VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

GENEROVÁNÍ KÓDU PRO MIKROKONTROLÉRY POMOCÍ AUTOMATICKÝCH NÁSTROJŮ

RAPID PROTOTYPING VIA AUTOMATIC SOFTWARE CODE GENERATION FOR EMBEDDED APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — DAVID KLIMEŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOSEF VEJLUPEK
SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky, mechatroniky a biomechaniky
Akademický rok: 2010/2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): David Klimeš

který/která studuje v **bakalářském studijním programu**

obor: **Mechatronika (3906R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Generování kódu pro mikrokontroléry pomocí automatických nástrojů

v anglickém jazyce:

Rapid prototyping via automatic software code generation for embedded applications

Stručná charakteristika problematiky úkolu:

Generování kódu pro mikrokontrolery dsPIC v prostředí MATLAB-Simulink, generování kódu C pomocí LabView.
Real-Time řízení. Implementace algoritmů do mikrokontroléru. Návrh a výroba elektroniky nezbytné pro funkci řídící jednotky spolu s řízením výukovým modelem.

Cíle bakalářské práce:

1/Seznámení se s možností automatického generování kódu v prostředí Simulink, LabView,... vypracování stručného přehledu dostupných nástrojů zaměřeného především na mikrokontrolery
2/Seznámení se s kódem pro platformu Car4
3/Úprava kódu pro Car4 – vylepšení komunikace (dálkové ovládání, komunikace mezi jednotkami)
4/Připravit dálkové ovládání postavené na dsPIC

Seznam odborné literatury:

www.ni.com
www.mathworks.com
www.mechlab.cz

Vedoucí bakalářské práce: Ing. Josef Vejlupek

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2010/2011.

V Brně, dne 18.11.2010

L.S.

prof. Ing. Jindřich Petruška, CSc.	prof. RNDr. Miroslav Doupovec, CSc.
Ředitel ústavu	Děkan fakulty

Abstrakt

Tato práce se zabývá problematikou automatického generování kódu pro mikrokontrolery. Následně jsou využity poznatky z této oblasti pro úpravu části kódu experimentálního vozidla Car4.

Hlavní téma této práce je návrh dálkového ovládání postavené na platformě dsPIC sloužícího pro řízení vozidla Car4. Na takto vytvořeném dálkovém ovládání jsou poté prakticky vyzkoušeny možnosti nástrojů pro automatické generování kódu při vytváření firmware pro toto dálkové ovládání.

Abstract

This work deals with issues of automatic code generation for microcontrollers. Knowledge of this area is subsequently used for adjustment of the part of code of the experimental vehicle Car4.

The main theme of this work is to design a control platform of the based platform dsPIC, which serve for driving Car4. Tooling capabilities for automatic code generation in creating firmware for this remote control are then tested in practice on this remote control.

Čestné prohlášení

Čestně prohlašuji, že jsem tuto práci vypracoval samostatně s použitím uvedené literatury a pod vedením vedoucího BP.

David Klimeš, Brno 2011

Poděkování

Chtěl bych tímto poděkovat všem, kteří mi poskytovali dobré rady při dosahování cílů této práce, zejména kolektivu z Mechlabu. Veliké poděkování pak patří především vedoucímu práce, kterým byl Ing. Josef Vejlupek, za jeho cenné rady, připomínky a za celé metodické vedení této bakalářské práce.

Obsah

1 Úvod

Mikrokontrolery jsou v dnešní době čím dál tím více rozšířené součásti obsažené ve větších celcích. Jejich použití je vhodné jednak z důvodu jednoduchosti implementace navrženého systému do programové paměti mikrokontroleru, ale především díky nízké ceně mikrokontrolerů, která pak znamená snížení ceny celého návrhu. Zejména v mechatronických aplikacích je mikrokontroler často použit. Mechatronik jako návrhář inteligentních systémů musí mít široký přehled znalostí, které mu umožní vytvořit dobrý návrh sytému. Je proto obtížné v každém oboru dosahovat špičkových znalostí.

Při návrhu každého systému musí jeho návrhář dokonale ovládat především znalosti pro navržení požadovaného sytému (např. z oblasti řízení, atd...), který pak při dobrém navržení bude mít žádané vlastnosti. Výhodným nástrojem jsou pak nástroje pro automatické generování kódu pro mikrokontrolery, kdy tyto nástroje slouží jako spojovací most, mezi návrhem systému a implementací do cílového zařízení. Zjednodušení ve formě automatického vygenerování kódu pro mikrokontroler pak umožňuje zaměřit síly především na návrh samotného algoritmu řízení. Zejména kvalitní a velmi rozšířené prostředí Matlab-Simulink, které slouží pro návrh a simulaci nejrůznějších aplikací, je pak vhodné pro převod z návrhu v tomto prostředí do formy, která je již vhodná pro mikrokontroler. Samotná implementace algoritmů řízení pak nevyžaduje podrobnou znalost programovacího jazyka a problematiky týkající se oblasti mikrokontrolerů.

Cílem této bakalářské práce je především představit základní možnosti automatického generování kódu pro cílové zařízení-mikrokontroler. Možným nástrojem pro návrh systému s následným generováním kódu je pak Matlab-Simulink, případně LabVIEW. Hlavním cílem této práce je pak již zmíněný Matlab-Simulink, který je vhodný a jednoduše použitelný nástroj především pro návrh a simulaci nejrůznějších mechatronických aplikací. Doplněním tohoto prostředí o vhodný toolbox, který obsahuje „propojení“ prostředí Simulinku a cílového zařízení pak vznikne ucelené prostředí pro návrh systému a výsledné vytvoření požadovaného kódu. Pod pojmem cílové zařízení si lze představit nejrůznější zařízení, v našem případě mikrokontrolery PIC.

Tato práce je rozdělena na dvě části: teoretickou a praktickou. Teoretická rešeršní část je zaměřená na oblast automatického generování kódu a popis průběhu generování kódu pro mikrokontroler (především v prostředí Simulinku).

V druhé části této práce je prezentováno využití popsaných metod tvorby kódu pro mikrokontroler na praktické aplikaci. Hlavní cíl praktické části, této bakalářské práce je návrh dálkového ovládání pro experimentální vozidlo Car4. Požadavkem je, aby dálkové ovládání pak bylo postavené na mikrokontroleru PIC. Při programování takto použitého mikrokontroleru jsou využity nástroje pro automatické generování kódu-prostředí Simulinku.

2 Mikrokontrolery PIC

2.1. Úvod k mikrokontrolérum PIC

Mikrokontrolery PIC jsou jednočipové počítače od firmy Microchip Technology (USA). U těchto mikrokontrolerů je navzájem oddělená paměť pro program a pro data.

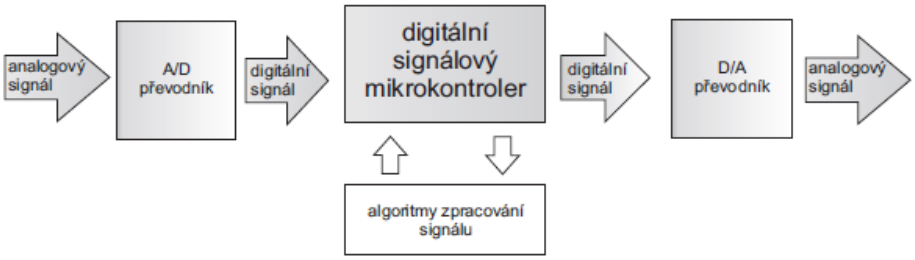
Dále budou v této práci zmiňovány především digitální signálové mikrokontrolery PIC (dsPIC). Digitální signálové mikrokontrolery jsou navrženy především pro zpracovávání digitálního signálu. Předností těchto digitálních signálových mikrokontrolerů je schopnost průběžně zpracovávat velký toku dat mikrokontrolerem.

Společnou vlastností digitálních signálových mikrokontrolerů je tedy to, že mají architekturu optimalizovanou pro matematické operace, které se uplatňují ve zpracování signálů.

Ve srovnání s procesory pro všeobecné použití mají digitální signálové mikrokontroléry menší spotřebu a vyšší výkon v úlohách zpracování signálů. Většina instrukcí trvá jeden instrukční cyklus a digitální signálový mikrokontroler má garantovanou dobu, během níž se dostane do smyčky aktivního přerušení (např. čtyři instrukční cykly), což je přínosem pro systémy Real-Time[1]. [1]

2.1.1. Zpracovávání analogového signálu v digitálních signálových mikrokontrolerech

Zpracovávání signálu nejprve začíná u A/D převodníku, kde je analogový signál (napětí) převeden na digitální a následně je zpracováván digitálním signálovým mikrokontrolerem. Po zpracování signálu mikrokontrolerem je opět signál převeden pomocí D/A převodníku na analogový signál. [2]



Obr. 2.1. Schéma zpracovávání signálu pomocí DSP

[1] *„Systém Real-Time je systém, který musí reagovat na externě vytvořený vstupní podnět v rámci specifikované časové periody“.* [10]

2.1.2. Rozdělení digitálních signálových mikrokontrolérů

Základním dělením digitálních signálových procesorů je dělení podle použité aritmetiky. Existují DSP pracující:

- v celočíselné aritmetice (integer[2])
- v aritmetice s pevnou řádovou čárkou (fixed point[3])
- v aritmetice s plovoucí řádovou čárkou (floating point[4])

U procesorů s celočíselnou aritmetikou je náročnější vývoj algoritmů, kdy je nutné převádět reálná čísla na celá a mezivýsledky výpočtů se musí neustále upravovat tzv. normalizacemi. Proto je vývoj algoritmů v těchto typech procesorů výrazně náročnější.

Hodí se proto zejména pro masovou produkci výrobků, kde nevadí poněkud vyšší cena vývoje, ale důležitá je zejména cena samotné součástky.

Procesory s plovoucí řádovou čárkou jsou opakem procesorů pracujících s celočíselnou aritmetikou, jsou sice složitější a dražší, ale vývoj programu je pro tento typ procesorů výrazně jednodušší.[2]

[2] integer-celočíselný datový typ

[3] fixed point-datový typ s pevnou desetinou čárkou

[4] floating point-datový typ s plovoucí desetinou čárkou

3 Generování kódu C v MATLAB-Simulink

3.1. Úvod ke generování kódu v prostředí Simulink

Matlab-Simulink je velice silný nástroj při návrhu nejrůznějších systémů, jejich simulaci a v neposlední řadě také pro tvorbu potřebného kódu pro cílové zařízení. Tento vygenerovaný kód umožní přenést vytvořený model v prostředí Simulink, do cílového zařízení, mikrokontrolerů.

Matlab-Simulink pomocí přidaného toolboxu pro podporu cílové skupiny mikroprocesorů (PIC) vygeneruje požadovaný kód přímo pro konkrétní, cílový procesor. V případě této práce se jedná o podporu zejména mikrokontrolerů PIC.

Tato kapitola se pak zabývá především Kerhuel Toolboxem [7], který je dále využit na praktických příkladech v kapitole 5 a kapitole 6.

