyhovovala zadání. Proto jsem vytvořil třídu `cGrCarCamBachelorView`, umožňující otočení kamery. Otočení kamery je řešeno pomocí matice transformace, kterou dokáže vytvořit sama knihovna Plib.

Nastavení kamery pro jednotlivou instanci probíhá v menu lobby závodu. Simulátor očekává natočené monitory pod úhlem 35°.

Listing 4.4: Vytvoření kamery otočené doprava

```
1 /* cam F12 = right view */
2 cam = new cGrCarCamBachelorView(myscreen,
3   id,
4   1, /* drawCurr */
5   1, /* drawBG  */
6   20, /* fovy */
7   20, /* fovymin */
8   20, /* fovymax */
9   0.03, /* near */
10  -1, /* xView */
11  0.0, /* offsetX */
12  0.0, /* offsetY */
13  0.0, /* offsetZ */
14  fixedFar ? fixedFar : 600.0 * fovFactor, /* far */
15  fixedFar ? fixedFar/2 : 300.0 * fovFactor, /* fogstart */
16  fixedFar ? fixedFar : 600.0 * fovFactor /* fogend */
17 );
```



Obrázek 4.1: Nastavení pohledů jednotlivých instancí

4.3 Implementace brány

Bránu tvoří tři zdrojové soubory. Soubory `UBS2CAN.c` obsahuje funkce zahrnující veškerou práci s převodníkem. `prints.c` zprostředkovává debugovací výpisy a `main.c` zaštiťuje veškerou funkci brány. V posledním zmíněném souboru běží dvě smyčky. První z nich běží na hlavním vlákně a má na starosti zprávy z převodníku. Druhá běží na POSIX vlákně obsluhující zprávy přicházející ze sítě a odesílaní zpráv pomocí knihovny ENet. Brána používá port č. 50001 definovaným na začátku souboru `main.c`.

Kapitola 5

Testování

5.1 Robot

Robot byl testován pomocí vývojové desky Digilent ZYBO s nainstalovým ovladačem SocketCAN. Pomocí nástroje `cansend` jsem byl schopen generovat zprávy volantu. Robot reagoval perfektně na vstupy z CAN sběrnice.

5.2 Synchronizace

Testování synchronizace probíhalo na jednom počítači v rámci loopbacku. V průběhu testovaní byly upraveny některé hodnoty zejména perioda synchronizace vozidel po síti, protože docházelo k trhání obrazu kvůli rozjetí jednotlivých instancí.

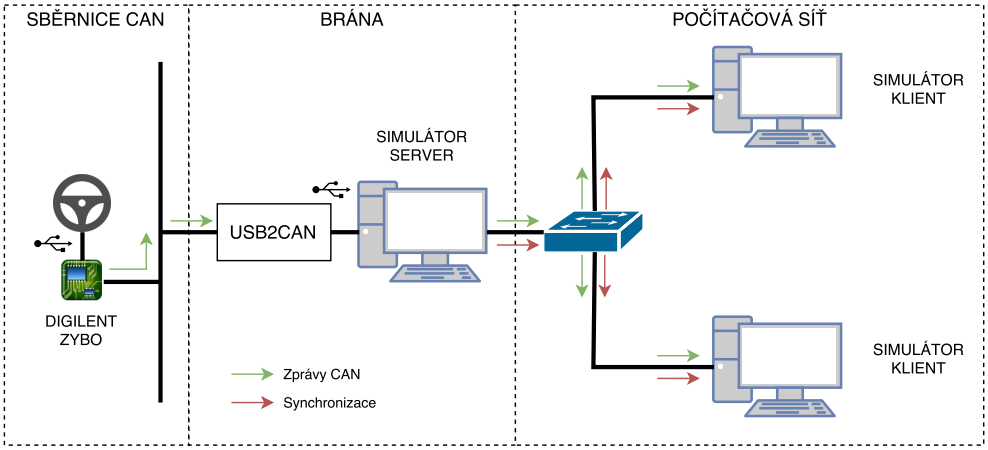
Obrázek 5.1: Výsledný vzhled synchronizace

5.3 Brána

Otestováním propojení bakalářské práce Pavla Zajíce, brány a robota v simulátoru jsme zjistili, že knihovna ENet není úplně thread safe a tak docházelo u odesílání k „SEGMENTATION FAULT“. Tento problém se odstranil přenecháním veškeré síťové komunikaci příslušnému vláknu.

5.4 Celkový test

K testu brány jsem přidal další dva počítače se simulátory a tím jsem mohl otestovat práci jako celek. Tento test nenarazil na žádný problém.



Obrázek 5.2: Testovací zapojení

Závěr

Náplní práce bylo získat povědomost o existujících simulátorech. Rovněž pak o branách mezi sběrnicí CAN a počítačovou sítí. Navrhnout a vytvořit vlastní bránu. Nakonec provést změny v simulátoru, pro možnost ovládaní ze sběrnice CAN a řízení více instancí simulátoru najednou.

