VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH JEDNODESKOVÉHO ŘÍDÍCÍHO SYSTÉMU PRO MODEL VOZIDLA POHYBUJÍCÍHO SE V AUTONOMNÍM KONVOJI

SINGLE BOARD COMPUTER BASED CONTROL DESIGN FOR MODEL OF AUTONOMOUS CONVOY VEHICLE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — DAVID ČERMÁK
AUTHOR

VEDOUCÍ PRÁCE — ING. STANISLAV VĚCHET PH.D.
SUPERVISOR

BRNO 2013

ZADÁNÍ ZÁVĚREČNÉ PRÁCE

ABSTRAKT

Tato bakalářská práce se zabývá návrhem řídícího systému pro pohyb vozidla v autonomním konvoji. Navrhnutý systém je založen na platformě BeagleBoard-xM. Jediná vstupní periferie je USB kamera a řídící požadavky jsou vysílány prostřednictvím sériové komunikace. Ovládání vozidla je zprostředkováno prostřednictvím mikrokontroléru ATmega16 a funkce je otestována ve vývojovém prostředí AVR Studio 4.19.

ABSTRACT

This thesis describes the design of a control system for autonomous vehicle motion in a convoy. The suggested system is based on the platform BeagleBoard-xM. The only input peripheral is an USB camera and the contorol requirements are then transmitted via serial communication. Vehicle handling is mediated by the microcontroller ATmega16 and the functionality is tested in a debug environment of AVR Studio 4.19.

KLÍČOVÁ SLOVA

Autonomní konvoj, BeagleBoard, řídící systém, ATmega16, UART, PWM

KEYWORDS

Autonomous convoy, BeagleBoard, control system, ATmega16, UART, PWM

PODĚKOVÁNÍ

Chtěl bych poděkovat především vedoucímu bakalářské práce Ing. Stanislavu Věchetovi Ph.D. za poskytnuté rady, konzultace, zkušenosti a vstřícný přístup.

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením Ing. Stanislava Věcheta Ph.D. a použil jsem literaturu uvedenou v bibliografii

V Brně, 24.5.2013

.....
David Čermák

BIBLIOGRAFICKÁ CITACE

ČERMÁK, D. Návrh jednodeskového řídicího systému pro model vozidla pohybujícího se v autonomním konvoji. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 59 s. Vedoucí bakalářské práce Ing. Stanislav Věchet, Ph.D..

Obsah

1 ÚVOD

Jednodeskové počítače (single-board computers, SBC) jsou pro své přednosti oproti standardním počítačům velmi rozšířenou součástí řídících systémů zejména v oblasti průmyslových aplikací, ale pro své neustále nově objevované možnosti aplikací třeba i v oblasti medicíny, zábavy a dalších. První část této práce je věnována právě hlavním vlastnostem jednodeskových počítačů, seznámením s aktuálně používanými SBC ve světě a srovnáním jejich parametrů na několika příkladech.

Podrobnější prostudování je pak věnováno jednodeskovému počítači Beagleboard, produktu společnosti Texas Instruments, konkrétně pak jeho modernější verzi BeagleBoard-xM. Větší prostor je tedy věnován nejenom parametrům a vlastnostem tohoto počítače, ale podrobně jsou zde rozebrány jeho komunikační rozhraní s jejich vlastnostmi a možnostmi využití. Na platformě Beagelboard je založen vestavěný systém, který slouží pro řízení modelu vozidla, pohybujícím se v autonomním konvoji. Zatímco vedoucí vozidlo konvoje je řízeno rádiovým vysílačem, kdy jednotlivé příkazy zadává obsluha na základě zrakových vjemů a vlastních rozhodnutí stran rychlosti a směru pohybu vozidla, pohyb dalšího vozidla v konvoji je pak ovládán řídícím systémem, jež je obsahem další části této práce. Zpracování samotného návrhu předchází analýza jak potřebných komponentů a funkcí jednodeskového počítače, tak i elektromotorů, které uvádějí vozidlo do pohybu, resp. mění jeho směr. Navržený řídící systém je následně experimentálně otestován a je prokázána jeho funkčnost.

2 JEDNODESKOVÉ POČÍTAČE

Jednodeskový počítač, také nazýván SBC(Single-Board Computer), je výpočetní stroj, jenž je sestaven na jedné polovodičové desce s plošnými spoji. Oproti standardním osobním počítačům mají tedy menší rozměry, nižší energetickou spotřebu a jsou méně poruchové. Tyto přednosti jednodeskových počítačů jsou blízce spjaté s jejich velikostí. Technologický pokrok umožňuje navyšovat hustotu plošných spojů a díky schopnosti vyrábět polovodičové desky s velkou hustotou plošných spojů se naskytla možnost umístit všechny komponenty potřebné pro funkci osobního počítače na jednu desku plošných spojů. Jednodeskový počítač nevyžaduje pro svou funkci připojení dalších komponentů pomocí sběrnic, které jsou častými zdroji poruch [1]. Díky své kompaktnosti jsou tedy jednodeskové počítače více spolehlivé.

2.1 Využití a výhody jednodeskových počítačů

Na trhu jsou k sehnání mnohé jednodeskové počítače určené pro různé účely. Jejich parametry se přímo odvíjí od předpokládaného využití a uživatel si s výhodou může zvolit celek nejvíce vyhovující jeho aplikaci. Počítačoví nadšenci, kteří nevyužívají tuto výpočetní techniku za marketingovými účely, využívají nejčastěji statickou paměť typu RAM a nízko-rozpočtové osmi nebo šestnácti bitové procesory. I přes poměrně malou cenu disponují jednodeskové počítače při volbě obdobných komponentů dostatečným výpočetním výkonem pro aplikace ve vestavěných systémech a automatizaci [2]. Jednodeskové počítače nemají využití pouze v této oblasti. Pro aplikace vyžadující nadstandardní výpočetní výkon, jako jsou například servery, se vyrábějí jednodeskové počítače disponující srovnatelnou pamětí a výkonem procesoru, jako mají osobní počítače, a navíc v kompaktním provedení s menšími rozměry. Využívání jednodeskových počítačů má tedy mnohé výhody a v určitých případech se i oproti strojům navrhnutým přímo pro danou aplikaci z finančních, či energetických a jiných důvodů vyplatí jejich využití více. V následující tabulce je srovnání výhod a nevýhod systémů obsahujících základní desku nebo jednodeskový počítač.

Tab. 1: Srovnání výhod a nevýhod systémů využívajících pro svou funkci jednodeskový počítač nebo základní desku.

Jednodeskový počítač	Systém se základní deskou [3]
Výhody	
Všechny potřebné komponenty v kompaktním provedení -> výměna v rámci pár minut.	Možnost nenáročného rozšíření o potřebné komponenty pomocí velké škály slotů.
Pro větší výpočetní výkon je možnost zapojení více SBC do společného systému.	
Větší odolnost proti mechanickým závadám.	
Velké množství sběrnic, pro provoz SBC je není nutné využívat.	
Nevýhody	
	Velká časová ztráta při demontáži v průmyslu (30 min až hodiny).
	Náhrada jinou základní deskou přináší často softwarové komplikace.
	Technologie využívající základními deskami se rychle vyvíjí a mění, někdy je tedy při výměně komponentu nutné vyměnit celý systém.
	Samostatně není provozuschopná, nutno zapojit vyžadované komponenty pomocí sběrnic.
	Sběrnice jsou náchylné vůči mechanickému poškození a zároveň nutné pro provoz stroje.

2.2 Rozdělení jednodeskových počítačů

2.2.1 Procesory používané v jednodeskových počítačích

Jedním z hlavních komponentů jednodeskových počítačů je CPU, čili centrální výpočetní jednotka. Vlastnosti CPU vychází z jeho konstrukční architektury a podle toho se dále odvíjí využití celého jednodeskového počítače. Nejvíce využívané jsou CPU, vyrobené s ARM architekturou, jejíž největší předností je relativně nízká spotřeba elektrické energie oproti druhé nejpoužívanější x86 architektuře [4]. Hlavní rozdíl mezi těmito architekturami je ve způsobu, jejímž procesory operují s instrukcemi. ARM architektura navíc nabízí mnoho funkcí pro úsporu energie. CPU s x86 architekturou jsou naopak méně úspornější, ale disponují větším výpočetním výkonem. Jednodeskové počítače s CPU založeném na ARM architektuře se z těchto důvodů s výhodou využívají v mobilních zařízeních a systémech napájených baterií. x86 architektura se hodí do systému napájených z elektrické sítě a jsou schopny plnit náročnější a komplexnější úlohy. Nejčastěji jsou jednodeskové počítače využívány pro aplikace

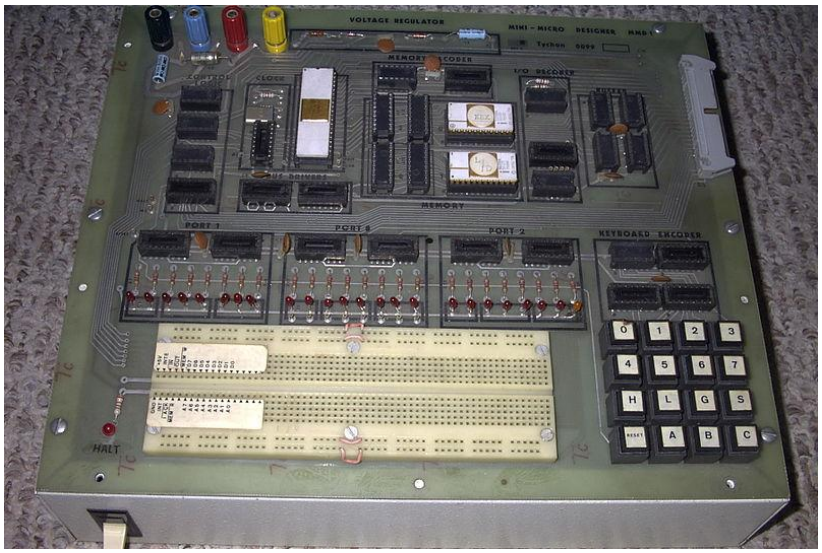
nevyžadující velký výpočetní výkon a je kladen důraz na spotřebu, proto je tedy nejhojněji využívaná ARM architektura, která je pro tyto případy vhodnější.

2.2.2 Jednodeskové počítače se sloty a bez slotů

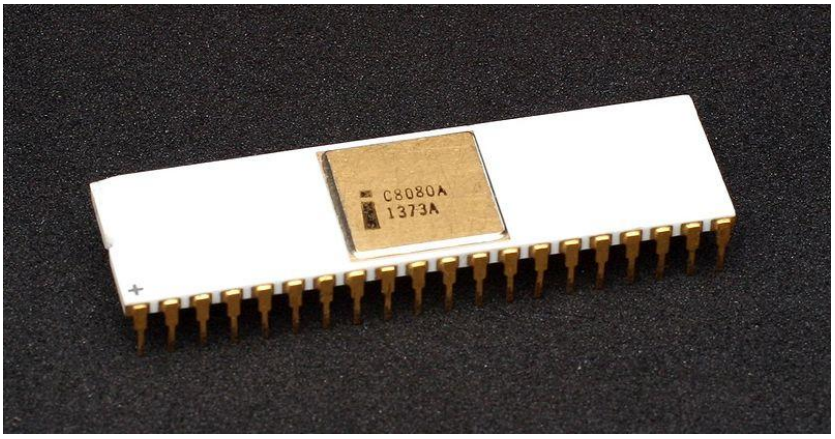
Jednodeskové počítače se dále dělí podle své vlastní konstrukční architektury na jednodeskové počítače se sloty a bez slotů. Architektura se sloty obsahuje stejné vstupní i výstupní rozhraní jako architektura bez slotů, navíc je díky přítomnosti slotů možné zapojit další podporované komponenty. Tato architektura se velmi podobá základním deskám, avšak vyniká menšími rozměry, spotřebou, menším počtem integrovaných prvků a svou hmotností.

2.3 Historie jednodeskových počítačů

Za první jednodeskový počítač se považuje „dyna-micro“ (viz. obr. 1), který byl přepracován společností E&L Instruments of Derby v roce 1976 a nazván „MMD-1“. Jako CPU měl procesor C8080A (viz. obr. 2) od firmy Intel a využíval také jejich první EPROM paměť C1702A. Proslavil se v knize „The 8080A Bugbook“, kde byl označen jako příkladový mikropočítač.



Obr. 1: První jednodeskový počítač Dyna-micro (převzato z [2]).