Stručný popis průběhu generování zdrojového kódu:

1. Vytvoření modelu s požadovanými vlastnostmi v prostředí Simulink.
2. Model je následně doplněn o potřebné bloky z Toolboxu, který slouží pro podporu cílového zařízení (v tomto případě jsou to mikrokontrolery PIC).
3. Simulink pomocí nástroje Real-Time Workshop vygeneruje soubor v jazyce C, který odpovídá modelu vytvořeném v Simulinku.
4. Použitý Toolbox (Kerhuel Toolbox, Microchip Toolbox) takto vytvořený kód „doplní“ o potřebnou část kódu pro nastavení mikrokontroleru jako je nastavení periferii a vstupních/výstupních portů mikrokontroleru.
5. Příslušný Compiler (u mikrokontrolerů PIC je to compiler HI-TECH C® od Microchipu) pak vytvořený kód v jazyce C přeloží do zdrojového kódu (soubor .hex).
6. Vytvořený zdrojový kód je pak již připraven pro přímé nahrání do programové paměti cílového mikrokontroleru PIC. Nahrání zdrojového kódu do programové paměti se provede pomocí MPLAB IDE

3.1.1. Matlab-Simulink

Matlab-Simulink[5] je prostředí od společnosti Mathworks[6] určené k provádění výpočtů, návrhu algoritmů, modelování a simulování dynamických systémů s možností prezentace výstupních dat.

Simulink je nadstavba Matlabu, která je určena k modelování a simulování dynamických systémů.

Nadstavbová část Matlabu, Simulink, obsahuje knihovny tematicky rozdělených grafických bloků, které představují různé matematické operace, logické funkce a jiné nejrůznější funkce. Z těchto funkčních bločků je možné jednoduše vytvářet modely dynamických soustav ve formě blokových schémat. S pomocí Simulinku a jeho grafického editoru je možné vytvářet lineární, nelineární, v čase diskrétní a spojité systémy.

[5] použitá verze Matlabu je R2009a

3.2. Real – Time Workshop

Real-Time Workshop je nástroj v prostředí Matlab-Simulink, který slouží pro generování kódu v jazyce C přímo ze Simulinkovského modelu. Generování kódu pomocí nástroje Real-Time Workshop je navrženo pro generování kódu pomocí „stisku tlačítka“. Použití je tedy možné u různých real-time systémů, nebo „můžeme spustit již vytvořený model v Simulinku“ na cílovém mikroprocesoru.

Generovaný kód je plně okomentovaný kód v jazyce C, vygenerovaný z jakéhokoliv Simulinkovského modelu, zahrnující lineární, nelineární, analogové, diskrétní, nebo hybridní modely.

Real-Time Workshop poskytuje real-time vývojové funkce jako je:

- rychlá a přímá cesta od návrhu systému po implementaci do cílového zařízení
- integrace s Matlabem a Simulinkem
- plně konfigurovatelný generátor kódu-každé hledisko generovaného kódu může být nastaveno užitím knihoven Target Language Compiler

3.2.1. Aplikace pro Real-Time Workshop

Real-Time Workshop je navržen pro různé aplikace. Jedním z mála příkladů použití je:

- **Real-Time control** – umožňuje návrh a ovládání systému za použití MATLABu a Simulinku generováním kódu z blokového schématu modelu. Umožní zkompilovat a následně nahrát model přímo do cílového zařízení.
- **Real-Time signal processing** – je možné navrhnout algoritmus zpracovávání signálů pomocí MATLABu a Simulinku. Vygenerovaný kód z navrženého blokového schématu je možné zkompilovat a nahrát do paměti cílového zařízení (např. mikroprocesor).
- **Generování univerzálního kódu v jazyce C**-pro export do jiných simulačních programů [3]

3.2.2. Target Language Compiler (TLC)

Target Language Compiler je nástroj, který je zahrnut v Real-Time Workshopu. TLC slouží jako překladač z tzv. mezistupně vytvořeného Real-Time Workshopem (souborem .rtw) a kódem v jazyce C. Target Language Compiler funguje jako „textový procesor“.

3.2.2.1. Target Files (soubory .tlc)

V kontextu s Real-Time Workshopem existují dva typy Target Files: *„System target files“* a *„Block target files“*.

„System target files“ určují strukturu generovaného kódu. Modifikací těchto souborů je možné přizpůsobovat generovaný kód.

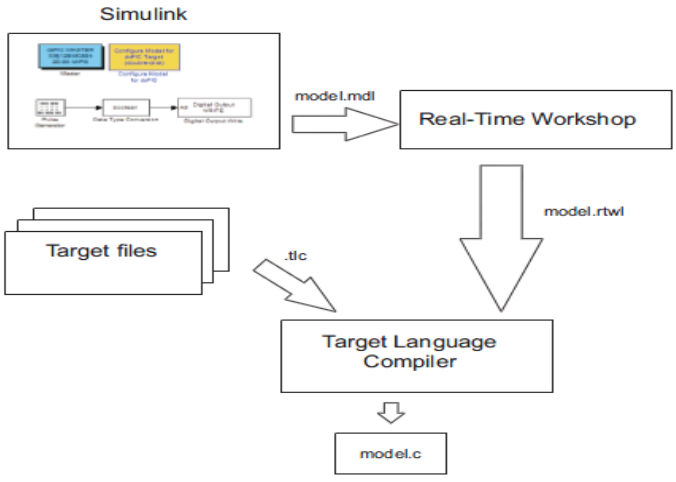
Každý blok v modelu Simulinku má svůj individuální *„Block target files“*, který popisuje jaká část kódu se má pro daný blok vygenerovat. Pro příklad, *„Block target files“* gain.tlc, generuje odpovídající kód pro blok *Gain.*

3.3. Průběh generování kódu v jazyce ANSI C s použitím nástroje Real-Time Workshop

Generování kódu začíná vytvořením modelu (model.mdl) v prostředí Simulink. Real-Time Workshop poté vytvoří z tohoto modelu soubor s příponou rtw (model.rtw). Takto vytvořený soubor reprezentuje dříve vytvořený blokový diagram v Simulinku (je možné soubor rtw vygenerovat v Matlabu pomocí příkazu *„rtwgen_model.mdl“*).

Soubor s příponou rtw tedy obsahuje informace, jako jsou hodnoty konstant, nastavení řešiče, informace o datových typech v modelu,velikost použitých vektorů, vzorkovací čas (sample time), čas potřebný na vykonávání jednotlivých bloků v modelu a spoustu dalších informací potřebných k dalšímu průběhu generování kódu z modelu model.mdl. Tyto informace jsou uloženy ve formátu, který je „jazykově nezávislý“. Jedná se o tzv. ASCII soubor, který obsahuje pouze znaky, číslice a jiné znaky (tento soubor je tedy možné otevřít i v běžném poznámkovém bloku).

Po vytvoření souboru model.rtw, pokračuje ve tvorbě požadovaného kódu Target Language Compiler, který transformuje soubor model.rtw do cílově-specifického kódu. Target Language Compiler začíná načtením soboru model.rtw, který následně zkompiluje. Kompilace probíhá tak, že se nejprve specifikují požadavky na cílový kód pomocí *target files,* což jsou soubory, které určují jak se má soubor model.rtw transformovat do cílově-specifického kódu. Target Language Compiler tedy z kombinace souborů model.rtw a Target files vytvoří plně okomentovaný kód v jazyce C, který reprezentuje model.mdl v prostředí Simulink. [4]



Obr. 3.1. Průběh generování kódu ANSI C

3.4. Kerhuel Toolbox

Kerhuel Toolbox[6] [7] je rozšiřující toolbox do prostředí Simulink, který je vytvořen pro propojení blokového schématu v Simulinku a cílového procesoru (jeho portů, periferii, atd.).

V tomto toolboxu jsou vytvořené bloky pro podporu mikrokontrolerů PIC. Tyto bločky umožní ovládat a propojovat různé periferie podporovaných procesorů (PWM, ADC, UART, atd...).

Mezi možné cíle, pro generování kódu patří skupiny mikrokontrolerů vyráběných firmou Micochip Technology jako je PIC24, dsPIC30, dsPIC33 a některé mikrokontrolery PIC32.

3.4.1. Blok *„Master“*

Každý model vytvořený v prostředí Simulinku určený pro vytvoření kódu pro mikrokontroler musí obsahovat blok *„Master“*. Jedná se o nejdůležitější blok v celém modelu. Tento blok musí být vložen v každém modelu, kde pro správnou funkčnost ostatních bloků vložených z Kerhuel Toolboxu je vhodné jej do modelu vložit jako první blok v pořadí.

V nastavení tohoto bloku se především nastaví cílový mikrokontroler (z řady PIC24, dsPIC30, dsPIC32, dsPIC33). Další z řady nastavení v bloku *„Master“* je nastavení použitého oscilátoru, nastavení časové reference (Timer1, ADC, Free Run), nastavení funkce *PLL* (phase-locked loop), která umožní přímo zadat požadovaný počet instrukcí provedených mikrokontrolerem za jednu sekundu. Další z mnoha nastavení bloku *„Master“* je nastavení funkce *„Busy Flag Port“* (použité v kap. 5).

3.4.2. Knihovna *„Digital I/O“*

Jedná se o knihovnu obsahující bloky pro ovládání vstupních/výstupních digitálních pinů mikrokontroleru PIC.

„Digital Output WRITE“

Pomocí tohoto bloku je možné zapisovat data typu boolean na digitální výstup mikrokontroleru. Nastavení bloku *„Digital Output WRITE“* spočívá ve vybrání požadovaného portu a poté i konkrétního pinu (pinů-výběr se zadává pomocí vektorové formy).

„Digital Input“

Tento blok slouží ke čtení logické hodnoty na vybraném digitálním vstupním pinu mikrokontroleru. Nastavení bloku je stejné jako u předchozího bloku *„Digital Output WRITE“*.

„Digital Output Read“

Blok *„Digital Output Read“* slouží ke čtení aktuálně poslední logické hodnoty na digitálně výstupním pinu mikrokontroleru.

[6] použitá verze Kerhuel Toolboxu je *R3.4* (s datem vydání 30.12.2010)

3.4.3. Knihovna *„Serial PORT“*

Použitím bloků v této knihovně lze rychle a snadno vytvořit komunikaci mezi jednotlivými zařízeními. V této knihovně jsou uloženy bloky pro ovládání a nastavování sériové linky (UARTu) mikrokontroleru.

Jedná se především o bloky pro vysílání („*Tx Output“*) a příjem („*Rx Intput“*) dat po sériové lince-UART[7].

Další bloky obsaženy v této knihovně jsou pak pro nastavování UARTové komunikace a pro poskytování dalších potřebných funkcí při vytváření této komunikace mezi jednotlivými zařízeními.

„Rx Input“

Blok „*Rx Input“* slouží k příjímání dat po komunikaci UART. Při vložení do Simulinkovského modelu reprezentuje tento blok vstupní pin mikrokontroleru-Rx pin (R-Recive).

Tento blok je pak podrobněji popsán v kapitole 5 -Úprava komunikace UART u Car4.

„Tx Output“

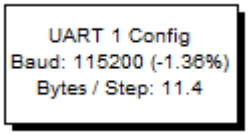
„Inverzním“ blokem k bloku „*Rx Input“* je blok *„Tx Output“*. Tento blok slouží pro vysílání dat při komunikaci UART. Při použití v Simulinkovském modelu tento blok reprezentuje výstupní pin mikrokontroleru – Tx pin (T-Transmit)

„UART Configuration“

Blok „*UART Configuration“* nastavuje parametry UARTu. Je možné zde nastavit jeden z více možných UARTů-číslo UARTu (množství UARTů závisí na použitém mikrokontroleru v bloku „*Master“*).

Každý UART obsahuje jeden Rx pin a jeden Tx pin, kde nastavené parametry UARTu jsou pro každý pin stejné.

V bloku UART Configuration lze nastavit především rychlost přenosu-Baud (kb/s). Tento blok pro nastavování parametrů UARTu obsahuje dále funkci, která vypočítá ze zadaných parametrů možný počet Bytů, který je takto nastavený UART schopný přenést za jeden časový krok simulace (Step).