V práci se mi úspěšně podařilo naplnit veškeré požadavky a vytvořit zmíněnou bránu a upravit simulátor tak, jak bylo předmětem zadání. Výsledné aplikace jsem otestoval a tím jsem ověřil jejich funkčnost.

V budoucnu by bylo možné práci rozšířit tak, že robot bude odesílat informace o vozidle a tak spojit všechny tři bakalářské práce zabývající se spojením jednotlivých komponent sběrnicí CAN.

Literatura

[1] About TORCS. [online], [cit. 2017-05-09]. Dostupné z: http://torcs.sourceforge.net/index.php?name=Sections&op=viewarticle&artid=1

[2] The Official TORCS FAQ (frequently asked questions). [online], [cit. 2017-05-09]. Dostupné z: http://torcs.sourceforge.net/?name=Sections&op=viewarticle&artid=30

[3] About Speed Dreams. [online], [cit. 2017-05-09]. Dostupné z: http://www.speed-dreams.org/release-20-en.html

[4] Hardware / Software requirements. [online], [cit. 2017-05-09]. Dostupné z: https://sourceforge.net/p/speed-dreams/wiki/HardSoftRequirements/

[5] About the project. [online], [cit. 2017-05-09]. Dostupné z: http://wiki.vdrift.net/index.php?title=About_the_project

[6] Requirements. [online], [cit. 2017-05-09]. Dostupné z: http://wiki.vdrift.net/index.php?title=Requirements

[7] CAN - Controller Area Network. [online], [cit. 2017-05-09]. Dostupné z: http://rs.canlab.cz/?q=cs/can_bus

[8] CCNA, M. M.: Exploring the anatomy of a data packet. *techrepublic.com [online]*, červenec 2001, [cit. 2017-03-26]. Dostupné z: http://www.techrepublic.com/article/exploring-the-anatomy-of-a-data-packet/

[9] CANLAB s. r. o.: *Popis USB komunikace mezi PC a CAN bus adaptérem USB2CAN*. [cit. 2017-05-09]. Dostupné z: http://rs.canlab.cz/sites/default/files/StrukturaUSBkomunikace1-3.pdf

[10] CANLAB s. r. o.: *ETH2CAN firmware*. [cit. 2017-05-09]. Dostupné z: http://rs.canlab.cz/sites/default/files/ETH2CAN_CAN_FW.pdf

[11] SYS TEC electronic GmbH: *CAN-Ethernet Gateway V2 System Manual*. [cit. 2017-05-09]. Dostupné z: http://www.systec-electronic.com/uploads/65/3b/653b129d548de2f98953d8d9e475eab8/L-1294e_10_CAN-Ethernet-Gateway-V2-System-Manual.pdf

[12] HMS Industrial Networks: *CAN@net II/Generic Manual*. [cit. 2017-05-09]. Dostupné z: https://www.ixxat.com/docs/librariesprovider8/default-document-library/products/repeater/can-at-net-manual-generic-english.pdf?sfvrsn=2

[13] Frequently Answered Questions. [online], [cit. 2017-05-09]. Dostupné z: http://enet.bespin.org/FAQ.html

Příloha **A**

Seznam použitých zkratek

AI Umělá inteligence

API Application programming interface

ASCII American Standard Code for Information Interchange

CAN Controller Area Network

CD Compact disc

CPU Central processing unit

CVUT České vysoké učení technické

DLC Data Length Code

DNS Domain Name System

EFF Extended Frame Format

FF Format Frame

FIT Fakulta informačních technologií

FTP File Transfer Protocol

HUD Head-up display

HTTP HyperText Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ID Identifier

IMAP Internet Message Access Protocol 4

IP Internet Protocol

MAC Media Access Control

OS Operating system

PC Personal Computer

POP3 Post Office Protocol version 3

POSIX Portable Operating System Interface

RAM Random-access memory

RTR Remote transmission request

R/W Read/Write

SFF Standard Frame Format

SMTP Simple Mail Transfer Protocol

SNMP Simple Network Management Protocol

SSH Secure Shell

TCP Transmission Control Protocol

TORCS The Open Racing Car Simulator

UDP User Datagram Protocol

USB Universal Serial Bus

XML Extensible Markup Language

ZYBO ZYnq BOard

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace brány
- `src`
  - `gwimpl` ..... adresář obsahující zdrojové kódy implementace gatewaye
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `sim` ..... adresář obsahující simulátor a patch
    - `speed-dreams-src-base-2.2.1-r6404.tar.xz` ..... simulátor
    - `patch.txt` ..... patch pro simulátor
    - `install.sh` ..... instalační script
  - `install.txt` ..... instrukce k instalaci
  - `XMLchanges.txt` ..... popis změn XML souborů simulátoru
- `text`
  - `thesis.pdf` ..... text práce ve formátu PDF