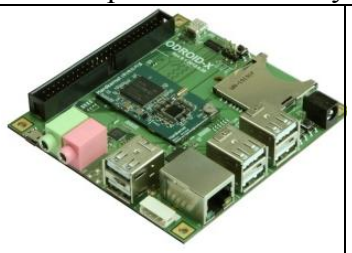
Obr. 2: Procesor C8080A, jenž tvořil CPU prvního jednodeskového počítače dyna-micro (převzato z [2]).

Situace na trhu je v dnešní době podstatně rozdílná. Počítače jsou řadovému uživateli mnohem přístupnější než v dobách prvních jednodeskových počítačů. Díky hromadné výrobě je cena i nejmodernějších jednodeskových počítačů poměrně nižší a výrobci se snaží zpřístupnit využití jejich systému i méně zkušeným zákazníkům pomocí datasheetů, ukázkových příkladů, open-source operačním systémům, uživatelským podporám a diskuzím. Jednodeskové počítače se tedy začaly hojně využívat i v zábavním průmyslu a domácnostech.

2.4 Nejvýkonější používané jednodeskové počítače

Mezi nejvýkonnější jednodeskové počítače dnešní doby patří například[9]: Odroid-X, ARNDALE-5250-A a Cotton Candy. Tento výčet neobsahuje všechny používané jednodeskové počítače, ale slouží pro představu, s přibližně jakými parametry se dnes SBC prodávají. Všechny tyto tři počítače mají velmi výkonné procesory postavené na ARM architektuře a jsou tedy vhodné i pro mobilní využití. Jejich jednotlivé parametry jsou zobrazeny v tabulce: (Tab. 2).

Tab. 2: Srovnání parametrů zvolených jednodeskových počítačů.

			
Název :	Odroid-X	ARNDALE-5250-A	Cotton Candy
Procesor :	Exynos4412 (1.4 GHz quad-core)	Exynos5 (1.7 GHz dual-core, ARM Cortex A15)	Exynos 4210 (1.2 GHz dual-core ARM Cortex A9)
Paměť (RAM) :	1024 MB	2 GB	1024 MB
3D akcelerátor :	Mali-400 Quad Core	Není uveden	Mali-400 Quad Core
Rozměry :	90 x 94 mm	140 x 195 mm	80 x 20 mm
Komunikační rozhraní :	• 6 x USB2.0 • HDMI • 3,5 mm jack na audio vstup i výstup • UART • Rozšiřující port pro LCD/I2C/UART/SPI/ ADC/GPIO rozhraní • SDHC slot a eMMC socket • MIPI-CAM konektor	• 2x USB3.0 • 2x USB2.0 • MicroUSB • HDMI • ITU 601 rozhraní • SATA 1.0/2.0/3.0 • eMMC 4.5 • SDIO 3.0 • UART h-s • SPI h-s • Akcelerátor, gyroskop, kompas • Zvuková karta	• USB 2.0 • MicroUSB • HDMI • Wifi 802.11b/g/n • Bluetooth 2.1 + EDR
Zdroj :	[6]	[7]	[8]

3 JEDNODESKOVÝ POČÍTAČ BEAGLEBOARD-XM

Jednodeskový počítač Beagleboard pochází z dílen firmy Texas Instruments, která na něm spolupracuje se čtvrtým největším distributorem elektronických komponentů, firmou Digi-Key. Počítač byl navržen malou skupinou inženýrů, kteří měli vizi vytvořit jednodeskový počítač hlavně pro výukové účely, ale také aby demonstroval systém „system-on-a-chip“ firmy Texas Instrument OMAP3530, založený na ARM architektuře a dle výrobců je sám o sobě schopný nabídnout „výkon hodný notebooku“. Pro výukové účely je architektura čipu ARM velmi vhodná, protože volně dostupné open-source operační systémy obsahují podporu těchto čipů. Také díky prodeji pod „Creative Commons“ licencí se stal tento jednodeskový počítač velmi oblíbeným prostředkem pro výuku hardwarových a softwarových open-source možností.

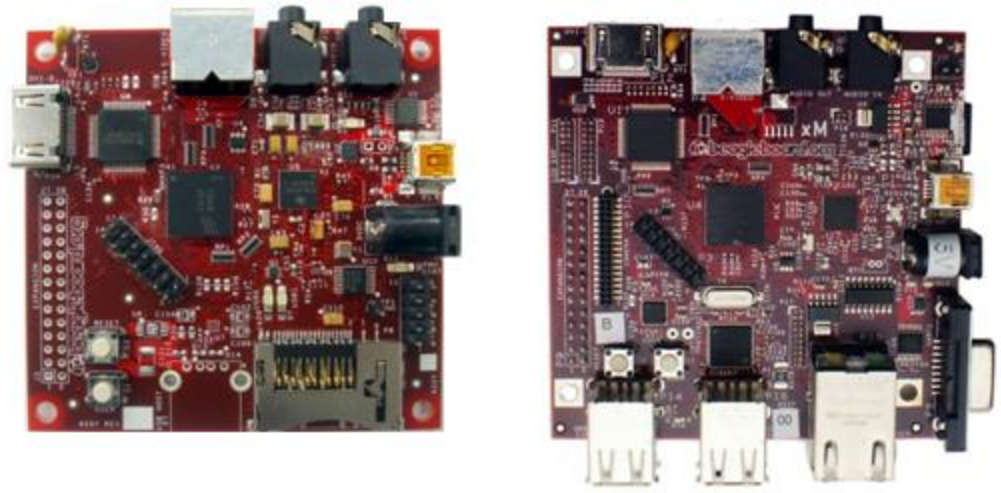
3.1 Srovnání parametrů jednotlivých verzí BeagleBoardů

Beagleboard prošel za dobu své existence značným vývojem z hlediska designu, výkonu, obsahu a pozice periferií. Vyvinuty byly dvě verze. Původní BeagleBoard a modernější BeagleBoard-xM disponující lepším výkonem a jiným hardwarovým rozhraním. Obě verze navíc prošly značným počtem revizí odstraňující některé předešlé chyby. Následující tabulka je vyjmuta s Datasheetu BeagleBoard-xM a vyjadřuje rozdílné parametry těchto počítačů.

Tab. 3: Srovnání parametrů a hardwarového rozhraní vyvinutých verzí BeagleBoardů.

Oblast srovnání	BeagleBoard-xM	BeagleBoard
Procesor	DM3730	OMAP3530
Frekvence jádra ARM	1GHz	720MHz
Frekvence digitálního signálu (DSP)	800MHz	520MHz
Frekvence grafického jádra (SGX)	200MHz	110MHz
Velikost paměti DDR	512MB	256MB
Frekvence paměti DDR	166MHz	166MHz
Velikost NAND paměti	0	256MB
Typ SD konektoru	uSD	MMC/SD
Počet USB Host Portů	4	1
Podpora rychlosti USB Host Portů	FS/LS/HS	HS
Sériový konektor	DB9	Sběrnice
Kamerová sběrnice	✓	×
Dodáváno s 4Gb SD kartou	✓	×
Přepěťová ochrana	✓	×
MMC3 rozšiřující sběrnice	✓	×
McBSP2 rozšiřující sběrnice	✓	×

Z údajů uvedených v tabulce (Tab.3) lze odvodit, že obě platformy jsou osazeny jinými periferiemi. Tato skutečnost je velmi dobře vidět na grafickém srovnání těchto počítačů níže (viz. Obr. 3).



Obr. 3: Grafické srovnání jednodeskového počítače BeagleBoard (vlevo) s novější verzí BeagleBoard-xM (vpravo)(převzato z [11]).

3.2 BeagleBoard-xM

Navržený řídící systém řízení autonomního konvoje obsahuje dle zadání jednodeskový počítač BeagleBoard-xM, a z tohoto důvodu se následující kapitoly týkaící pouze této verze BeagleBoardu. V uvedené tabulce (Tab. 4) jsou shrnuty podrobnější parametry a funkce vestavěných komponentů.

Tab. 4: Shrnutí parametrů a podporovaných funkcí vestavěných komponentů v jednodeskovém počítači BeagleBoard-xM.

Komponent	Parametry a funkce	
Procesor	Cortex A8 1GHz procesor firmy Texas Instruments	
POP paměť	Micron 4Gb MDDR SDRAM (512MB) 200MHz	
PMIC TPS65950	Regulátor napájení	
	Audio CODEC	
	Reset	
	USB OTG PHY	
Podpora ladění	14-pinový JTAG	GPIO piny
	UART	3 LED
Indikátory	Napájení, Chyba napájení	Ovládání uživatelů
	PMU	Napájení USB
Hostitelský USB 2.0 OTG port	Mini AB USB konektor	
	TPS65950 I/F	
USB hostitelské porty	SMSC LAN9514 Ethernet HUB	
	4 FS/LS/HS	Každý Port napájen proudem až 500mA
Ethernet	10/100	Prostřednictvím USB HUBu
Audio konektor	3,5mm L+R výstup	3,5mm L+R vstup
SD/MMC konektor	MicroSD	
Uživatelské rozhraní	Nastavitelné tlačítko	Tlačítko RESET
Video	DVI-D	S-Video
Kamera	Kamerový konektor	Podpora Leopard Imaging Modulu
Napájecí konektor	USB	DC 9V jack
Přepěťová ochrana	Vypnutí při přepětí	
Hlavní rozšiřující sběrnice	Napájení (5V a 1,8V)	UART
	McBSP	McSPI
	I2C	GPIO
	MMC2	PWM
2 LCD konektory	Přístup k ovládání LCD signálů a I2C	3,3V, 5V, 1,8V
Auxiliární Audio	4-pinový konektor	McBSP
Auxiliární rozšíření	MMC3	

3.3 Hardwarové rozhraní BeagleBoard-xM

3.3.1 Procesor

Verze BeagleBoard-xM využívá jako CPU procesor DM3730CBP 1GHz, který je obsažen v takzvaném „Package on Package“ balení. Toto balení obsahuje ve své spodní části (přilehlé k desce) samotné CPU a ve svrchní části jsou obvody pro operační paměť. Z tohoto důvodu nelze na BeagleBoardu-xM najít součástku označenou „DM3730CBP“, ale pouze označení vestavěné operační paměti.

3.3.2 Operační paměť

Operační paměť, obsažená v jednodeskovém počítači BeagleBoard-xM, je k dostání ve dvou různých variantách. Sestava s příponou -00 využívá paměť firmy Micron a sestava s příponou -01 paměť Numonyx. Přes rozdíly ve vnitřní struktuře těchto pamětí mají obě disponovat stejnými parametry 4Gb MDDR SDRAMx32(512MB @166MHz).

Starší verze BeagleBoardu měla navíc k dispozici 256MB NAND paměti. Nová verze BeagleBoard-xM už tuto paměť nemá, avšak s ohledem na možnosti nainstalovaného operačního systému lze následujícími způsoby připojit další non-volatile paměť:

- Připojit uSD kartu do uSD konektoru
- Použít USB OTG port a pomocí napájeného USB hubu připojit USB pevný disk
- Nainstalovat pevný disk na jeden z USB portů
- Zapojit USB adaptér do pevného disku a připojit ho do USB portu

3.3.3 Správa napájení

BeagleBoard-xM rev.C obsahuje dva 3.3V regulátory, jenž slouží k napájení DVI-D enkodéru, RS232 sériové linky a USB hubu. Pro ostatní zařízení a případy, jako jsou napájení Stereo Audio výstupu a vstupu, fyzické vrstvy USB OTG, LED indikátorů a napájení při resetování je vestavěn integrovaný obvod TPS65950, který slouží k těmto zmíněným účelům.

Samotný BeagleBoard-xM může být napájený přivedením stejnosměrného napětí 5V na jack konektor označený „5V“ nebo také přes USB OTG port pomocí USB kabelu.