Obr. 3.2. Blok *„UART Configuration“*

„TX Output Multiplexed for Matlab/Labview“

Pro komunikaci mezi mikrokontrolerem a prostředím Simulinku (případně LabVIEW) slouží blok „*TX Output Multiplexed for Matlab/Labview“*. Propojením sériového portu u PC a mikrokontroleru je možné posílat data z mikrokontroleru do PC-Simulinku.

U tohoto bloku je možné využít až 16 kanálů, na kterých je možné posílat osmi-bitová nebo šestnácti-bitová data.

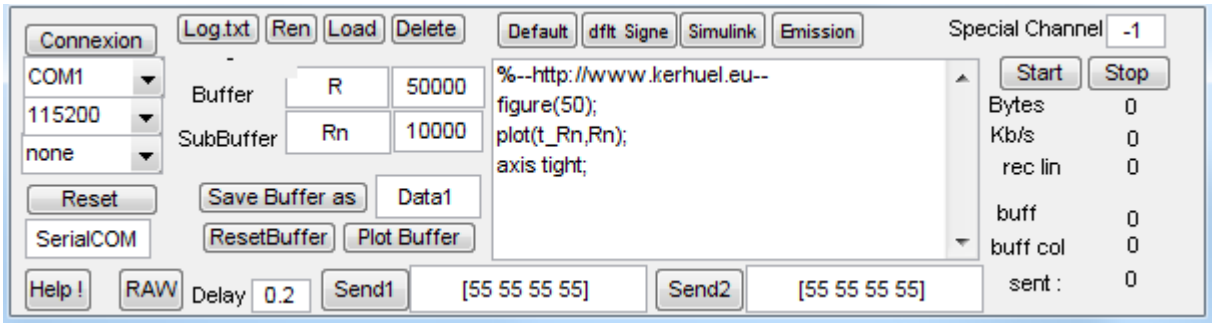
[7] UART- Universal Asynchronous Receiver and Transmitter

„Graphical interface“

Blok „*Graphical interface“* je uživatelské grafické rozhraní sloužící k propojení s blokem „*TX Output Multiplexed for Matlab/Labview“*. Tento blok obsahuje funkci pro posílání dat na Rx pin mikrokontroleru (pomocí sériového portu u PC) a následné zobrazovat data z bloku „*TX Output Multiplexed for Matlab/Labview“*, kde je možné tato data graficky zobrazit přímo předchystanou funkcí pro vykreslování grafů v Matlabu (funkce „*plot“*) pomocí příkazu „*plot Buffer“*.

V nastavení tohoto grafického rozhraní je nutné nastavit rychlost komunikace Baud (kb/s), která by měla být stejná jako v nastavení bloku „*UART Configuration“*. Dále je také potřeba vybrat správný sériový port PC (většinou COM1).

Při použití tohoto grafického rozhraní je nutné v prvním kroku otevřít sériovou komunikaci mezi PC a mikrokontrolerem-tlačítko „*Connexion“*. V druhém kroku, při správném nastavení, je možné tlačítkem „*Start“* zahájit přenos dat-spustit sériovou komunikaci vytvořenou v předchozím kroku.



Obr. 3.3. Otevřené grafické rozhraní bloku *Graphical interaface*

3.4.4. Knihovna *Others*

V této knihovně jsou obsažený „obslužné“ bloky. Jako je blok „*Software RESET“*, pro resetování mikrokontroleru (reset se provede v případě, když na vstupu bloku je nenulová hodnota).

Další z více bloků obsažených v knihovně „*Others“* je např. blok použití v kapitole 5, blok „*Calculus Time Step“* pro zjištění času kdy mikrokontroler vykonává nějaký výpočet.

3.4.4. Knihovna *Peripheral Mapping*

Tato knihovna obsahuje jediný blok – „*PIN Mapping“*, kterým je možné obsluhovat remapovatelné[8] piny, které mikrokontroler obsahuje (*dsPIC33FJ128MC804*). Tento blok slouží pro přiřazení jednotlivých periferii mikrokontroleru (PWM, UART, SPI, Timers, ECAN, ...) jednotlivým remapovatelným pinům na mikrokontroleru.

[8] „*remapovatelný pin“* znamená, že mikrokontroler nemá pevně definovaný pin, na kterém by se příslušná periferie musela nutně provozovat

3.5. Microchip Toolbox

Microchip Toolbox je přídavný toolbox do prostředí Simulinku. Do jisté míry je tento toolbox obdobou dříve zmiňovaného Kerhuel Toolboxu. Podporovány jsou zde řady mikrokontrolerů dsPIC30 a dsPIC33. Mikrokontrolery s remapovatelnými piny ovšem tímto toolboxem nejsou podporovány. Generování kódu zde opět funguje s podporou nástroje Real-Time Workshop, Target Language Compileru a příslušných Target files.

Právě z důvodu použitého mikrokontroleru, obsahujícího remapovatelné piny nebyl Microchip Toolbox použit pro praktické aplikace (kap.5, kap.6). Proto je v této části zmíněn spíše informativně. Jsou zde zmíněny možnosti tohoto Toolboxu, které nejsou obsaženy v dříve zmíněném Kerhuel Toolboxu.

Odlišnost od Kerhuel Toolboxu je v mírně odlišném složení bloků, které jsou prostřednictvím Microchip Toolboxu k dispozici pro propojení portů a periferii mikrokontrolerů s modelem v Simulinku.

Tento toolbox obsahuje, podobně jako Kerhuel Toolbox, několik tematicky rozdělených knihoven. Z toho je úzký výběr uveden níže.

3.5.1. Knihovna *Configuration*

Tato knihovna obsahuje základní bloky pro nastavení modelu pro příslušný mikrokontroler – „*dSPIC33fxxMain“* a „*dSPIC30fxxMain“*, bloky pro nastavení periferii mikrokontroleru (ADC, PWM, UART), blok pro nastavení vstupu/výstupu na příslušném portu. V neposlední řadě obsahuje také blok pro nakonfigurování celého modelu – „*Config“*.

3.5.2. Knihovna *Run Time Library*

Tato knihovna obsahuje např. bloky pro zapisování/čtení na výstupní/vstupní digitální porty mikrokontroleru, bloky pro UARTovou komunikaci („*UART Recive/Transmit“*), atd.. Tato knihovna obsahuje také blok pro zápis dat do konkrétního registru (paměti) mikrokontroleru- „*Write Memory/Register“*. Je tak možné jednoduchým zadáním příslušné adresy do nastavení bloku v hexadecimální formě zapsat data do požadovaného registru.

Knihovna dále obsahuje veškeré vstupně/výstupní bloky, jako např. výstup z AD převodníku – „*ADC Read“*. Je zde i blok pro přímou obsluhu LCD displeje-„*dsPIC33f LCD Display“*. Tento blok umožňuje posílat znaky zadáním příslušného znaku číselnou hodnotou z ASCII tabulky.

3.5.3. Knihovna *QMathLib*

V této knihovně jsou obsaženy základní matematické operace (sin, cos, abs,...) optimalizované pro dsPIC.

3.6. Shrnutí možností použití Toolboxů v prostředí Simulink

Propojením patřičného Toolboxu (zejména Kerhuel Toolboxu) a prostředí Simulinku za účelem generování kódu pro cílový mikrokontroler je z takto získané struktury získán velmi efektivní nástroj pro jednoduché programování mikrokontrolerů, bez velké znalosti programovacího jazyka. Především Kerhuel Toolbox, který je již odladěný a vyzkoušený na řadě aplikací. Kerhuel Toolbox také podporuje větší řadu mikrokontrolerů, především ty s remapovatelnými piny, což dělá z Kerhuel Toolboxu vhodnou volbu.

Takový nástroj poskytuje značné možnosti využití v mechatronických návrzích, kde je důležitý rychlý a efektivní vývoj dané aplikace.

4 Generování kódu v prostředí LabView

LabVIEW je prostředí určené ke grafickému programování zahrnující vývoj sofistikovaného měření, testování a řízení systémů použitím intuitivních grafických bloků a spojovacích drátů.

Grafické programování, pomocí kterého se v LabVIEW tvoří „program“ je možné pomocí integrovaných nástrojů převést do jazyka C a následně použít na cílovém zařízení.

Tato krátká kapitola nemá za cíl seznámit čtenáře s podrobným popisem generování kódu, ale má jen nastínit možnost generovat kód v LabVIEW.

4.1. LabVIEW C code Generator

Nástroj *LabVIEW C code Generator* slouží pro vytvoření programové struktury v jazyce C pomocí jednoduchého grafického programování v prostředí LabVIEW.

Tento nástroje je jakýmsi prostředníkem mezi návrhem systému a jeho následnou implementací do cílového zařízení. LabVIEW C code generator z navrženého souboru VI (virtual instrument) jednoduše vygeneruje optimalizovaný kód v jazyce C.

S použitím nástroje LabVIEW C generator je možné vytvořit cílově nespecifický kód (v jazyce ANSI C), bez potřeby znát cílovou platformu, pro kterou je kód z LabVIEW generován. [8]

4.2. Stručná představa průběhu generování kódu

Průběh, na jehož konci je vygenerovaný kód, je následující. Nejprve se v LabVIEW vytvořením nového projektu nastaví specifikace pozdějšího generování kódu v jazyce C- „*Build Specification“*. Tyto specifikace definují, jak se bude kód C generovat z vytvořeného souboru VI. Po takto vytvořeném projektu se vytvoří již zmíněný soubor VI kde je „graficky naprogramovaný“ systém. Pravým klinutím na „*Build Specification“*, které se vytvořili dříve se vybráním položky „*Build“* vygeneruje požadovaný kód.[9]

5 Úprava komunikace UART u Car4

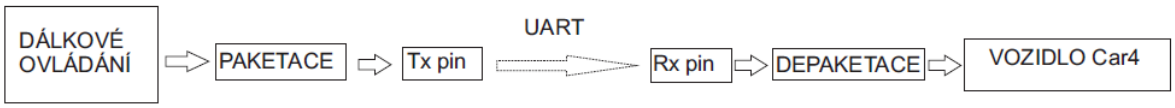
5.1. Úvod ke komunikaci u experimentálního vozidla Car4

Car4 [5] je experimentální vozidlo se čtyřmi samostatně hnanými koly. Přičemž u každého kola je možné řídit samostatně jeho natočení.

Informace sloužící pro základní řízení vozidla Car4 jsou: hodnota momentu na hnacích elektromotorech, natočení přední nápravy, natočení zadní nápravy. Přičemž hodnoty těchto „informací“ musí být vysílány z nějakého zdroje – z ovládání (samostatně řešeno v kapitole č.6). Cílem této části práce je vytvořit lepší příjem dat po UARTu oproti stávajícímu příjmu, který umožňoval příjem jediného Bytu v každém časovém kroku řídící smyčky. K realizaci je použito prostředí Simulinku a Kerhuel Toolboxu.

Struktura a celý algoritmus byl navržen pro UARTovou komunikaci mezi mikrokontrolerem *dsPIC33FJ128MC804* a PC, případně dvěma mikrokontrolery.

5.2. Řešení algoritmu příjmu paketu devíti bytů



Obr. 5.1. Komunikace mezi dálkovým ovládáním a vozidlem Car4.