3.3.4 Uživatelská tlačítka

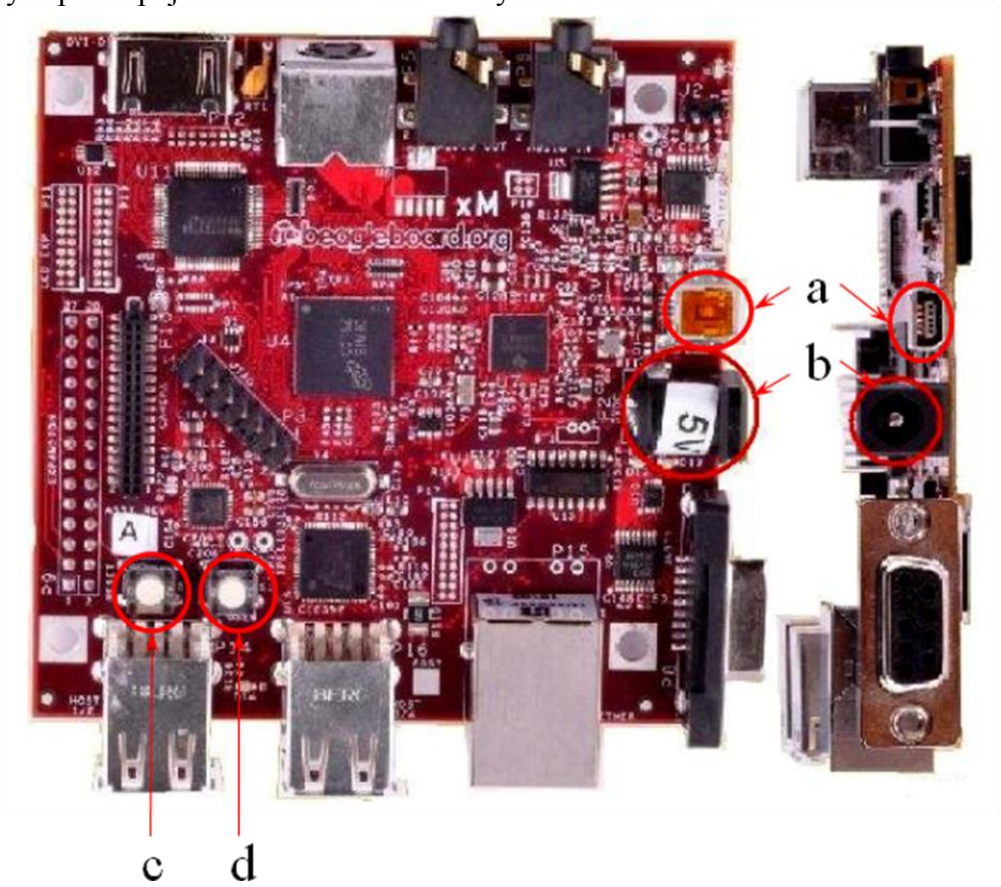
Tlačítko Reset

Když je tlačítko Reset zmáčknuto a posléze uvolněno, dojde k přerušení napájení jednodeskového počítače, což má za následek jeho opětovné spuštění.

Tlačítko User

U dřívějších verzí bylo toto tlačítko využíváno pro vynucené bootování z SD karty místo z NAND paměti. Verze BeagleBoardu-xM již paměť typu NAND neobsahuje a tedy se bootování provádí přednostně z uSD karty. Toto tlačítko může být u nové verze použito bootovacím softwarem na změnu mezi uživatelskými skripty a zavedením jiné bootovací konfigurace. Pokud je tlačítko zmáčknuto při zapnutí jednodeskového počítače, tak nedojde k nabootování.

Na následujícím obrázku (Obr. 3) jsou zvýrazněny pozice umístění konektorů určených pro napájení a umístění uživatelských tlačítek.



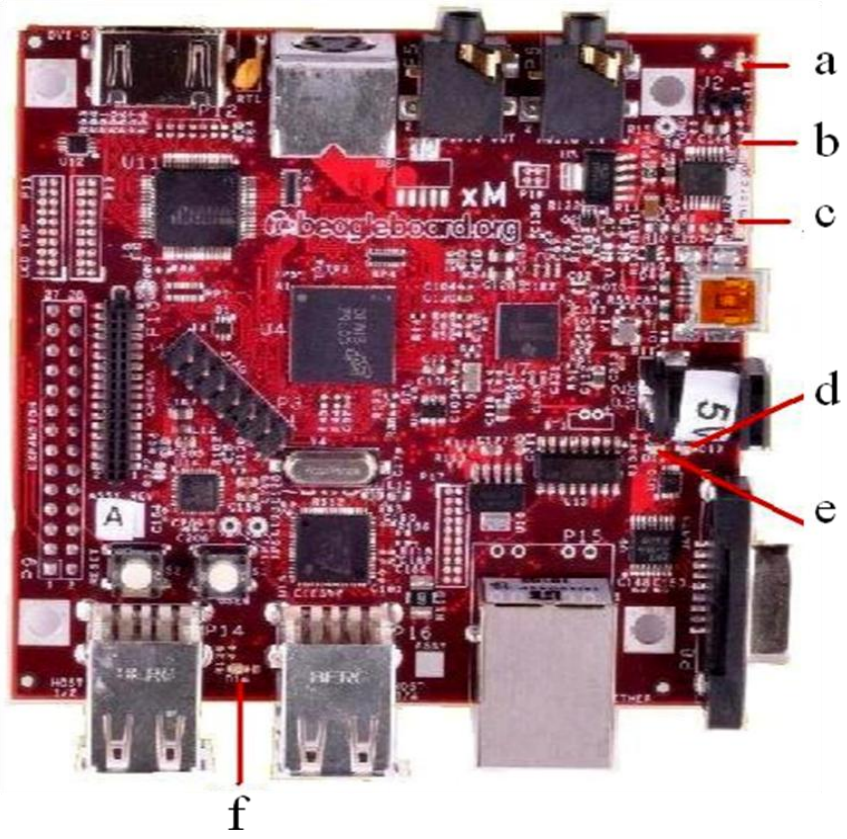
Obr. 3: a) USB OTG, b) 5V jack, c) tlačítko RESET, d) tlačítko USER (převzato z [11]).

3.3.5 Indikátory

BeagleBoard-xM je osazen pěti zelenými indikačními LED diodami, které uživatele informují o následujících stavech :

- Probíhá programování pomocí I2C rozhraní
- Programovatelné stavy řízené GPIO piny, které se indikují dvěma LED diodami
- Indikace přítomnosti správného napájení desky, tato indikace je možná vypnout pomocí softwaru
- Indikace napájení USB HUBu, která jde pomocí software řídit

Dále je přítomna jedna červená indikační dioda, která uživatele informuje o přesáhnutí napájecího napětí. Na následujícím obrázku jsou označeny pozice jednotlivých indikačních diod.



Obr. 4: a) indikace správného napájení, b,c) indikace programovatelných stavů, d) indikace přepětí, e) dioda ovládaná čipem pro správu napájení, f) indikace napájení USB HUBu (převzato z [11]).

3.3.6 Konstrukční specifikace

Následující tabulka (Tab. 5) shrnuje fyzické parametry jednodeskového počítače BeagleBoard-xM:

Tab. 5: Shrnutí fyzických parametrů BeagleBoardu-xM.

Velikost:	85 x 85 mm
Tloušťka desky plošných spojů:	1,5 mm
Počet vrstev desky:	6
Vyhovuje RoHS:	Ano
Váha:	~37 g

3.4 Komunikační rozhraní BeagleBoard-xM

3.4.1 USB 2.0 hostitelské porty

Deska je vybavena čtyřmi USB 2.0 porty typu A, které podporují LS/FS/HS standardy. Tyto porty nemohou být samostatně napájeny přes OTG USB port, ale pokud je napájení stejnosměrným proudem pomocí 5V jack konektoru o velikosti nejméně 3A, tak jsou porty v USB HUBu napájeny pěti volty o napětí 500mA.

3.4.2 Audio konektory

Výstupní Stereo Audio konektor

Výstup audio signálu je vyveden pomocí standardního 3.5mm audio jacku. Výstupní signál obstarává vestavěný audio CODEC, který se fyzicky nachází v integrovaném obvodu TPS65950.

Vstupní Stereo Audio konektor

Vstup pro vstupní audio signál je také 3.5mm audio jack a ten je dále zpracován vestavěným CODECem.

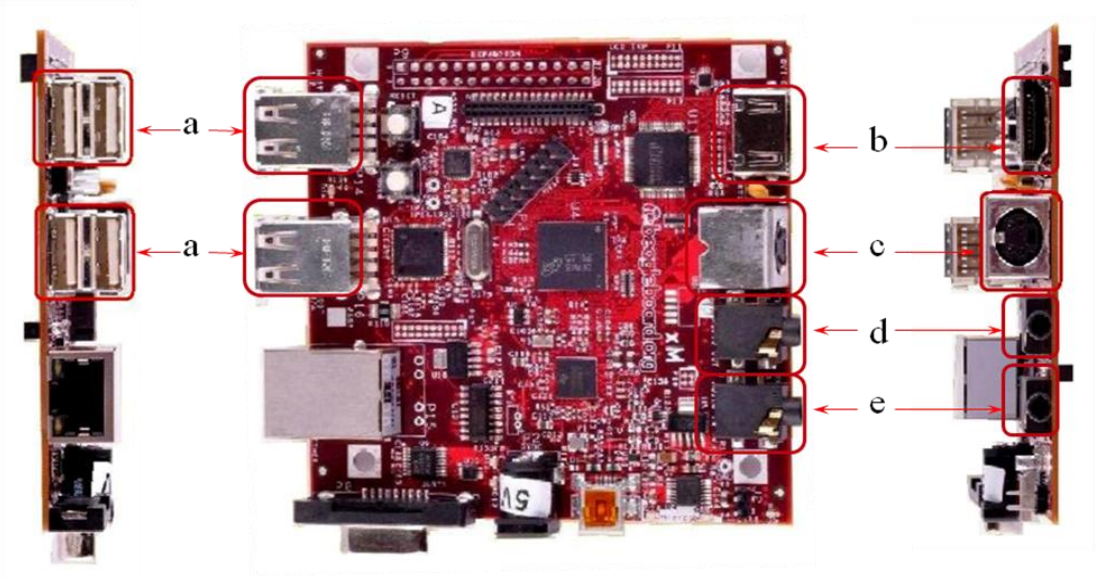
3.4.3 HDMI konektor

Tento konektor slouží k připojení monitoru, jenž má DVI-D rozhraní a procesor je schopen přes něj vysílat video signál s barevnou hloubkou 24b. Pro svou malou velikost byl na desku osazen HDMI konektor, avšak tento výstup je vybaven DVI-D rozhraním. Pro připojení DVI-D kabelu do tohoto konektoru je nutná redukce HDMI na DVI-D. Pro správnou funkci při připojení přímo HDMI kabelu není nutná žádná redukce. Podpora DDC2B nebo EDID je v BeagleBoardu-xM zprostředkovávána pomocí integrovaného obvodu I2C a slouží k identifikaci typu připojeného monitoru a požadovaných nastavení.

3.4.4 S-Video konektor

Tento konektor slouží k přístupu k S-Video signálu BeagleBoardu. Výstupy z procesoru pro HDMI a S-Video konektory jsou rozdílné a pomocí softwaru se dá na obou výstupech generovat jiný signál. BeagleBoard podporuje formáty NTSC a PAL, z nichž je formát NTSC přednastaven jako výchozí ale toto nastavení se dá změnit pomocí softwaru.

Na následujícím obrázku (Obr. 5) jsou zvýrazněny výše zmíněné komunikační rozhraní.



Obr. 5: a) USB hostitelské porty, b) HDMI konektor, c) S-Video konektor, d) výstupní audio konektor, e)vstupní audio konektor (převzato z [11]).

3.4.5 LCD sběrnice

Na desce se nachází dvojice sběrnic 2x10, které slouží k přístupu ke generovaným LCD signálům.

3.4.6 microSD konektor

Tento konektor slouží pro připojení paměťové karty. Oproti minulé verzi BeagleBoardu byl nahrazen 6 in 2 SD/MMC konektor tímto uSD konektorem. Tento paměťový člen tvoří hlavní non-volatile paměť BeagleBoardu a standardně je z něj spouštěn Bootloader s operačním systémem.

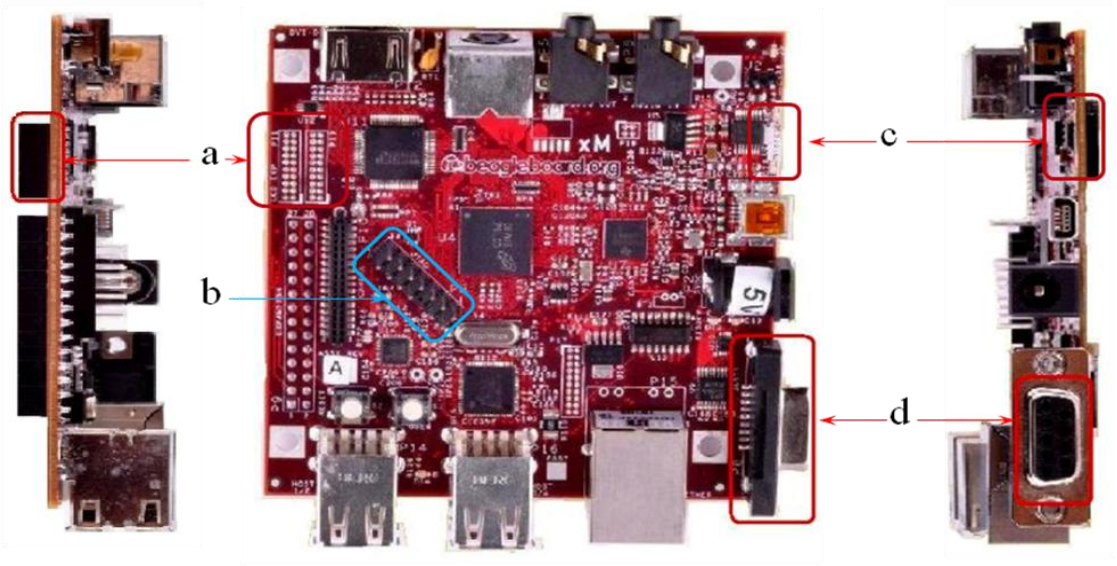
3.4.7 JTAG konektor

Tento 14-ti pinový konektor slouží k vývoji a ladění softwaru BeagleBoardu při využití různých JTAG emulátorů. Tento konektor podporuje pouze signály s úrovní 1,8V.

3.4.8 RS232 DB9 konektor

Podpora sériové komunikace pro RS232 pomocí UART3 je vyvedena prostřednictvím DB9 konektoru. Pomocí DB9 konektoru se tedy přistupuje k vestavěnému RS232 vysílači. S využitím redukce USB na sériový DB9 kabel lze do tohoto konektoru připojit přímo kabel USB bez použití dalších modulů.

Na následujícím obrázku (Obr. 6) jsou zvýrazněny některé z výše zmíněných komunikačních rozhraní.



Obr. 6: a) LCD sběrnice, b) JTAG sběrnice, c) microSD konektor, d) RS232 DB9 konektor (převzato z [11]).

3.4.9 Hlavní rozšiřující sběrnice

Tato sběrnice sestává z 28 pinů a umožňuje uživateli připojení libovolné rozšiřující karty. Díky multiplexování výstupů této sběrnice lze na každém výstupu generovat jiný signál.

3.4.10Konektor kamery

Kvůli podpoře kamerových modulů byl přidán na BeagleBoard-xM konektor pro kamery. Mezi podporované kamerové moduly patří VGA, 2MP, 3MP a 5MP.
MMC rozšiřující sběrnice

3.4.11MMC rozšiřující sběrnice

Tento 20-ti pinový konektor umožňuje přístup k dalším signálům obsahující GPIO a MMC3 port. Tato sběrnice je u BeagleBoard-xM novinkou.
McBSP rozšiřující sběrnice

3.4.12McBSP rozšiřující sběrnice

Tento konektor sestává ze čtyř pinů a slouží k přístupu k McBSP2 signálům pro zvukové aplikace. Pokud chce uživatel využívat tyto signály, je nejprve nutné softwarově zakázat zvukové rozhraní TPS65950.

4 NÁVRH STRUKTURY ŘÍDÍCÍHO SYSTÉMU

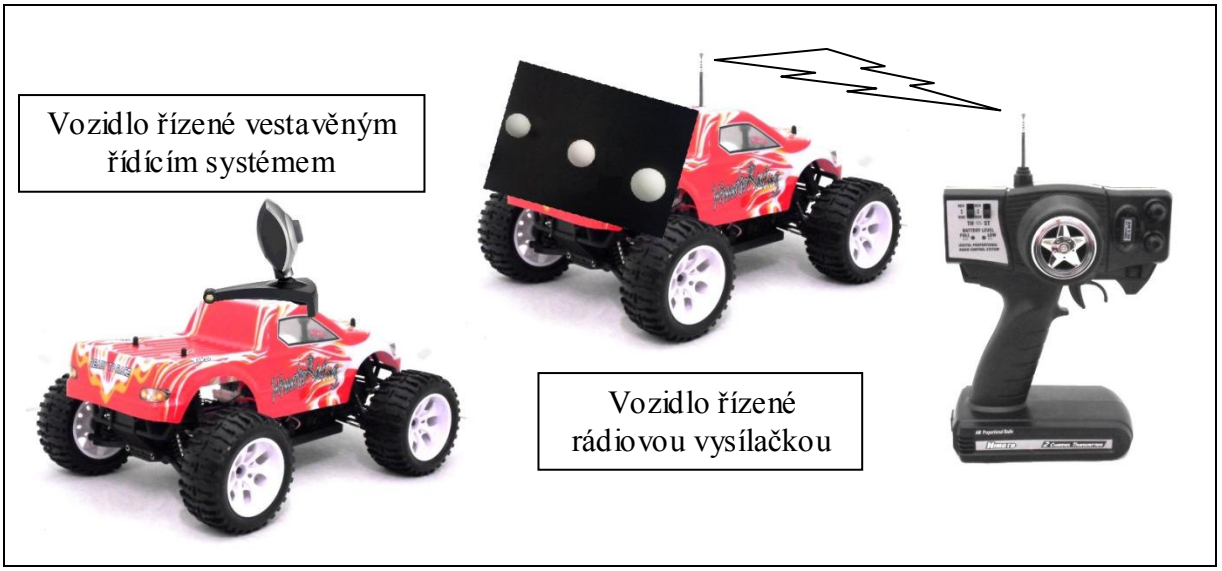
Dle zadání je vyžadováno, aby druhé vozidlo pohybující se v autonomním konvoji bylo řízeno navrhnutým řídícím systémem a ovládáno pouze pomocí vestavěného systému. V naší úloze byl jako řídící systém zvolen jednodeskový počítač BeagleBoard-xM. Funkce tohoto řídicího systému bude hlavně regulační a plně automatizovaná.

4.1 Periferie řídícího systému

Navrhnutý řídící systém vyžaduje pro plnění své funkce vstupní i výstupní periferie, protože nemá vestavěny prostředky pro vyžadované úlohy senzoriky a ovládání dvou rozdílných elektromotorů.

4.1.1 Vstupní periferie

Jediná použitá vstupní periferie je USB kamera. Propojení USB webkamery s BeagleBoardem-xM je pomocí USB kabelu, který je zapojen do jednoho z USB hostitelských portů. Tato webkamera má za úkol snímat prostor před vozidlem s vestavěným řídícím systémem a tyto data dále BeagleBoard zpracovává pomocí aplikace, pracující pod operačním systémem Linux UBUNTU, jež byl zvolen jako vhodný operační systém podporovaný BeagleBoardem-xM a je vhodný pro naší aplikaci. Navržený program využívá ke zpracování obrazu funkce OCR a je nastaven tak, aby ve snímaném obrazu rozpoznával barevné přechody bílá/černá ve tvaru kružnice. Na manuálně řízeném automobilu v autonomním konvoji je na zadní straně umístěna černá matná deska, na které jsou symetricky umístěny stejné bílé míčky. Vyobrazení způsobu řízení vozidel v konvoji je na obrázku Obr. 7. S využitím goniometrických funkcí je dále aplikace vyhodnocující polohu prvního vozidla naprogramována tak, aby ze snímaných rozměrů a polohy míčků vypočítala vzájemnou vzdálenost vozidel v konvoji a relativní orientaci prvního vozidla k vozidlu druhému. Aplikace na základě těchto údajů vyhodnotí, jakým má automobil řízený řídícím systémem jet směrem a rychlostí.



Obr. 7: Grafické znázornění způsobu ovládání vozidel v autonomním konvoji.

4.1.2 Výstupní periferie

Výstupní periferie sestává ze systému původně vestavěných hardwarových prvků automobilu a přidaných komponentů, nutných pro regulovatelné řízení jeho elektromotorů řídícím systémem. Jako prvek ovládající servomotor pro zatáčení a hnací elektromotor byl zvolen mikrokontrolér ATmega16 od firmy Atmel. Tento mikrokontrolér je schopen sériové komunikace přes UART rozhraní a jeho prostřednictvím lze generovat příslušné PWM signály, kterými se ovládají elektromotory vestavěné v automobilu. Na softwarové vrstvě BeagleBoardu je tedy nutné vytvořit aplikaci, která bude přes sériovou linku vysílat regulační požadavky do komponentu ovládajícího elektromotory. Ovládací prvek ATmega16 obsahuje PORTy pro sériovou komunikaci prostřednictvím UART komunikace. BeagleBoard vysílá regulační požadavky pomocí USB kabelu, zapojeného do jednoho z hostitelských USB PORTů. Pro převod UART signálu z USB na UART rozhraní je potřeba převodník. Jako převodník USB/UART byla použita polovodičová deska s plošnými spoji a její výpočetní člen je logický obvod FT232RE. Takto použitá sériová komunikace umožňuje fullduplexní komunikaci, avšak řešení regulace vyžaduje pouze komunikaci BeagleBoard -> ATmega16.

Použitý automobil z výroby obsahuje dvoukanálový rádiový přijímač Himoto 2CH RECEIVER, který přijímá řídící požadavky vysílané ovládací vysílačkou. Přijímač předá tyto požadavky zařízení Himoto 3 Way Electronic Speed Controlleru, který na jejich základě generuje PWM signály pro příslušné servomotory. Ty jsou zapojené přímo na výstupy tohoto controlleru. Controller je napájen přímo z baterie automobilu Dualsky X power 4000-2s disponující napětím 7,2V a z něj jsou dále napájeny ostatní komponenty automobilu. Tímto způsobem je řízen první automobil v konvoji. U automobilu řízeného vestavěným řídícím systémem je třeba odpojit tyto členy a nahradit je navrhnutým řídícím systémem. Komponenty řídícího systému je potřeba napájet požadovaným napětím z baterie automobilu, proto je třeba vytvořit logický obvod pro převod hodnoty napětí baterie ze 7,2V na 5V pro BeagleBoard [11]. Pokud Je

BeagleBoard-xM napájen prostřednictvím 5V napájecího jacku, tak lze napájet ostatní zařízení připojená do jeho USB HUBu prostřednictvím jednotlivých USB PORTů.

4.2 Princip ovládání vestavěných elektromotorů

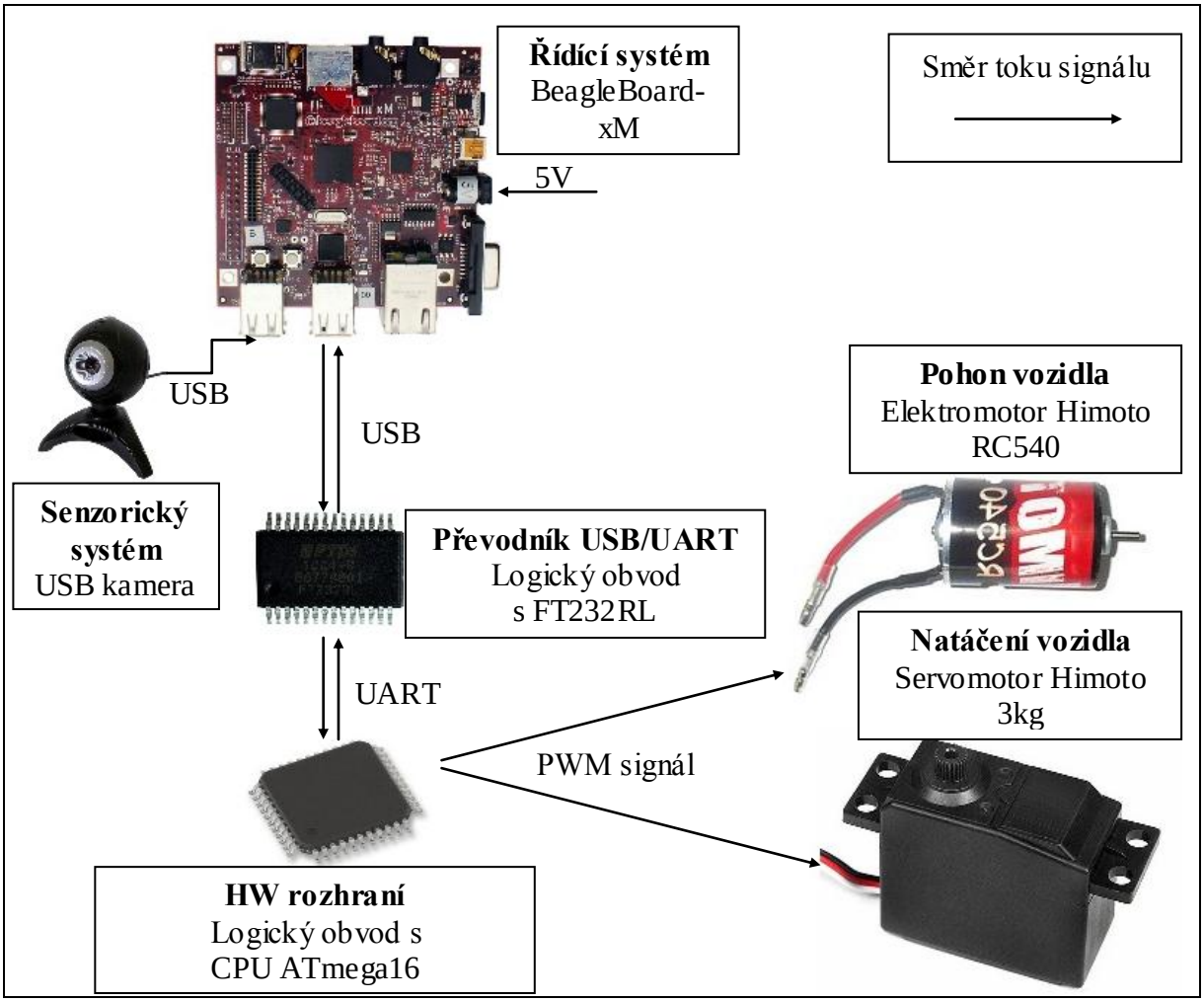
Mikrokontrolér ATmega16 je programovatelný a má vestavěnou hardwarovou funkci pro generování jednoho PWM signálu. Automobil obsahuje dva různé elektromotory a každý z nich je ovládán signálem o jiné vlnové délce. Z tohoto důvodu je potřeba generovat dva PWM signály o rozdílné vlnové délce. PWM signál lze generovat i pomocí softwaru a to na kterémkoliv z PORTů ATmegy16.