Na Obr. 5.1. je graficky znázorněna komunikace mezi dálkovým ovládáním a vozidlem Car4. Kde jsou data vysílána z dálkového ovládání ve formě čtyř dvanáctibitových hodnot. V bloku PAKETACE se tyto čtyři dvanáctibitové hodnoty převedou na šest osmibitových bytů, dále se zde k těmto šesti „datovým“ bytům přidají dva kontrolní byty pro provedení kontrolního součtu při příjmu těchto dat-dva CRC[9] byty (pro provedení kontrolního součtu). K takto vytvořeným osmi bytům se přidá jeden tzv. START byt pro zjištění začátku datové zprávy-devíti bytového paketu. Po tzv. paketaci se data posílají po sériové komunikaci (UART) z vysílacího Tx pinu na přijímací Rx pin. Po přijetí zprávy jsou data dále opět depaketována. Z takto získaných dat jsou opět získány informace pro řízení vozidla Car4.

„Vylepšení“ spočívá v příjmu více bytů v jednom kroku simulačního modelu vytvořeného v prostředí Simulink. Celý algoritmus příjmu dat je řešen tedy pomocí bločků v Simulinku, kde se vytvoří model celého příjmu. Takto vytvořený model v Simulinku je pomocí nástrojů v Simulinku zkompilován do jazyka C, ze kterého je následně vytvořen binární soubor (s příponou .hex) umožňující přímé nahrání vytvořeného algoritmu příjmu dat do cílového mikrokontroléru.

[9] CRC-*Cyclic redundancy check (Cyklický redundantní součet)*-sloužící k detekci chyb během přenosu

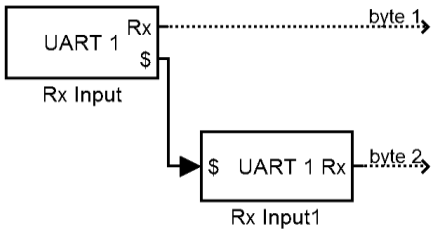
Celé toto vylepšení příjmu bytů při komunikaci po UARTu je možné především díky novější verzi Kerhuel Toolboxu (použitá verze R3.4), dále pak použitím funkce „*Block Ordering Input/Output“*. Tato funkce umožní to, že je v jednom časovém kroku Simulinkovského modelu postupně v každém bloku UARTové komunikace „*Rx Input“*, po „spojení“ těchto bloků, přijmuto všech devět bytů patřící do jedné datové zprávy.

Předchozí verze[10] tuto funkci nepodporovala, nemohl být tedy použitím Kerhuel Toolboxu zajištěn příjem více bytů v jednom výpočetním kroku mikrokontroleru. Vylepšení tedy spočívá především v urychlení UARTové komunikace.

5.2.1. Použité bloky z Kerhuel Toolboxu

Při vytváření algoritmu příjmu paketu devíti bytů jsou použity bloky Kerhuel Toolboxu – *„Rx Input“* (knihovna „*Serial PORT“*) a jejich následné „propojení“ pomocí funkce „*Block Orderig Input/Output“* .

Při použití funkce „*Block Orderig Input/Output“* je k dispozici nový vstup (výstup) bloku *Rx Input* se znakem *$*, který slouží k propojení bločků vzájemně mezi sebou. Následně takto vytvořená struktura bloků *Rx Input* zajistí postupný příjem bytů od prvního bytu až po poslední, kde podle pořadí první byt, je přijat na výstupu toho bločku, který je umístěn jako první v řetězci propojených bloků *Rx Input.* Takto vytvořená část kódu pro mikrokontroler z modelu, kde je takto použitá funkce „*Block Orderig Input/Output“*, zajistí požadovaný příjem devíti bytů v jednom výpočetním kroku mikrokontroleru v určeném pořadí.



Obr. 5.2. Propojení bločků Rx Input pomocí funkce Block Ordering Input/Output

5.2.2. „Nalezení“ kompletní datové zprávy

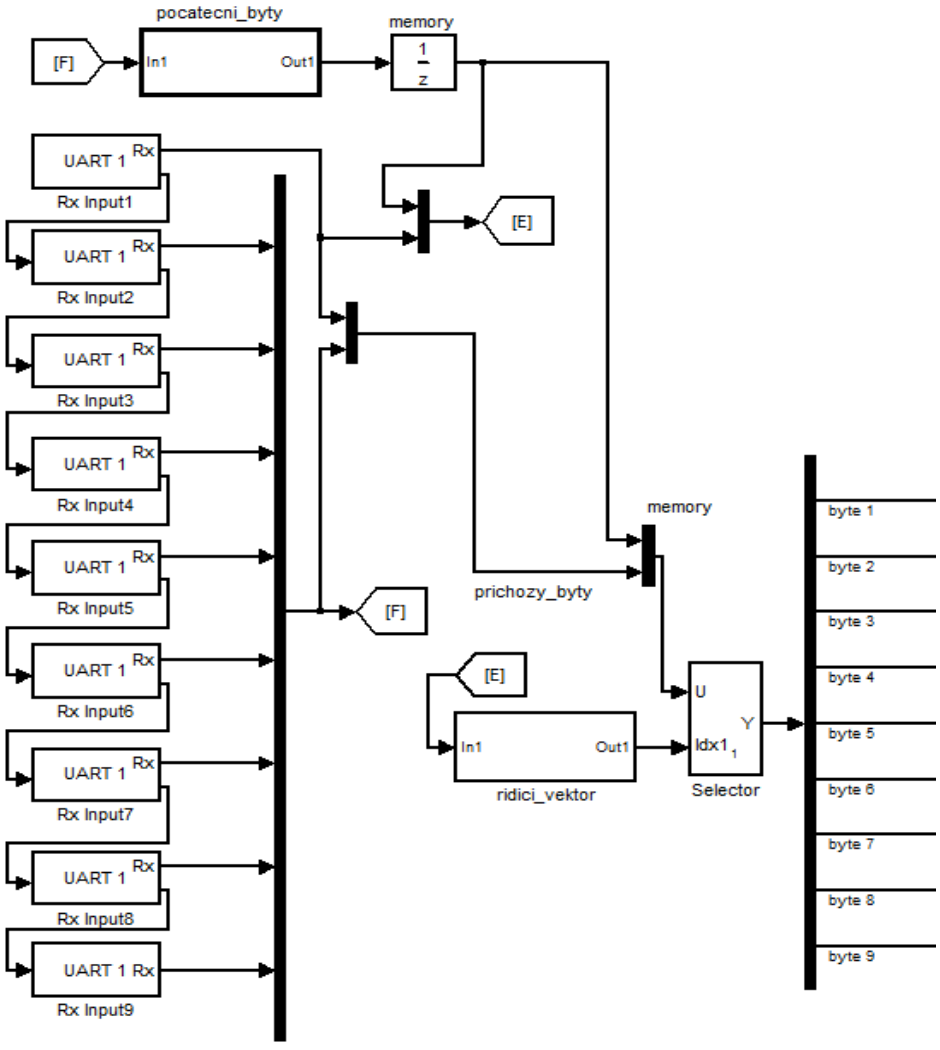
Při posílání jednotlivých bytů po sériové lince –UART nelze zajistit to, aby se jednotlivé byty začaly přijímat od tzv. START bytu, který signalizuje začátek jedné datové zprávy (devítibytového paketu). Je tedy nutné počítat s tím, že část jedné datové zprávy bude přijata v jednom výpočetním kroku mikrokontroleru a zbylá část pak v dalším kroku. Žádoucí je pak tyto dvě části najít a poskládat z nich jednu devítibytovou datovou zprávu pro následné depaketování. Pro úspěšné nalezení všech devíti bytů jedné kompletní datové zprávy je tedy nutné uložit do paměti předchozích osm bytů, které byly získány v předchozím výpočetním kroku mikrokontroleru (na obr. 5.3. je to blok s názvem „*memory“*).

[10] funkce „*Block Ordering Input/Output“* je dostupná v Kerhuel Toolboxu až od verze 3.3c (s datem vydání 24.12.2010)

Tímto způsobem se v každém cyklu[11] získá vždy devět nových bytů a osm bytů z předchozího cyklu. Z těchto sedmnácti bytů je potřeba najít a selektovat hledanou datovou zprávu začínající tzv. START bytem, který má v tomto případě hodnotu rovnu číslu 254.

Hlavní částí tohoto algoritmu hledání požadované datové zprávy je blok s názvem „*ridici_vektor*“, ve kterém se podle tzv. START bytu nalezne začátek datové zprávy a následně se zjistí vektor pozic bytů, které jsou součástí hledané datové zprávy.

Takto nalezený „řídící“ vektor je pak použit v bloku „*Selector*“, který podle tohoto vektoru selektuje z původních sedmnácti bytů (složených z nově příchozích devíti bytů a osmi bytů uložených do paměti z předchozí série devíti bytů) pouze devět hledaných.

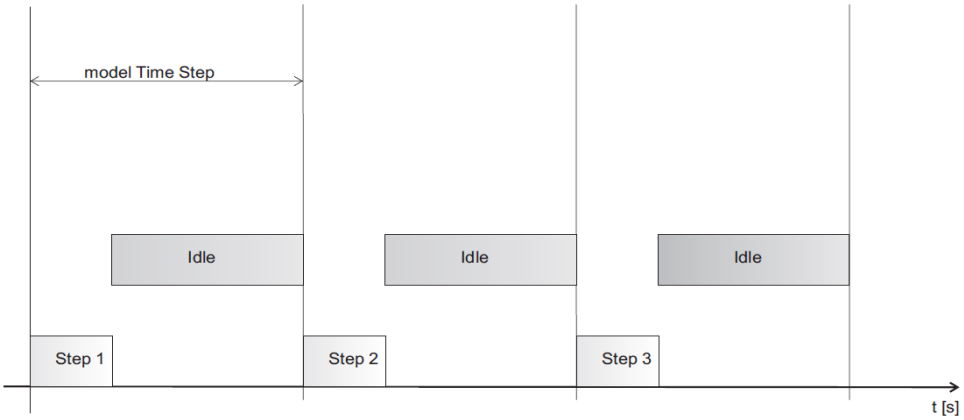


Obr. 5.3. Základní schéma algoritmu pro „nalezení“ jedné kompletní datové zprávy.

[11] jeden cyklus = jeden výpočetní krok mikrokontroleru

5.3. Zjištění vytíženosti mikrokontroleru

Časový krok výpočtu, vykonávání v mikroprocesoru lze rozdělit do dvou částí. První část výpočetního časového kroku, je část, ve které se vykonává program uložený v paměti procesoru (Step1, Step2, Step3,...). Následuje druhá část časového kroku, ve které mikroprocesor čeká(Idle) na čas, kdy bude ukončen časový krok. V okamžiku ukončení jednoho časového kroku[12] (model Time Step) začíná další časový krok, který probíhá stejným způsobem.



Obr. 5.4. Grafické znázornění rozložení jednoho časového kroku výpočtu probíhajícího v mikrokontroleru.

Cyklus algoritmu hledání datové zprávy a následné depaketace se provádí v časovém kroku 5ms (model Time Step). Pro správné fungování sériové komunikace UART je potřeba zajistit s dostatečnou rezervou „stíhání“ mikrokontroleru provádět takto vytvořený algoritmus-pro přijímání devíti bytové datové zprávy. Je potřeba, aby vytíženost byla dostatečně malá vzhledem k tomu, že na mikrokontroleru nepoběží jen takto jednoduchý model příjmu datové zprávy, ale poběží na něm celý model Car4.

Mikrokontroler tedy každých 5ms „volá“ tuto funkci, určitý čas mikrokontroleru zabere výpočet tohoto algoritmu. V případě, že by výpočet trval déle než dříve uvedených 5ms, algoritmus celého hledání datové zpráva a následné depaketace by se nevykonal celý a přijímaná datová zpráva by nebyla kompletní (komunikace by selhala). Proto je nutné zjistit čas, který mikrokontroler potřebuje pro výpočet celého algoritmu.