Pro přesné a bezpečné ovládání servomotorů je nutné znát jejich parametry. Následující tabulka (Tab. 6) shrnuje parametry a funkce použitých motorů.

Tab. 6: Tabulka obsahující funkce elektromotorů a parametry pro jejich ovládání, čerpáno z [12,13].

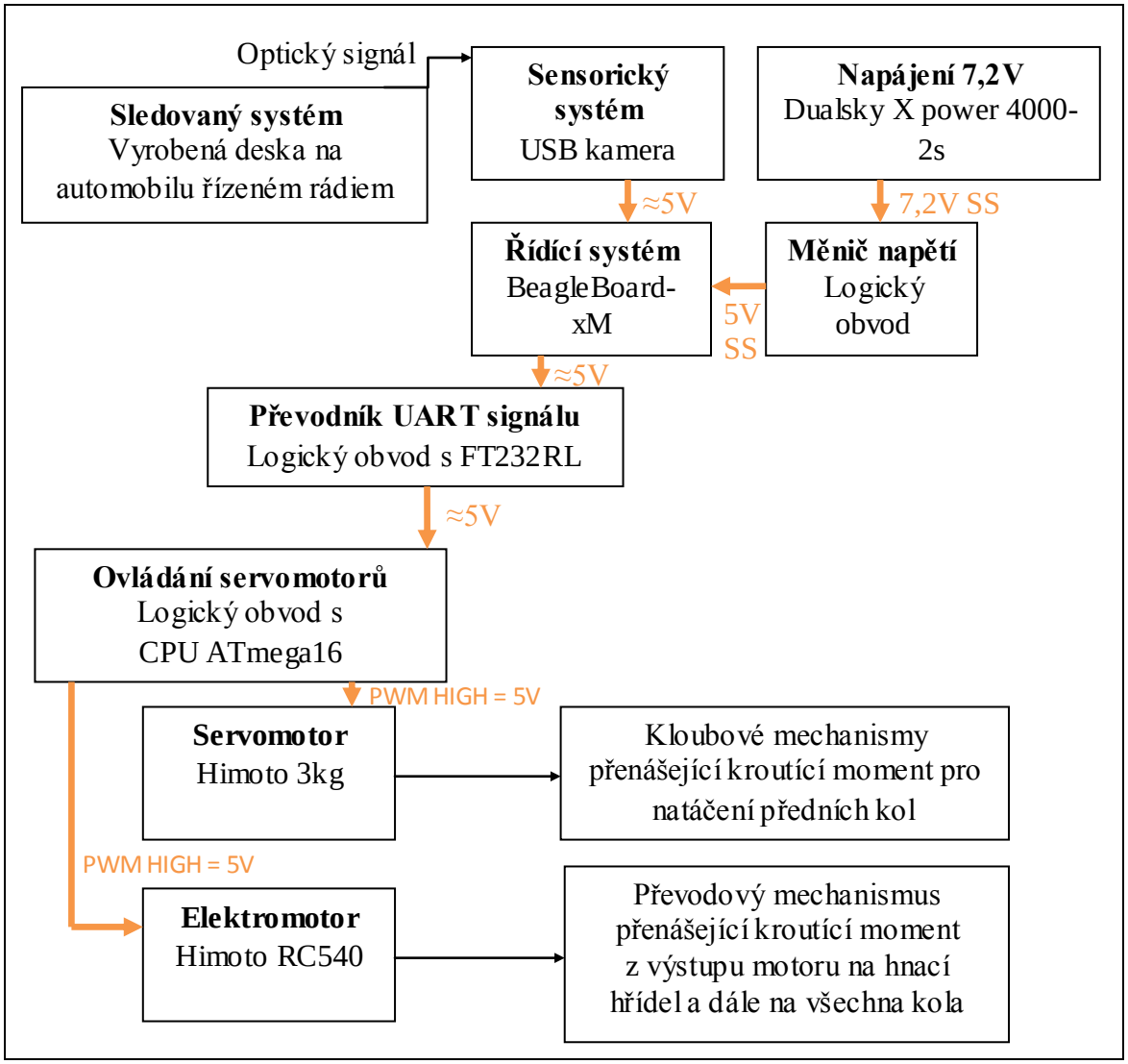
Název motoru:	Himoto RC540	Himoto 3kg servomotor
Typ motoru:	Elektromotor	Servomotor
Funkce motoru v řešeném systému:	Slouží k vytvoření kroutícího momentu na hnací hřídeli prostřednictvím vestavěných mechanismů.	Slouží k natáčení kol prostřednictvím vestavěných mechanismů.
Perioda signálu:	20ms	50ms
Šířka jednoho pulzu:	Běh vpřed: • Začíná vibrovat:1586µs • Plynulý pohyb:1610µs • Plná rychlost:2120µs Běh vzad: • Začíná vibrovat:1428µs • Plynulý pohyb:1422µs • Plná rychlost:875µs	• Plný náklon doleva:2100µs • Střední poloha:1500µs • Plný náklon doprava:876µs

Následující obrázek (Obr. 8) obsahuje schéma shrnující komponenty a rozhraní vyžadující pro ovládání motorů řídícím systémem.



Obr. 8: Schéma znázorňující komponenty řídícího systému a ovládané elektromotory. Šipky představují tok signálu, vedle šipek je napsané rozhraní, přes které probíhá přenos.

Následující obrázek (Obr. 9) znázorňuje blokové schéma vozidla ovládaného řídícím systémem.



Obr. 9: Blokové schéma se všemi prvky systému od sledovaného systému až po generování kroutícího momentu v mechanismu pro otáčky hnacích kol a natáčení předních kol. Schéma znázorňuje směr toku signálu mezi komponenty, jejich napěťové úrovně a druhy signálů.

4.3 Tvorba software pro navrhnutý řídící systém

Navrhnutý systém vyžaduje přípravu software pro dvě zařízení. Dle zadání je potřeba navrhnout aplikaci na softwarové vrstvě pro jednodeskový počítač BeagleBoard-xM, která bude zajišťovat generování UART signálu přes USB PORT. Dále je potřeba navrhnout software pro mikrokontrolér (MCU) ATmega16, pomocí kterého bude MCU přijímat regulační požadavky vysílané s jednodeskového počítače BeagleBoard-xM prostřednictvím UART rozhraní a zároveň na dvou PORTech generovat potřebné PWM signály pro elektromotory dle přijímaných regulačních požadavků.

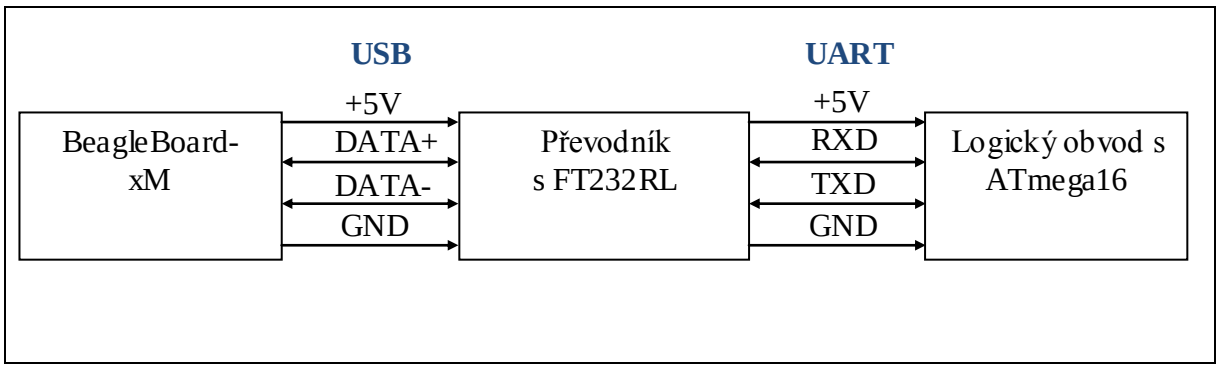
4.3.1 Realizace odchozí UART komunikace z BeagleBoardu-xM

Jednodeskový počítač BeagleBoard-xM potřebuje pro svou funkci vhodný operační systém. V této aplikaci byl jako operační systém zvolen Linux Ubuntu. Jedná se o open-source software, který je k dispozici zdarma a procesor BeagleBoardu-xM Texas Instruments Cortex A8 bylo použito s účelem podpory open-source systémů, jako jsou například Linux, Android, Symbian a další. Pro operační systém Linux Ubuntu jsou k dispozici všechny potřebné ovladače pro správu periferií podporovaných BeagleBoardem-xM. Naše aplikace vyžaduje, aby řídící systém vyslal v případě potřeby regulační požadavek ve zvoleném formátu.

Před zapojením USB UART převodníku do BeagleBoardu je vhodné odinstalovat rušící moduly Linuxu, které by mohly zapříčinit nechtěné funkce programu. Balíček linuxu „brltty“ je určený pro přístup nevidomých ke konzoli, prostřednictvím měkkého braillova displaye. Tento modul by mohl způsobovat problémy a pro jeho odinstalaci použijeme konzolový příkaz [14]:

```
sudo apt-get remove brltty
```

Po zapojení USB UART převodníku do BeagleBoardu a správném propojení s PINy ATmegy16 je řídící systém schopen komunikace prostřednictvím UART rozhraní. Tímto rozhraním je z BeagleBoardu-xM napájena ATmega16 s převodníkem pěti volty. Schéma tohoto propojení s přenášenými signály je znázorněno v obrázku: (Obr. 10).



Obr. 10: Schéma propojení BeagleBoardu-xM s Atmegou16 prostřednictvím UART rozhraní. Ve schématu jsou vyznačeny názvy PORTů a směry toku signálů.