[12] časový krok *„model Time Step“* je shodný s parametrem „*Sample Time“* nastaveném v modelu Simulinku (nastavení časového kroku je možné v záložce *Simulation/Configuration Parameters)*

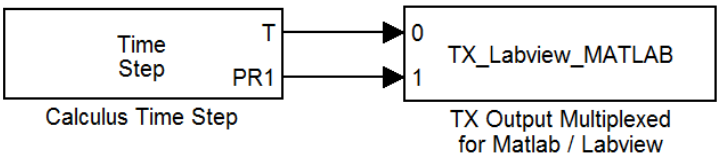
5.3.1. Výpočetní vytíženost mikrokontroleru zjišťovaná pomocí bloku v Kerhuel Toolboxu

Výpočetní vytíženost lze v Simulinku snadno zjistit za pomocí Kerhuel Toolboxu a v něm použitém bloku „*Calculus Time Step"* (knihovna *OTHERS*).

Tento blok se vloží do cílového modelu, ve kterém který má cíl zjistit vytíženost mikrokontroleru (v tomto případě je použit modelu příjmu a depaketace devítibytové datové zprávy.

Blok „*Calculus Time Step"* má dva výstupy. Na prvním výstupu, označeným znakem *T*, je vždy hodnota, která je závislá na čase, kdy mikrokontroler vykonává nějakou činnost (provádí nějaké výpočty). Hodnota výstupu *T* odpovídá počtu instrukcí potřebných k provedení daného algoritmu mikrokontrolerem (na Obr. 5.4. tato hodnota odpovídá hodnotě *Step1, Step2,...*) během jednoho časového kroku. Čím větší hodnota *T,* tím více instrukcí potřebuje mikrokontroler k výpočtu daného algoritmu, tím pádem potřebuje k celému vyřešení algoritmu i delší dobu.

Hodnota na druhém výstupu *PR1* (Period Register) je počet instrukcí, které mikrokontroler stihne vykonat během jednoho časového kroku. Při nastavení velkého časového kroku *(Simulation/Configuration Parameters/Sample Time*) je tedy i větší hodnota *PR1* a tím má i mikrokontroler více prostoru pro vykonání požadovaného algoritmu.



Obr. 5.5. Blok Calculus Time Step připojený do bloku TX Output Multiplexed for Matlab/Labview.

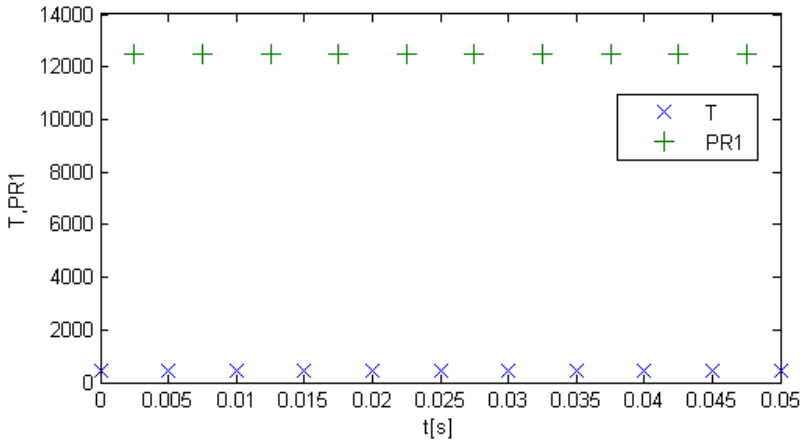
Blok „*Calculus Time" Step* je použit společně s blokem „*TX Output Multiplexed for Matlab/Labview"*, který slouží jako „zobrazovací jednotka" pro hodnoty *PR1* a *T* z bloku „*Calculus Time" Step.* Hodnoty *PR1* a *T* je možné odečíst přímo z grafického zobrazení v bloku „*TX Output Multiplexed for Matlab/Labview",* nebo přímo ve Workspace Matlabu (kde se hodnoty odečtou přesně).

Hodnota registru *T* byla tedy zjištěna na čísle 428, PR1 pak na čísle 12499.
Výslednou vytíženost mikrokontroleru je pak možné zjistit ze vztahu (1). Z grafu 5.6. je pak vidět, že v každém po sobě jdoucích časových kroků je vytíženost mikrokontroleru konstantní (není také důvod, aby se vytíženost lišila).

$$\text{vytíženost} = \frac{T}{PR1} \cdot 100[\%] \qquad (1)$$

$$\text{vytíženost} = \frac{428}{12499} \cdot 100[\%]$$

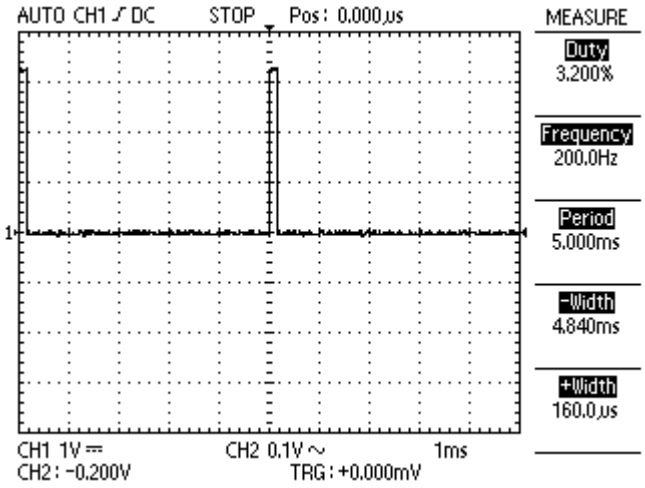
$$\text{vytíženost} = 3{,}424\ \%$$



Graf 5.6. Grafické znázornění hodnot T a PR1 z bloku „*Output Multiplexed for Matlab/Labview*“ v několika po sobě jdoucích časových krocích (jeden časový krok = 5 ms).

5.3.2. Ověření výpočtu vytíženosti s použitím osciloskopu

Dalším způsobem jak zjistit vytíženost mikrokontroleru je použití funkce „*Busy Flag Port"* v bloku „*Master"*. Tato funkce má dva stavy. Je v logické jedničce v čase, kdy mikrokontroler vykonává nějaké výpočty daného algoritmu. Je v logické nule, v čase kdy mikrokontroler nevykonává žádné výpočty a už jen pouze čeká na ukončení právě probíhajícího výpočetního kroku mikrokontroleru (viz. Obr. 5.4.). Pro sledování průběhu vytíženosti je potřeba tuto funkci vyvést na volný pin mikrokontroleru, na kterém se připojením osciloskopu sleduje „průběh vytíženosti" (obr. 5.7.).



Obr.5.7. Obraz z osciloskopu při měření vytíženosti mikrokontroleru.

Na osciloskopu je pak patrný obdélníkový průběh stavu kdy mikrokontroler vykazuje nějakou činnost a stavu, kdy mikrokontroler čeká na ukončení jednoho výpočetního kroku. Z dat naměřených na osciloskopu je vidět, že perioda s jakou mikrokontroler volá námi vytvořený program (pro přijímání a depaketaci datové zprávy) se shoduje s časovým krokem nastaveným v modelu Simulinku (Fixed-step size) - nastavený časový krok v modelu Simulinku byl 5ms.

Při zjišťování vytíženosti mikrokontroleru je nejdůležitějším údajem na osciloskopu položka „*Duty"*-střída, která udává poměr doby, kdy mikrokontroler vykonává výpočet (Step) a doby, kdy je v klidové části(Idle) a čeká na ukončení právě probíhajícího výpočetního kroku mikrokontroleru (model Time Step).

	Perioda časového kroku [µs]	Vytíženost mikrokontroleru [%]	Čas výpočtu mikrokontroleru [µs]
Calculus Time Step	5000	3,424	171
Busy Flag Port + osciloskop	5000	3,200	160

Tab. 5.8. Srovnání výpočtu vytíženosti pomocí bloku „*Calculus Time Step"* a pomocí funkce „*Busy Flag Port"* + osciloskop.

5.4. Shrnutí využití Simulinku při tvorbě programu pro UARTovou komunikaci a následné ověřování vytíženosti mikrokontroleru

Simulink a v tomto prostředí použitý Kerhuel Toolbox poskytuje značné zjednodušení při vytváření sériové komunikace mezi dvěma zařízeními. V tomto případě se jednalo o UARTovou komunikaci (především pak příjem dat).

Využití již připravených boků v Kerhuel Toolboxu značně zjednoduší práci a poskytne tak možnost vytvořit celý algoritmus komunikace pomocí bloků v Simulinku, kde je v schématu pak do jisté míry vidět, jak celá komunikace funguje. Dobře využitelnou funkcí v Kerhuel Toolboxu se pak zdají funkce pro zjišťování vytíženosti mikrokontroleru. Je možné velice jednoduchým způsobem za velmi krátký čas zjistit, zda námi vytvořený program – model v prostředí Simulink je mikrokontroler při dané taktovací frekvenci (nastaveném počtů instrukcí za sekundu v bloku „*Master"*) schopen zvládnout vykonat. Velmi vhodnou metodou pro zjištění vytíženosti při potřebě rychlého ověření vytíženosti, je pak dříve uvedená funkce bloku „*Master"*- „*Busy Flag Port"* s použitím osciloskopu.

Ve srovnání výsledné vytíženosti mikrokontroleru s použitím metody měřením na osciloskopu a metody s použitím bloku „*Calculus Time Step"* je v dříve uvedené porovnávací tabulce (Tab. 5.8.) vidět malý rozdíl ve výsledné vytíženosti. Metoda s použitím bloku „*Calculus Time Step"* je zajisté přesnější, protože výsledná vytíženost je určována z „naměřených" počtů instrukcí provedených mikrokontrolerem v době kdy provádí výpočty a celkového výpočetního kroku tohoto mikrokontroleru. Zatímco metoda s použitým osciloskopem může být ovlivněna jednak okolním rušením, ale zejména může rozdíl mezi dříve zmíněnou metodou být zapříčiněn vzorkovací frekvencí použitého osciloskopu. Průběh vytíženosti mikrokontroleru změřený osciloskopem by pak mohl být delší (kratší), než ve skutečnosti je, z čehož plyne jistá nepřesnost. Výhoda použití osciloskopu spočívá v tom, že není potřeba využívat linky UART ke komunikaci s PC.

6 Dálkové ovládání pro Car4 postavené na DS PIC

6.1. Úvod-základní požadavky na dálkové ovládání pro Car4

Cílem je vytvořit samostatné dálkové ovládání pro ovládání experimentálního vozidla Car4 [5]. Dálkové ovládání by mělo umožnit ovládat Car4 bez použití PC. Jako dálkové ovládání musí také splňovat požadavek na mobilitu. Základním stavebním prvkem je použitý mikrokontroler PIC, konkrétně *dsPIC33FJ128MC804*, který obsahuje všechny potřebné periferie (UART, ADC, I/O, ...).