K navázání spojení BeagleBoard-xM -> ATmega16 je nejprve nutné odeslat příkaz shellu, určený k navázání spojení se zvolenou baudovou frekvencí spojení v bitech za sekundu při zvoleném počtu datových a stop bitů. Příkaz také určuje prostřednictvím kterého konektoru bude navázáno spojení. Dle zvolených parametrů má následující tvar:

```
screen /dev/ttyUSB0 76800 7N2
```

Příkaz „screen“ otevře nové okno pro žádanou komunikaci, „/dev/ttyUSB0“ určuje místo, kam je připojeno USB, prostřednictvím kterého je přenášen signál UART komunikace, „76800“ je baudová frekvence přenosu a „7N2“ znamená, že přenos

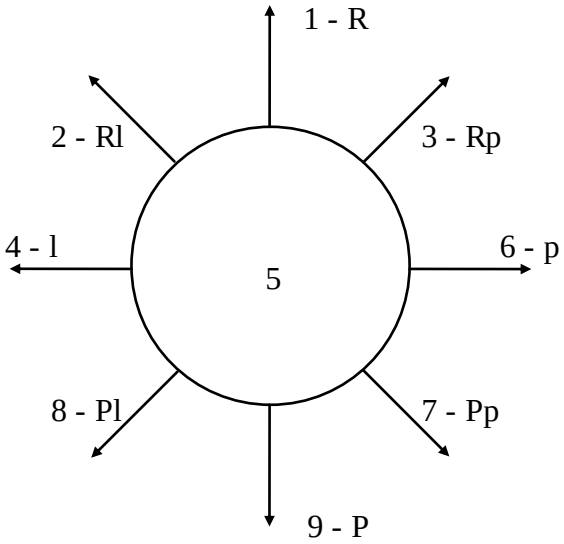
jednoho paketu probíhá po sedmi datových bitech a následují dva stop bity. Použitá baudová frekvence je převzata ze seznamu používaných baudových frekvencí [10], z důvodu omezené možnosti výběru komunikační frekvence v operačním systému Linux Ubuntu. Pokud není zahlášena chyba spojení, tak lze z tohoto shellu posílat textové řetězce do ATmegy přímo zapsáním řetězce do konzole. Z tohoto shellu lze vykonávat také další příkazy a zachovat komunikační spojení. Když řídící systém vyhodnotí potřebu změny rychlosti nebo natočení autonomního vozidla, tak odešle regulační požadavek do zmíněného shellu a ten ho vyšle prostřednictvím UART po sériové lince. Signál dále zpracuje ATmega16, vyhodnotí regulační požadavek a upraví způsob generování PWM signálu dle potřeby regulace.

4.3.2 Princip regulačních příkazů

Pro příkaz regulace rychlosti a natočení automobilu je potřeba vyslat pouze jeden znak typu integer, jenž nese informaci o požadované změně ovládání. Smysl jednotlivých regulačních příkazů je uveden v následujícím obrázku (Obr. 11):

Smysl prvků regulačních kombinací
R – zvýšit rychlost
P – snížit rychlost
l – natoč více doleva
p – natoč více doprava
1 až 9 – vysílaný znak

Přístup k požadavku:
≤3 … R
≥7 … P
{3;6;7} … p
{2;4;8} … l
5 … předchozí stav

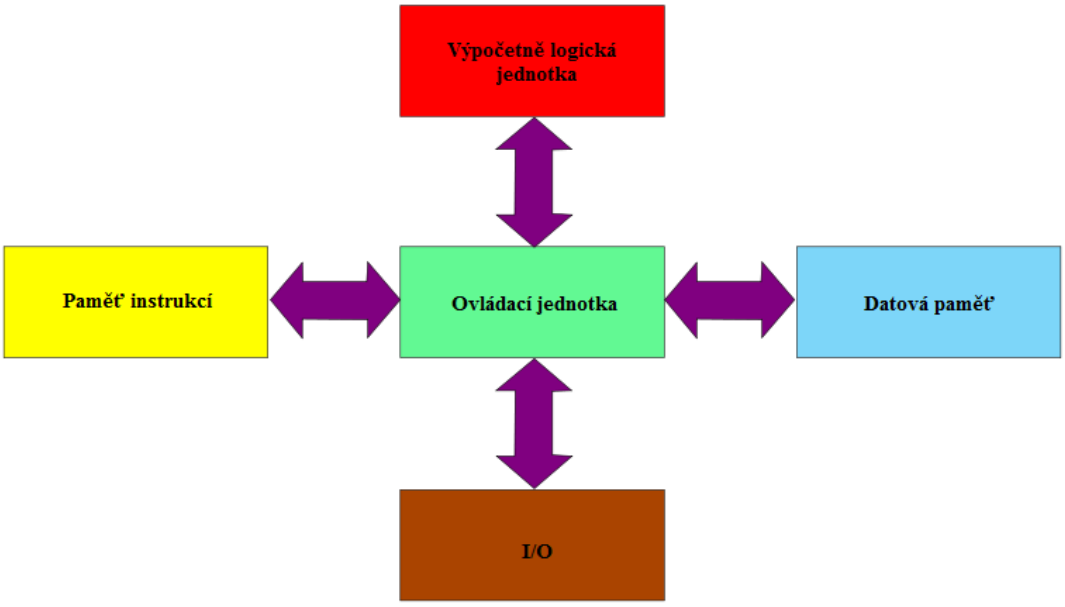


Obr. 11: Obrázek obsahuje diagram znázorňující všechny možné kombinace regulačních požadavků. Šipky představují požadovanou změnu rychlosti nebo natočení v daném smyslu. Uvedená čísla jsou regulační informace vysílané prostřednictvím UART. V obrázku je také napsán přístup ke každé přijaté regulační konstantě.

Z diagramu (Obr. 11) vyplývá, že může nastat 8 případů regulačních požadavků a stav, kdy má automobil udržovat svou rychlost a natočení kol. ATmega16 tedy přijme jeden znak a podle jeho hodnoty se změní přístup ke generování PWM signálu.

Funkce MCU ATmegy16, jenž má v navrhnutém systému funkci příjmu regulačních požadavků a na základě jejich hodnoty generování potřebných PWM signálů pro ovládání elektromotorů automobilu, je založena na Harvardské architektuře

a pro svou funkci nevyžaduje instalaci operačního systému.[15] Následující obrázek (Obr. 12) znázorňuje schéma s komponenty nutnými pro funkci MCU s Harvardskou architekturou.



Obr. 12: Schéma Harvardské architektury (převzato z [15]).

Pro plnění požadované funkce ATmegy16 je potřeba nahrát do datové paměti program, který bude MCU vykonávat. Jak bylo dříve zmíněno tento program má za úkol přijímat regulační požadavky a podle jejich hodnoty generovat příslušné PWM signály ovládající elektromotory. Regulace PWM musí být nastavena tak, aby po příjmu jakéhokoliv regulačního požadavku nenastala velká skoková změna otáček elektromotoru nebo skokové natočení servomotoru, jenž by mohlo způsobit poškození nebo dokonce zničení elektromotorů.

4.3.3 Ověření vhodnosti baudové frekvence

Pro příjem vysílaných informací prostřednictvím UART rozhraní, přesněji tedy RXD PINu MCU ATmega16 je nutné dodržet následující podmínky [17]:

- Zařízení spolu musí komunikovat na stejné baudové frekvenci s maximální možnou chybou dle pravidla bezporuchového přenosu
- Data musí být zpracována s ohledem na formát, ve kterém byla vyslána, to znamená, že přijímací zařízení musí mít nastaveno kolik bitů jednoho slova signálu tvoří start bity, stop bity, paritní bity a datové bity, aby bylo možné signál správně dešifrovat

Velikost baudové frekvence je závislá na frekvenci hodinových pulzů vysílanými krystalem do mikrokontroléru. Frekvence hodinových pulzů je dána použitým krystalem v logickém obvodu s MCU, který mu vysílá pulzy s velmi přesnou

a stabilní frekvencí [18]. Použitý krystal vysílá pulzy s frekvencí 16MHz a zvolená požadovaná baudová frekvence je 76800 bitů za sekundu. ATmega16 tedy přijímá na RXD na zvolené baudové frekvenci, pokud nastavíme registr UBRR na „n“ dle rovnice (1) převzaty z [19]:

$$n = (\,frekvence\,CPU\,/\,(zvolená\,baudová\;frekvence * 16)) - 1 \qquad (1)$$

Tento výsledek se v programu zapíše do registrů vyžadovaných pro UART komunikaci, avšak zápis podporuje vložení pouze celých čísel. Frekvence MCU je 16MHz, nastavená baudová frekvence 76800 bitů za sekundu. Přepočítaná dělička frekvence mikrokontroléru je tedy 12,02083¯3 a po zaokrouhlení 12. Po zpětném přepočtení z frekvence krystalu a zaokrouhlené(zadávané) hodnoty děliče vyjde efektivní(generovaná) baudová frekvence. Dovolená odlišnost těchto frekvencí je dána počtem bitů v jednom paketu. BeagleBoard-xM vysílá v jednom paketu nejdříve start bit, poté sedm data bitů, paritní bit a naposled 2 stop bity. Za předpokladu, že bude paket přijat bez chyby je tedy možná maximální chyba půl bitu na paket. Za těchto podmínek je možné z posílaných jedenácti bitů v paketu dosáhnout maximální chyby přibližně 4,5% mezi frekvencemi. Frekvence MCU vydělená zaokrouhlenou děličkou vyšla 76923 bitů za sekundu a chyba oproti požadované je 0,16%, což je menší, než maximální dovolená chyba. Parametry přenosu (zvolená baudová frekvence vzhledem ke zvolenému počtu datových bitů, stop bitů a frekvenci krystalu) jsou zvoleny vhodně a vyhovují požadavkům pro bezchybný přenos. Zadaná hodnota baudové frekvence je pouze frekvence na které spolu zařízení komunikují.

4.3.4 Programový kód pro UART komunikaci

Komunikaci lze provádět ve dvou režimech. Pomocí dotazů nebo přerušení[16]. Režim dotazů má výhodu kompaktního kódu, avšak během komunikace je CPU zaneprázdněno a místo aplikace probíhá komunikace. UART ovládaný pomocí přerušení běží v pozadí aplikace a tudíž je možné přijímat data a u toho vykonávat další funkce. Pro generování PWM signálů je výhodnější použít režim přerušení, protože je kladen důraz na přesnost generovaných signálů. Kdyby komunikace v režimu dotazu přerušila program v průběhu změny aktuální hodnoty signálu, docházelo by k výrazným rozdílům generovaných signálů od požadovaných a mohla by tak zapříčinit nežádaný chod elektromotorů nebo dokonce jejich poškození. Pro chod v režimu přerušení je nutné před inicializací UART nastavit velikost bufferů pro odchozí a příchozí data. V tomto případě stačí nastavit velikost bufferu jen pro příchozí data, protože není potřeba z ATmegy vysílat. Velikost bufferu pro příchozí komunikaci se nastaví předáním požadované hodnoty do proměnné UART_RX_BUFFER_SIZE. Tato hodnota musí být n-tou druhou mocninou, v jiném případě kompilátor zahlásí chybu a kompilace neproběhne úspěšně. Velikost jednoho paketu je 11 bitů, proto je zvolená velikost bufferu 16 bitů.

Známe tedy parametry potřebné pro příjem regulačních požadavků prostřednictvím UART. Na začátku programu je nutné inicializovat vyžadované hodnoty registrů ATmegy16 pro sériovou UART komunikaci a také ji povolit. Program je psán v jazyce C a z něho je dále překompilován do strojového kódu, který je nahrán do datové paměti, jenž MCU využívá. Zdrojový kód pro správné nastavení registrů

mikrokontroléru pro příjem dat a načtení přijímaného znaku do nastavené proměnné v jazyce C vypadá následovně:

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

Příkaz #include <avr/io.h> vloží do programu hlavičkový soubor obsahující jednotlivé vstupně-výstupní (I/O) definice pro různá AVR zařízení, které jsou specifikovány kompilačním příkazem „–mmcu=“ [20]. Jedná se o sjednocení definicí funkcí z jednotlivých <avr/ioXXXX.h> hlavičkových souborů, které by neměly být nikdy vkládány samostatně. Některé názvy registrů jsou shodné pro různá AVR zařízení a jsou definována v hlavičkovém souboru <avr/common.h>, který je v <avr/io.h> obsažen. Hlavičkový soubor interrupt.h obsahuje instrukce požadované pro přerušení.

```
#define UART_BAUDRATE 76800
#define F_CPU 16000000
#define BAUD_PRESCALE(((F_CPU / (UART_BAUDRATE*16))) - 1)
```

Příkaz #define „proměnná“ „hodnota“ nastaví danou proměnnou na uvedenou hodnotu. Za hodnotu lze také dosadit dříve definovanou proměnnou nebo také vzorec, jehož výslednou hodnoty bude nabývat definovaná proměnná. První dva #define příkazy definují baudovou frekvenci a frekvenci použitého krystalu. Tyto frekvence nejsou dále v programu použity, ale jsou nutné pro následující výpočet a jejich definování je pouze z praktických důvodů, kdyby se uživatel rozhodl použít hodnoty jiné. Třetí příkaz provádí výpočet baudového děliče, se kterým dále pracuje procesor za účelem generování komunikačních signálů na dané baudové frekvenci. V přímé závislosti na použitém krystalu a zvolené baudové frekvenci je dle uvedeného vzorce vypočtena a na jednotky zaokrouhlena hodnota děliče. Díky zaokrouhlení bude generovaná baudová frekvence odlišná od požadované, ale v mezích zachování bezchybného přenosu.

```
int main (void)
{
char ReceivedByte; // proměnná, do které se uloží přijatý byte
UCSRB |= (1 << RXEN);
UCSRC |= (1 << USBS) | (1 << UCSZ1) | (1 << UPM1);
UBRRH = (BAUD_PRESCALE >> 8);
UBRRL = BAUD_PRESCALE;
```

V hlavní funkci programu main() je potřeba nejdříve nastavit registry mikrokontroléru, vyžadované pro použití UART rozhraní. Každý registr obsahuje 8 flagů a každý flag může nabývat dvou stavů a je unikátní [21]. Registry mohou tedy nabývat hodnot 0 až dva na počet flagů. Význam flagů je popsán v datasheetu mikrokontroléru ATmega16 [10]. Flagy jsou standardně nastaveny na logickou nulu a proto je nutné přenastavit flagy u kterých je potřeba logická 1. Pro správnou funkci rozhraní musí být nastaveno :

- Flag RXEN na logickou 1 pro povolení příjmu dat a zahájení UART komunikace. Flag RXEN je obsažen v UCSRB registru.