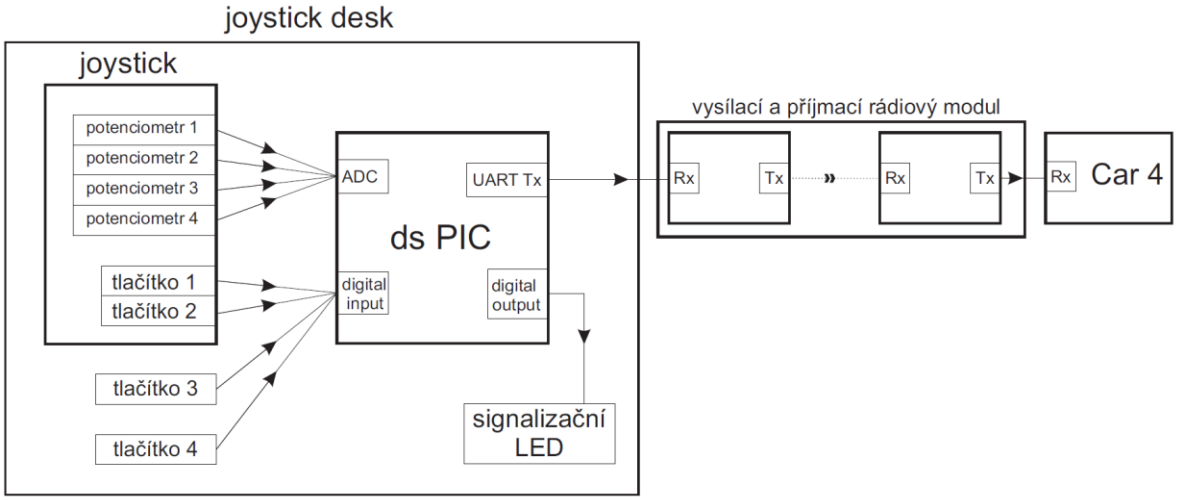
Díky použitému, již hotovému, vysílacímu a přijímacímu modulu, který je využit při návrhu dálkového ovládání jako bezdrátová sériová komunikace (bezdrátový UART) je návrh značně zjednodušen.

Shrnutí požadavků na dálkové ovládání:

- založeno na dsPIC
- možnost ovládat dvě nápravy a jednu rychlost/moment-potřeba min. tří os
- tlačítka pro nastavování řízení (mód řízení, rychlostní stupeň,...)
- LED diody pro signalizaci
- možnost pozdějšího připojení displeje
- volné konektory-pro připojení další elektroniky (např. akcelerometr, ...)

6.2. Základní schéma dálkového ovládání

Na Obr.6.1. je zobrazené zjednodušené schéma dálkového ovládání. Základem tohoto celku je mikrokontroler dsPIC, ovládací prvky, signalizační LED a pak také modul pro bezdrátové vysílání dat do experimentálního vozidla Car4, který zajistí přenos dat z dálkového ovládání do experimentálního vozidla Car4 po komunikaci UART.



Obr. 6.1. Základní schéma dálkového ovládání pro Car4 postavené na dsPIC

Pro vozidlo Car4 je nutné ovládat především jeho směr, a rychlost. K nastavování těchto „veličin“ bude u dálkového ovládání sloužit dvojce joysticků, kde každý obsahuje dva potenciometry. Takto použitý potenciometr slouží jako napěťový dělič. Při pohybu joysticku v jeho ose se nastavuje natočení daného potenciometru. Podle dané polohy páčky joysticku (a natočení potenciometru) se plynule mění odpor a tím i výstupní napětí na potenciometru. Takto nastavované napětí je pomocí AD převodníku převedeno na digitální signál, který se dále zpracovává v dsPIC. Digitální signál je reprezentován dvanáctibitovou hodnotou na výstupu AD převodníku.

Jako další ovládací prvky je použita skupina čtyř tlačítek. Zejména tlačítka obsažená v každém ze dvou joysticků, která budou sloužit k nastavování rozsahu rychlosti, neboli bude možné „řazení“. Rychlost tedy bude možné nastavovat na různé stupně rychlosti (velice pomalá jízda, pomalá jízda, rychlá jízda, atd...).

Výstupem z dsPIC je pak především jeho Tx pin, který je dále připojen k Rx pinu vysílacího bezdrátového modulu (další zpracovávání dat bylo již zmíněno v předchozí kapitole č.5).

Jako signalizační prvek je k digitálním výstupním pinům dsPIC připojena skupina LED diod. Takto připojené LED mohou zobrazovat např. „stupeň rychlosti“ navolený na dálkovém ovládání pomocí tlačítek.

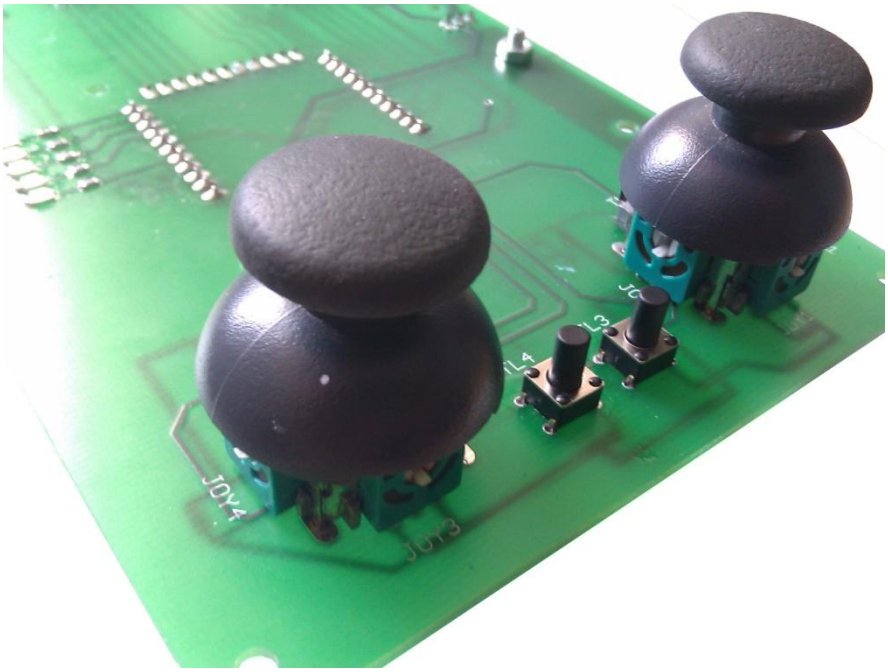
6.3. Elektronika dálkového ovládání

Základem dálkového ovládání je deska plošných spojů (DPS), na které jsou umístěny již zmíněné prvky, jako jsou joysticky, tlačítky, diody. Na takto vytvořené desce je umístěn modul s mikrokontrolerem (*dsPIC33FJ128MC804*) osazen krystalem, stabilizátorem pro napájecí napětí mikrokontroleru (3,3V) a dalšími nezbytnými součástkami pro správnou funkci mikrokontroleru. Pro snížení šumu na výstupu joysticků je u každého potenciometru paralelně připojen kondenzátor 100nF pro filtraci napětí.

Další prvky na základní desce jsou kromě napájecí části (stabilizátor na 5V) také vyvedené volné piny mikrokontroleru na konektorech, které jsou určený pro další využití mikrokontroleru (např. displej). Nejdůležitějším konektorem na takto vytvořené desce je konektor UART, na který je připojen vysílací (Tx) pin, přijímací (Rx) pin a následně pak také zem elektroniky.

U každého tlačítka, které je následně připojeno na digitální vstup mikrokontroleru je připojen „pull-down“ rezistor. Tento rezistor zajistí menší proud tekoucí do mikrokontroleru při sepnutém stavu (logická jednička), při rozpojeném stavu zajistí připojení digitálního vstupního pinu na zem elektroniky, čímž je na digitálním vstupu mikrokontroleru zajištěna logická nula.

Návrh DPS pro dálkové ovládání byl proveden v programu *Eagle*[13]. Celé elektrické schéma a návrh DPS je umístěn v příloze.



Obr. 6.2. Ovládací prvky použité pro dálkové ovládání-především dvojce joysticků

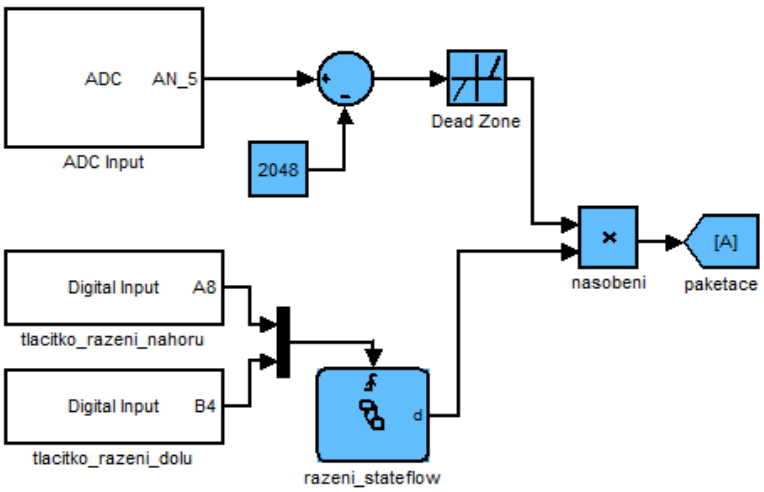
[13] *Eagle*-software pro návrh elektrických plošných spojů. Použitá verze 5.7.0

6.4. Firmware dálkového ovládání-model v Simulinku

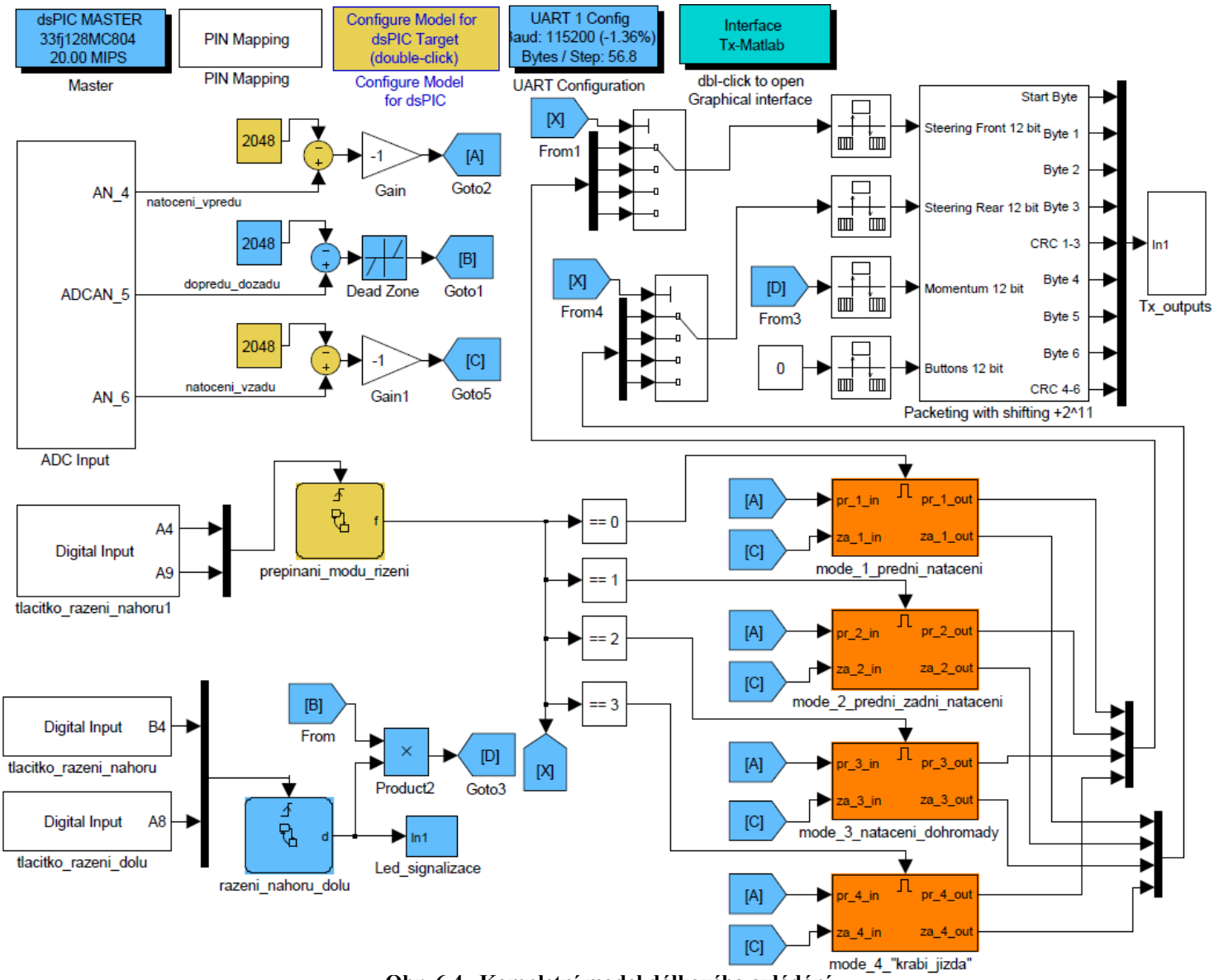
Programové vybavení dálkového ovládání je vytvořeno v prostředí Simulinku, vytvořením modelu algoritmu funkce dálkového ovládání s použitím nástroje pro generování kódu (Real-Time Workshop, atd...). Nutnou součástí pro generování kódu pro mikrokontroler dsPIC je také Kerhuel Toolboxu, který je použit pro „propojení“ Simulinkovského modelu a použitého mikrokontroleru dsPIC. Model v Simulinku musí obsahovat potřebné bloky, pro správné nastavení mikrokontroleru, zejména blok „*Master“*. Mikrokontroler *dsPIC33FJ128MC804* obsahuje remapovatelné periferie, Simulinkovský model musí proto obsahovat blok pro nastavení pinů pro příslušnou periferii - blok *PIN Mapping*. Při návrhu dálkového ovládání tento blok slouží hlavně pro nastavení pinů UARTové komunikace (Tx pin, Rx pin).