- Počet data bitů vysílaných v paketu se nastaví, jako kombinace tří flagů UCSZ2, UCSZ1, UCSZ0. Pro případ sedmi data bitů v paketu tomu odpovídá kombinace UCSZ1 = 1 a UCSZ2, UCSZ0 = 0. Flag UCSZ1 je obsažen v registru UCSRC.
- Počet stop bitů v paketu je nastaven flagem USBS. Pro dva stop bity bude flag nabývat hodnoty 1. Flag je obsažen v registru UCSRC.
- Nastavení kontroly parity se nastavuje pomocí dvou flagů UPM1, UPM0 a může být zakázána nebo povolena s ohledem na sudou, či lichou paritu. Pro případ povolení sudé parity odpovídá kombinace nastavení flagů UPM1 = 1 a UPM0 = 0. Flagy jsou obsaženy v registru UCSRC.
- Nastavení asynchronního přenosu se nastaví flagem UMSEL na hodnotu 0 a je obsažen v registru UCSRC.
- Zadání hodnoty baudového děliče se provádí do UBRR registru o velikosti 16 bitů, který je rozdělen na dva registry UCSRH a UCSRL o velikosti 8 bitů. Registru UBRR může tedy obsahovat hodnotu děliče do 65536, kde do registru UCSRL se zapíše 8 prvních (low) bitů a do UCSRH dalších 8 high bitů. Pro nastavení registru UBRR na hodnotu 12 je tedy potřeba nastavit pouze UCSRL registr právě na hodnotu 12.

```
    for (;;)
    {
       while ((UCSRA & (1 << RXC)) == 0) {};
       ReceivedByte = UDR;
    }
}
```

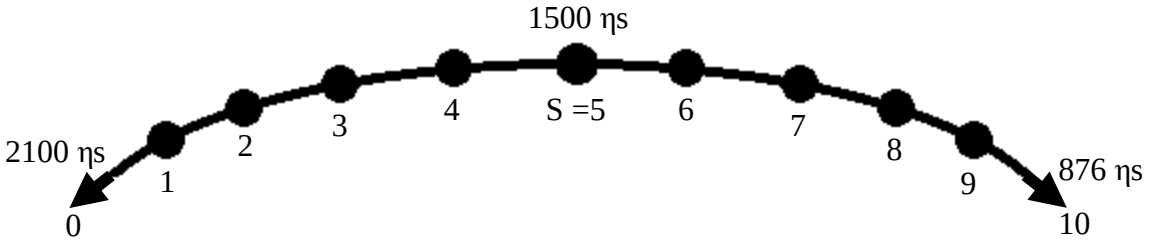
V této nekonečné smyčce se neustále po sobě opakuje cyklus typu while. V podmínce se nachází registr UCSRA, který obsahuje 8 flagů diagnostikujících různé chyby přenosu. Cyklus se tedy provede pouze, pokud při přenosu nenastala chyba. Zároveň s touto podmínkou musí u cyklu while platit, že hodnota flagu RXC nesmí mít hodnotu jedna. RXC nabývá hodnoty 1, když jsou nepřečtená data v přijímacím bufferu a automaticky je změněn na 0, když jsou data z registru UDR přečtena. Cyklus while tedy načte přijatá data z UDR registru do proměnné ReceivedByte a to pouze za podmínky bezchybného přenosu a přítomnosti nepřečtených přijatých dat v bufferu.

4.3.5 Programový kód pro vyhodnocení regulačního požadavku

ATmega je nyní schopna přijímat regulační požadavky a je třeba napsat kód pro jejich vyhodnocení. Nejprve je ale nutné stanovit rozsah, ve kterém se bude rychlost a natočení regulovat, počet kroků v rozsahu a frekvenci regulací. Regulace by neměla být moc častá, aby motor nezatěžovaly frekventované skokové rázy a také ani občasná, protože by se projevila velká prodleva mezi regulačním požadavkem a regulací. Jako vhodná doba, za kterou je automobil schopen zrychlit z nulové rychlosti na rychlost maximální při neustálém regulačním požadavku na zvýšení rychlosti, byl zvolen časový

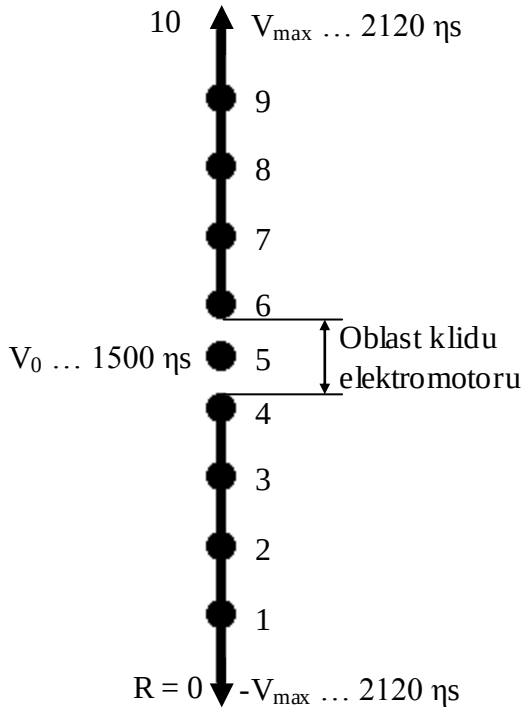
úsek jedné sekundy. Tento čas byl zvolen také pro dobu natočení kol ze střední do krajní polohy. Mezi střední a krajní polohou byly zvoleny čtyři stupně rychlostí a natočení. Pokud je tedy neustále vysílán požadavek na zrychlení automobilu, regulace je nastavena tak, aby se otáčky motoru regulovaly co jednu pětinu sekundy a z nulové hodnoty dosáhnou krajní za pět regulačních cyklů. Celý rozsah se tedy rozdělí na 11 bodů.

Následující schéma představuje stavy natočení kol. Časové konstanty doby náběžných hran pro střední a krajní stavy servomotoru jsou převzaty z tabulky(Tab. 6). Informaci o aktuálním stavu servomotoru nese proměnná „S“, která v krajních stavech nabývá hodnoty 0 nebo 10 a ve střední poloze nabývá hodnoty 5, což je její inicializační hodnota. Rozdíl šířky hran generovaných PWM signálů mezi jednotlivými stavy natočení je zaokrouhlen na 100 ηs. Zobrazení všech stavů natočení je na následujícím obrázku (Obr.13):



Obr. 13: Schéma stavů natočení s časovými délkami horních hran PWM signálu pro mezní a střední stav.

Následující schéma představuje schéma se zvolenými rychlostními stavy automobilu. Jeho princip je obdobný, jako u schématu předchozího s tím rozdílem, že střední poloha má výrazný interval možných dob trvání náběžných hran pro stav, kdy elektromotor nemá otáčky. Tento interval je dle tabulky (Tab. 6) mezi časy, kdy servomotor začíná cukat. Jako časovou konstantu pro generování nejmenšího počtu otáček elektromotorem v daném smyslu se volí časová konstanta pro klidný pohyb elektromotoru dle tabulky (Tab. 6). Informaci o požadovaném rychlostním stavu elektromotoru nese proměnná R. Rozdíl šířky hran generovaných PWM signálů mezi jednotlivými stavy rychlosti je zaokrouhlen na 100 ηs. Na následujícím obrázku (Obr. 14) je schéma s vyznačenými stavy požadované rychlosti a vyznačené doby trvání horních hran generovaného PWM signálu.



Obr. 14: Schéma stavů požadované rychlosti s časovými délkami horních hran PWM signálu pro mezní a střední stav. Oblast klidu elektromotoru je rozmezí, ve kterém elektromotor nemá otáčky nebo začíná cukat.

Aby se proměnné R a S regulovaly co pětinu sekundy, je vhodné využít přerušení prostřednictvím časovačů(timeru). Samotné generování PWM signálu je obstaráno obdobným způsobem, je tedy potřeba brát ohled na priority těchto přerušení. Regulace je méně důležitá, než generování PWM, protože při jejím zanedbání nedochází k poškození elektromotorů. Přerušení pro změnu stavových proměnných R a S má tedy nejmenší prioritu. Dle datasheetu [10] má nejmenší prioritu přerušení s vektorem s vyšší programovou adresou. Hodnoty adres jednotlivých vektorů se nachází také v datasheetu pro ATmega16. Z vektorů přerušení časovačů má tedy nejvyšší prioritu vektor přerušení časovače TIMER2, poté TIMER1 a nejmenší prioritu má TIMER0.

V úvodu programu je potřeba inicializovat TIMER0 s potřebnými parametry. Registr TIMERu obsahuje tři flagy, CS00, CS01 a CS02, jejichž kombinace udává zdroj hodinového signálu a jeho dělení. Význam jednotlivých kombinací je obsažen v datasheetu [10]. Stav čítače je uložen v registru TCNTx daného TIMERU. TIMER0 je osmibitový čítač, může tedy dosáhnout maximální hodnoty 255. Při nastavení TIMERU jako inkrementálního bez dělení s vlastním hodinovým zdrojem, je při každém

hodinovém pulzu, čili pulzu generovaném krystalem, inkrementována hodnota čítače o jedna. TIMER0 lze využít pouze na přerušení prostřednictvím jeho přetečení, čili když jsou všechny flagy registru TCNT0 nastaveny na „1“ a má tedy hodnotu 255, je vygenerováno přerušení časovače přetečením, TIMER se nastaví na hodnotu 0 a inkrementuje při hodinových pulsech od nuly. Při frekvenci krystalu 16MHz je za pětinu sekundy generováno $16 * 10^6 / 5$ hodinových pulzů, což činí $3,2 * 10^6$ hodinových pulzů. Bez nastavení děličky hodinového zdroje by čítač o kapacitě 255 přetekl 12549 krát a s nastavením nejvyššího dělení 1024 krát by přetekl 61 krát. Dělení by šlo rozšířit prostřednictvím pomocné proměnné a podmínky regulace na ní závislé ve vlastní funkci přerušení, avšak při své nízké prioritě by funkce nemusela proběhnout a regulace by se neprováděla. Funkce obstarávající regulaci je proto vložena do jedné z funkcí pro generování PWM signálu. Obě funkce pro PWM probíhají s konstantní periodou, pro elektromotor činí perioda 20ms a pro servomotor 50ms. Perioda změny stavových konstant činí 200ms. Vhodnější je regulační funkci vložit do funkce s periodou 20ms, protože funkce s periodou 50ms jí přeruší s menší pravděpodobností a také po přechodu z horní hrany, protože PWM signál má spodní hranu podstatně delší a není šance, že by se funkce v tuto dobu přerušila sama. Je potřeba tedy funkci napsat tak, aby se při každé desáté sestupné hraně provedla regulační funkce. Čítač s největší kapacitou je TIMER1. TIMER1 je čítač o kapacitě dvou bytů, může tedy dosahovat hodnot až 65535. Při nastavení čítače s děličkou hodinového signálu osmi je inkrementován za 20ms o 40000, protože 20ms = 0,02s a při 16ti miliónech pulzů za vteřinu jich proběhne za 20ms $16 * 10^6 * 0,02 = 32 * 10^4$ => příliš moc, proto při použití děličky osmi $32 * 10^4 / 8 = 4 * 10^5 < 65535$ => dělička o velikosti osmi vyhovuje. Pro přerušení po 20ms je tedy nutné aby čítač přetekl po inkrementaci o $4 * 10^5$ při dělení hodinových pulzů osmi. Funkce, přerušení a nastavení čítače tedy vypadá následovně. Uvedená část kódu zatím neobsahuje funkci pro generování PWM, která bude v závislosti na hodnotě příslušné stavové konstanty nastavovat TIMER1 na požadovanou hodnotu. Funkce pro změnu stavových konstant se provede vždy na sestupné hraně generovaného PWM o periodě 20ms a tím bude zaručena perioda 20ms pro funkci určené pro změnu stavových konstant.

```
TCNT1 = 0x63BF;
```

Dvoubytový registr čítače je nastaven na hodnotu (hodnota přetečení – počet inkrementací do přetečení při děličce 8, frekvenci 16MHz a době do přetečení 20ms = stav čítače) 65535-40000=25535.

```
TCCR1 |= (1 << CS11) ;
```

Nastavení flagu CS11 registru TCCR1 pro děličku = 8 dle kombinací uvedených v datasheetu [10].

```
TIMSK |= (1 << TOIE1);
```

Povolí použití vektoru přerušení čítače.

```
sei();
int cykly = 0;
```

Povolí globální přerušení programu a nastaví počet provedených cyklů do změny stavových konstant na nulu.