Pro ovládání rychlostního stupně byl v prostředí Simulink použit nástroj Stateflow, který je doplňkem klasického Simulinku. Stateflow umožňuje vytvořit tu část programu pro dálkové ovládání, která by s klasickým Simulinkem byla obtížně vytvořitelná. Jedná se hlavně o „reagování“ na vstupní impulzy z tlačítek pro ovládání rychlostního stupně (jedno tlačítko pro snižování a jedno pro zvyšování rychlostního stupně). Každé toto tlačítko je připojeno na digitální vstup mikrokontroleru. V modelu Simulinku jsou proto pak použity bloky „*Digital Input“*, z kterých je pak výstup ve formě logické jedničky nebo logické nuly. Pomocí části modelu vytvořeného v prostředí Simulink/Stateflow je pak dosaženo toho, že výstupem bloku „*razeni_stateflow“* je pak hodnota v rozsahu 0-1. Z bloku „*Digital_input“* je tedy vyveden signál ve formě obdélníkového signálu, který je přiveden do řídícího vstupu bloku „*razeni_stateflow“*. Přepínání rychlostního stupně je v tomto bloku nastaveno na každou náběžnou hranu signálu z tlačítek. Je li tedy v řídícím signálu bloku razeni_stateflow např. dvakrát zaznamenána náběžná hrana signálu z bloku „*tlacitko_razeni_nahoru“*, je pak na výstupu bloku „*razeni_stateflow“* navolena v pořadí druhá „násobící“ konstanta rychlostního stupně.

V programu pro dálkové ovládání je volba rychlostního stupně jednoduše provedena násobením dvanáctibitové hodnoty, která je výstupem z AD převodníku (posunutou na rozsah -2^{11} až 2^{11} viz dále) a „násobící“ konstanty rychlostního stupně (výstup *d*). Výstupem z AD převodníku je sice hodnota v rozsahu $0\text{-}2^{12}$ (2^{12} = 2048), pro pozdější paketaci je od této hodnoty odečtena hodnota 2^{11}, čímž se dostaneme do rozsahu od -2^{11} do 2^{11}. Kde hodnota -2^{11} reprezentuje zápornou rychlost a kladná hodnota reprezentuje kladnou rychlost. Toto posunutí rozsahu je pak vhodné pro pozdější kalibraci joysticků, kdy lze jednoduše nastavit nulovou hodnotu.



Obr. 6.3. Část modelu vytvořeného v prostředí Simulink- volba rychlostního stupně pomocí Stateflow



Obr. 6.4. Kompletní model dálkového ovládání

6.4.1. Popis modelu dálkového ovládání

Na Obr. 6.4. lze vidět kompletní model dálkového ovládání vytvoření v prostředí Simulink. V levé části jsou umístěny „vstupní“ bloky z použitého Kerhuel Toolboxu, jako jsou *„ADC input“* a *„Digital Input“*. Na pravé straně jsou pak umístěny bloky „*Tx Output“*, sloužící jako „výstupní“ bloky. Bloky „*Tx Output“* jsou „pospojovány“ funkcí „*Block Orderig Input/Output“*, jejíž podrobnější popis je proveden v kap. 5.

Pomocí vstupního obdélníkového signálu, vytvořeného použitými tlačítky, je docíleno blokem *„prepinani_modu_rizeni“* (blok Stateflow a čtveřice bloků *„Enabled Subsystém“*- bloky zvýrazněné oranžovou barvou) volení vždy jednoho, ze čtyř předchystaných módů řízení:

- mode1 – ovládání natáčení pouze přední nápravy, zadní náprava je pevně nastavena na nulové poloze (lepší řízení při vyšších rychlostech)
- mode2 – ovládání obou náprav, nezávisle na sobě (vždy jeden z joysticků ovládá pouze jednu nápravu)
- mode3 – ovládání obou náprav pouze jedním joystickem, kdy se přední náprava natáčí podle nastavené polohy na joysticku, zadní náprava se pak natočí stejně, jen opačným směrem (lepší řízení v nižších rychlostech)
- mode4 - ovládání obou náprav, jedním joystickem, kdy se obě nápravy natáčí stejným směrem (je pak možná jízda do boku – tzv. „krabí chod“)

6.4.2. Kalibrace dálkového ovládání

Do modelu dálkového ovládání bylo nutné zahrnout části pro kalibraci dálkového ovládání-nastavení středu páčky joysticku a tím i nulové rychlosti, nebo nulové natočení kol. Předchozí část kalibrace byla provedena přičítáním konstanty (případně odečítáním konstanty) na výstupu AD převodníku. V lepším případě by tato část „přičítání konstanty“ mohla být nahrazena přičítáním „hodnoty“ z výstupu připojeného potenciometru na další AD převodník, který by nastavil nulovou polohu řízení lépe podle potřeby.

Další blok, který je použit pro potřeby kalibrace je blok „*Dead Zone“*, který definuje polohu páčky joysticku, která se ještě považuje za nulovou polohu (potřeba necitlivosti na malé výchylky). Nastavení „necitlivé polohy“, pomocí dříve uvedeného bloku *„Dead Zone“* bylo nastaveno postupně, zkoušením různě velké „necitlivosti“.

6.4.3. Použité bloky z Kerhuel Toolboxu

Blok ***„ADC input„***

Blok analogově-digitálního převodníku („*ADC input“*) slouží jako vstupní blok do modelu dálkového ovládání. Tento blok reprezentuje hodnoty nastavené na potenciometrech joysticků, kde je rozsah napětí 0V-3,3V. V nastavení tohoto bloku je možné nastavit typ převodníku, který spočívá v nastavení rozlišení AD převodníku.

Použitý mikrokontroler *dsPIC33FJ128MC804* umožňuje využít desetibitový a dvanáctibitový AD převodník. Pro převod analogové hodnoty napětí (0V-3,3V) je využit dvanáctibitový AD převodník, který umožní získat hodnotu z výstupu bloku „*ADC input“* v rozsahu $0\text{-}2^{12}$ (hodnota v decimální soustavě). Při použití AD převodníku pro ovládání natáčení kol a rychlosti jsou využity tři kanály (4,5,6) tohoto převodníku z celkových devíti možných (0,1,...,7,8).

Blok „***Digital Input“ a „Digital Output WRITE“***

Bloky Digital Input reprezentují v modelu Simulinku veškerá tlačítka dálkového ovládání. Na dálkovém ovládání jsou celkem obsáhnuta čtyři tlačítka.

Pro indikaci stavu aktuálně nastaveného rychlostního stupně na LED diodách je použit blok „*Digital Output WRITE“*.

Jediným nastavení v těchto blocích je nastavení příslušného portu a pinu mikrokontroleru, který je spojen s příslušným zařízením na desce dálkového ovládání.

Blok „***Tx Output“***

Tento blok je použit jako hlavní výstupní blok z celého modelu. Blok „*Tx Output“* reprezentuje výstupní pin mikrokontroleru-vysílací Tx pin v UARTové komunikace mezi mikrokontrolerem a modulem pro bezdrátové vysílání dat.

6.5. Zhodnocení návrhu a následné funkčnosti dálkového ovládání

Výrobu dálkového ovládání lze rozdělit do dvou částí. První část spočívala v návrhu a následné výrobě elektroniky. Tato fáze výroby dálkového ovládání byla založena na návrhu jednoduché desky, která byla osazena potřebnými ovládacími prvky především pro ovládání směru a rychlosti vozidla Car4. Samotná deska byla zadána specializované firmě[14], osazení takto vyrobené desky již bylo provedeno svépomocí v laboratoři.

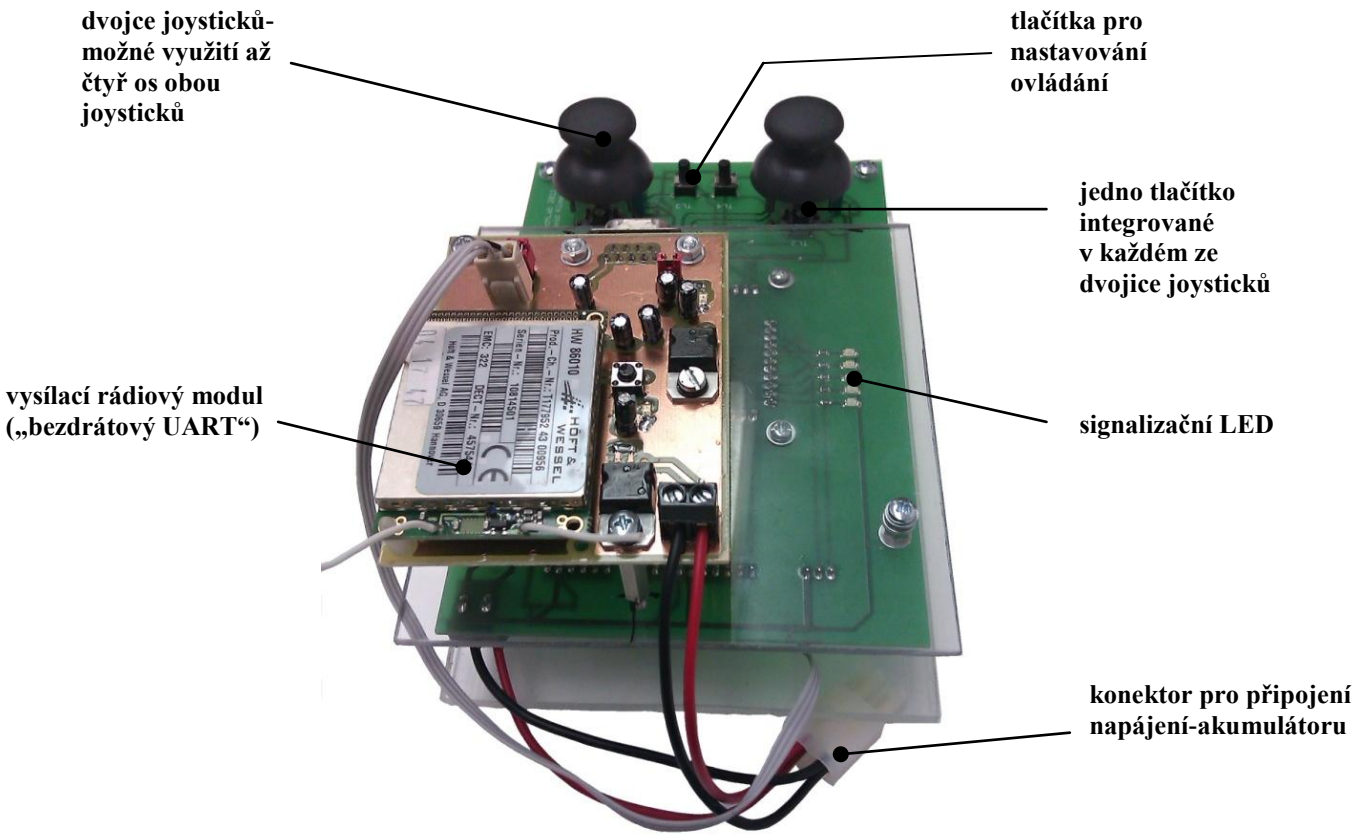
Konektory umístěné na základní desce dálkového ovládání počítají s dalším rozšířením dálkového ovládání, zejména pak např. o displej, který by se vhodně využil při řízení vozidla Car4.

Druhá část dálkového ovládání-programová část byla vytvořena především díky Kerhuel Toolboxu. Použitím tohoto toolboxu v prostředí Simulinku bylo docíleno značného zjednodušení návrhu programu-algoritmu pro dálkové ovládání postaveného na platformě dsPIC. Nastavení a následná implementace použití např. analogového převodníku je pak velmi jednoduché a v celkovém návrhu dálkového ovládání především rychlé. Kompletní model programu dálkového ovládání je přiložen v příloze této práce.

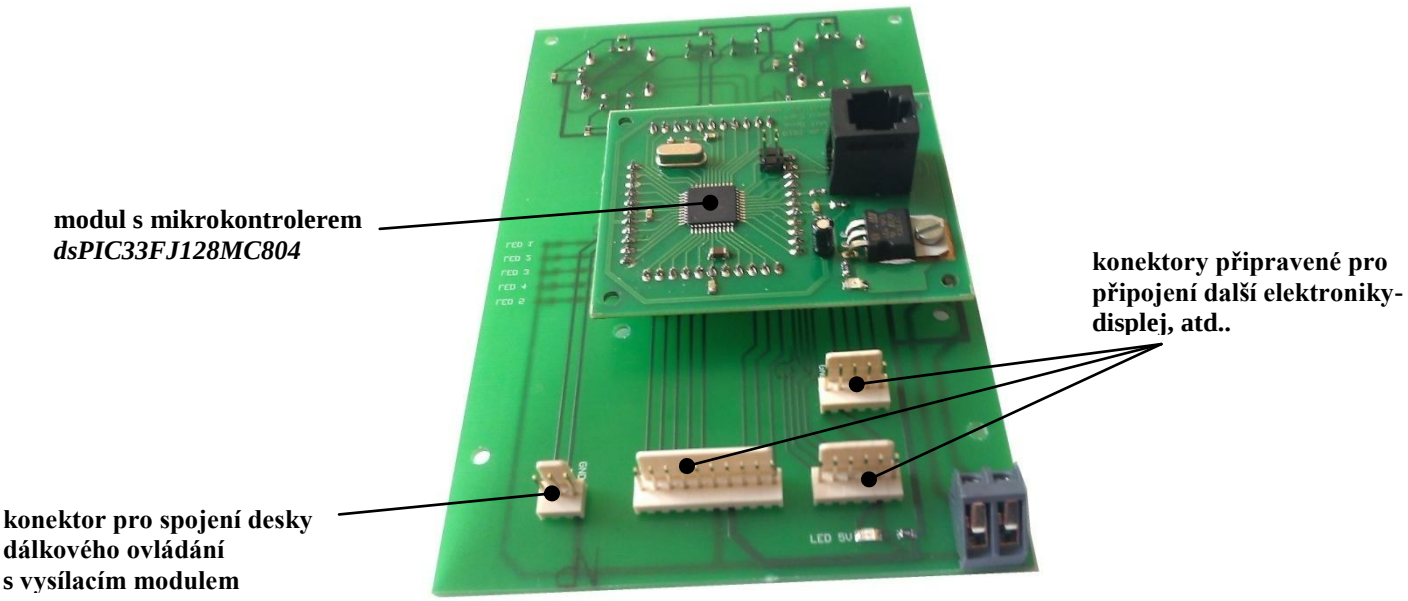
Shrnutí výsledných specifikací dálkového ovládání:
(grafické znázornění je na následující straně)

- byl použit mikrokontroler *dsPIC33FJ128MC804* (modul s tímto mikrokontrolerem)
- plynulé řízení obou náprav a rychlosti zajištěno dvojicí joysticků
- možnost změny módu řízení pomocí tlačítek (řízení pouze jedné nápravy, obou náprav, atd...)
- možnost změny „rychlostního stupně“ pomocí tlačítek integrovaných do joysticků
- zobrazování aktuálního rychlostního stupně na signalizačních LED diodách
- připravené konektory pro připojení displeje a další doplňkové elektroniky
- napájení 6V- zajištěno nikl-metal hydridovým akumulátorem o kapacitě 2700 mAh

[14] *PragoBoard s.r.o.*-výroba plošných spojů (http://www.pragoboard.cz/)



Obr. 6.5. Kompletní dálkové ovládání (deska dálkového ovládání, vysílací rádiový modul, ...) –horní pohled



Obr. 6.6. Samostatná deska dálkového ovládání –dolní pohled

7 Shrnutí dosažených výsledků

V této práci byly splněny všechny cíle, které byly stanoveny v zadání bakalářské práce. Byl nejprve prostudován průběh generování kódu, hlavně v prostředí Simulink. Pomocí Kerhuel Toolboxu byly zjištěny možnosti s použitím Simulinku jako vhodného adepta pro vytvoření modelu a výsledné generování kódu pro cílové zařízení. Byla nastíněna základní funkce nástroje Real-Time Workshop, který spolu s Target Language Compilerem tvoří základ pro generování kódu v jazyce C.

Dále proběhlo seznámení se s projektem experimentálního vozidla Car4, kde bylo nutné se důkladněji seznámit s komunikací po sériové lince - UART. Na základě takto získaných informací bylo možné upravit část kódu pro Car4 pro příjem dat do Car4 (např. z dálkového ovládání).

Hlavní část práce se zabývala ovládáním vozidla Car4. Cílem bylo ovládat vozidlo Car4 pomocí dálkového ovládání a využít přitom mikrokontroler PIC, na kterém by se prakticky aplikovaly poznatky získané kolem problematiky automatického generování kódu. Návrh takovéhoto dálkového ovládání je pouze základní možnost, kdy komunikuje pouze dálkové ovládání s vozidlem Car4 v jednom směru. Opačná komunikace vozidla Car4 s dálkovým ovládáním zatím nebyla využita (i když je k dispozici). Tato možnost je pak připravena na další využití-posílání dat z Car4 např. na displej umístěný na dálkovém ovládání, kde by se zobrazovala např. aktuální rychlost. Návrh s použitím displeje nebyl součástí této práce.

Takto vytvořený základní návrh dálkového ovládání je pak připraven na další zlepšování např. využitím akcelerometru a gyroskopického senzoru, které by sloužili k „zadávání“ požadovaného směru a rychlosti místo zatím použitých joysticků.

Při návrhu programu pro dálkové ovládání byly využity nástroje pro automatické generování kódu. Na základě takto vygenerovaného kódu byl pak naprogramován použitý mikrokontroler-základ dálkového ovládání. Program v mikrokontroleru dálkového ovládání bylo vždy možné rychle a jednoduše upravit podle potřeby pro lepší ovládání vozidla v prostředí Simulink a poté opět pouze nechat vygenerovat nový kód a nahrát takto aktualizovaný program do mikrokontroleru. Tímto způsobem probíhalo testování dálkového ovládání (testování při ovládání vozidla Car4), díky čemuž se rychle a především jednoduše dosáhlo požadovaných vlastností dálkového ovládání, kdy se vozidlo Car4 dalo nakonec jednoduše ovládat. Nástroj pro automatické generování kódu, složený z prostředí Simulink doplněný o Kerhuel Toolbox, byl v tomto případě velkým přínosem pro celkový návrh dálkového ovládání.

8 Literatura a odkazy

[1] MICROCHIP. *16-bit PIC24 MCUs and dsPIC® DSCs*. [online]. 2011-04-20. Dostupné z http://www.microchip.com/en_US/family/16bit/

[2] WIKIPEDIA: *Mikrokontrolér PIC*. [online]. 2011-04-10. Dostupné z http://cs.wikipedia.org/wiki/Mikrokontrolér_PIC

[3] MATH WORKS: *Real-Time Workshop User's Guide.* Version 1.2, 1999.

[4] MATH WORKS: *Target Language Compiler Reference Guide.* Version 3.0, 1999.

[5] VEJLUPEK, J.: *Vývoj elektroniky pro řízení trakce experimentálního vozidla.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. Vedoucí diplomové práce Ing. Robert Grepl, Ph.D.

LAMBERSKÝ, V.: *Vývoj algoritmů pro odhad stavu experimentálního vozidla.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. Vedoucí diplomové práce Ing. Robert Grepl, Ph.D.

ŠIMURDA, M.: *Návrh a realizace doplňkového sensorického systému pro experimentální vozidlo.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. Vedoucí bakalářské práce Ing. Robert Grepl, Ph.D.

JASANSKÝ, M.: *Návrh dynamických modelů pro řízení trakce experimentálního vozidla.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 123 s. Vedoucí diplomové práce Ing.Robert Grepl, Ph.D.

[6] MATHWORKS: *Generate C and C++ code from Simulink and Stateflow models.*[online]. 2011-04-10. Dostupné z http://www.mathworks.com/products/simulink-coder/

[7] KERHUEL L.: *dsPIC Blockset*. [online]. 2011-04-20. Dostupné z http://www.kerhuel.eu/wiki/Simulink_-_Embedded_Target_for_PIC

[8] NATIONA INSTRUMENTS: *LabVIEW C Code Generation Technology Basics*. [online]. 2011-03-15. Dostupné z http://zone.ni.com/devzone/cda/tut/p/id/11784

[9] NATIONA INSTRUMENTS: *Getting Started with the NI LabVIEW C Generator*. [online]. 2011-03-15. Dostupné z http://digital.ni.com/manuals.nsf/websearch/6DD40C65B07DCE3F8625776000685F6C

[10] PELÁNEK R.: *Real Time Systems Introduction.*[online]. 2011-05-13. Dostupné z: http://www.fi.muni.cz/~xpelanek/IA158/slides/intro.pdf