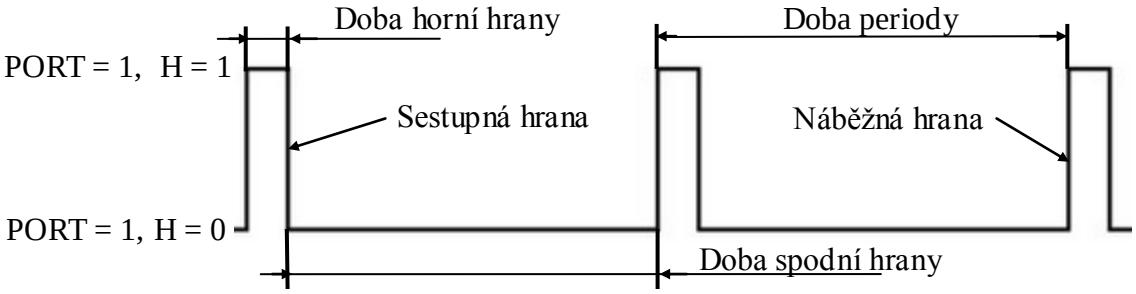
```
SIGNAL(SIG_OVERFLOW1)
   {
   Část přerušení generující sestupnou část hrany
   {
   if (cykly < 10)

   cykly++;
   else
   {
   cykly = 0;
   if (data != 5 && data !=null)
         {if (data >= 7 && R > 0)
               R--;
         else if (data <= 3 && R < 10)
               R++;
         if (data == 2 || data == 4 || data == 8 || && S > 0)
               S--;
         else if (data == 3 || data == 6 || data == 7 && S < 01)
               S++;}}}}
```

Podmínka „if (cykly < 10)“ obstarává, aby se regulace provedla každou pětinu sekundy dle výše uvedených výpočtů. Každou pětinu sekundy se tedy kontroluje, jakou hodnotu má proměnná data, která byla přijata ze řídícího systému a na základě logiky regulačních požadavků uvedené v Obr.(Obr. 11) vyhodnotí prostřednictvím podmínek if požadovanou změnu stavových konstant a provede ji. Podmínky obsahují také nerovnosti, jejichž prostřednictvím je hlídáno, aby stavové konstanty zůstaly ve zvoleném rozsahu.

4.3.6 Programový kód pro generování PWM signálů

PWM znamená v překladu Pulsně Šířková Modulace a v případě ovládání elektromotorů se jedná o digitální dvoustavový signál. Napěťová úroveň signálu je pevně daná a ovládat lze prostřednictvím šířky horních hran generovaného signálu. Stavové hodnoty šířek horních hran pro jednotlivé elektromotory jsou uvedeny v tabulce tab.(Tab. 6). Součet doby spodní a horní hrany signálu musí dát dohromady dobu periody signálu, aby byl motor napájen napětím pro požadované funkce. Vlastní generování PWM signálu je provedeno pomocí střídavého přepínání jednoho PORTu, na jehož výstup je zapojen datový vstup elektromotoru. Na následujícím obrázku (Obr. 15) je schéma ovládacího PWM signálu



Obr. 15: Schéma PWM signálu s vyznačenou logickou a popisem jednotlivých hran.

Přerušení je potřeba upravit tak, aby po inicializaci v náběžné hraně (příslušný port musí být nastaven na „1“ a jako výstup) setrval PORT ve stejném stavu dobu potřebnou dle stavových konstant a poté se vyvolalo opět přerušení, které způsobí sestupnou hranu signálu pomocí funkcí pro změnu stavu příslušného PORTU a navíc se provede v místě sestupné hrany funkci změny stavových konstant. Pro identifikaci právě generovaného typu hrany slouží konstanta „H“. Následující část programu slouží pro generování PWM signálu pro ovládání rychlosti prostřednictvím elektromotoru HIMOTO RS540 dle výše uvedených parametrů a zvolených principů.

```
#define setbit(PORT,PIN)     PORT |= 1<<PIN
#define clrbit(PORT,PIN)     PORT &= ~(1<<PIN)
```

Definuje funkce pro nastavení PINů daných PORTů. Funkce setbit PIN nastaví na „1“ a clrbit PIN nastaví na „0“.

```
TCNT1 = 0xF447;
```

Jako inicializační hodnota čítače je nastavena hodnota, aby do doby přetečení uplynulo 1,5ms, což je doba trvání horní hrany signálu pro střední stav elektromotoru.

```
TCCR1 |= (1 << CS11) ;
TIMSK = |= (1 << TOIE1);
sei();
int cykly = 0;
H20=0
SIGNAL(SIG_OVERFLOW1)
{
      if (H20 == 1)
            { clrbit(PORTA,0);
            switch (true)
                  { case (R == 5):
                        TCNT1 = 28535; H20 = 0; break;
                   case (R > 5):
                        TCNT1 = 28535-220-((R-5)*200);H20 =
                        0; break;
                  case (R < 5):
                        TCNT1 = 28535+160+((5-R)*200);H20 =
                        0; break;}
            (funkce pro změnu stavových konstant uvedená
            výše)}
      else
            { setbit(PORTA,0);
            switch (true)
                  { case (R == 5):
                        TCNT1 = 62535; H20 = 0; break;
                   case (R > 5):
                        TCNT1 = 62535+220+((R-5)*200);H20 =
                        0; break;
                  case (R < 5):
                        TCNT1 = 62535-160-((5-R)*200);H20 =
                        0; break;}}
```

Inicializační část uvedeného programu uvádí signál do stavu náběžné hrany. H = 0, což platí pro tento případ a doba do přerušení je nastavena na 1,5ms. Podle aktuálního stavu signálu vyplývajícího z proměnné H20 a hodnoty stavové proměnné R se dále ve funkci přerušení přepočítá a změní aktuální hodnota TCNT1 čítače TIMER1 a změní se aktuální stav výstupu PORTU A na výstupu PORTA7. Význam hodnot v přepočtu čítače:

- Čím větší hodnota TCNT1, tím dříve dojde k přerušení
- TCNT1=28535 … přerušení za 18,5ms
- TCNT1=62535 … přerušení za 1,5ms, tyto hodnoty platí pro v=0 a R = 5
- 220 = 0,11ms; 160=0,08ms … časové rozmezí pásma nulové rychlosti kolem střední hodnoty trvání hran ve smyslu tendence ke kladné a záporné rychlosti
- ((R-5)*200) = ((R-5)*0,1ms) … platí pro v>0, jedná se o n-násobek přírůstku doby trvání hran o přírůstku 0,1ms
    - Pro nejmenší kladnou rychlost platí: R = 6 => n =1

  - Pro maximální kladnou rychlost: R = 10 => n=5; 5*0,1ms + 0,11ms + 1,5ms = 2,11ms, což dle schématu(Obr. 14) odpovídá šířce horní hrany signálu pro maximální rychlost elektromotoru
- Princip ostatních výpočtů je obdobný

Softwarové generování PWM signálu pro servomotor vyžaduje užití druhého čítače. MCU ATmega16 nemá k dispozici dva 16-ti bitové čítače, jenž jsou schopny čítat po delší časový úsek. Je tedy nutné použít čítač osmibitový a pro dobu delší, než je čítač schopen čítat i s využitím největší děličky je nutné využít pomocné proměnné a podmínek, stejně jako tomu bylo u funkce na změnu stavových proměnných s proměnou „`cykly`“. Aby se proměnná s jistotou inkrementovala při každém intervalu, je nutné tento signál PWM generovat prostřednictvím přerušení s vyšší prioritou, než tomu bylo u signálu pro elektromotor. Vyšší prioritu má vektor přerušení při přetečení čítače TIMER2. Maximální doba do přerušení, jenž je osmibitový čítač schopen obsáhnout při frekvenci krystalu 16MHz a použití děličky čítače o hodnotě 1024 je 16,32ms. Signál pro servomotor nepracuje s výrazným místem klidu, jak je tomu u elektromotoru, přírůstek doby trvání hran signálu se proto rovnou přičítá nebo odečítá od střední doby trvání hran. Následující část programu slouží pro generování PWM signálu pro ovládání natočení kol prostřednictvím servomotoru HIMOTO 3kg dle výše uvedených parametrů a zvolených principů.

```
TCNT2 = 0xF4;
```

Jako inicializační hodnota čítače TIMER2 je nastavena hodnota, aby do doby přetečení uplynulo 1,5ms, což je doba trvání horní hrany signálu pro střední stav servomotoru.

```
TCCR2 |= (1 << CS20) | (1 << CS22);
```

Dělička je dle kombinace nastavení CS2 registru nastavena na 1024.

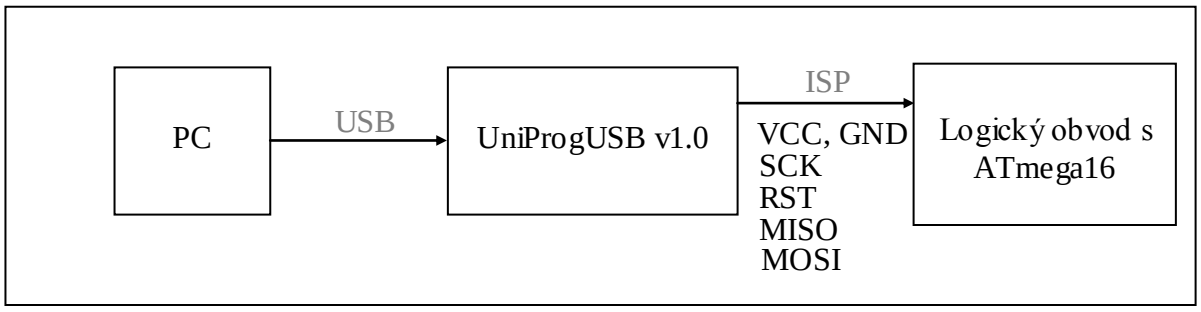
Pokud by se měl servomotor nacházet v poloze s nejkratší dobou trvání horní hrany, tak poté následuje část signálu s nejdelším časem trvání spodní hrany. V krajních stavech elektromotoru trvá tato spodní hrana 49,1ms nebo 47,9ms. Při generování spodní hrany se využije pomocná konstanta, která se inkrementuje po každých 8ms. Při páté inkrementaci je spodní hrana v čase 40ms, kde do krajních stavů zbývá čas 9,1ms nebo 7,9ms. Osmibitový časovač je schopen čítat do těchto časů, tuto metodu lze tedy použít. Při generování spodní hrany je proto nejdříve nutné pětkrát inkrementovat pomocnou konstantu po 8ms a poté nastavit časovač na dobu do přerušení dle požadovaného stavu servomotoru. Uvedený zdrojový kód je určen ke generování PWM signálu pro ovládání servomotoru, využívající TIMER2 a pomocnou konstantu `pomocna`. Princip je obdobný, jako u zdrojového kódu určeného ke generování PWM signálu pro ovládání elektromotoru.

```
H50=0
TIMSK |= (1 << TOIE2);
pomocna = 0;
SIGNAL(SIG_OVERFLOW2)
{
      if (H50 == 1){
         if (pomocna >= 5)
            { clrbit(PORTA,1);
            switch (true)
                  { case (S == 5):
                        TCNT2 = 122; H250 = 0; break;
                   case (S > 5):
                        TCNT2 = 122-((R-5)*2);H50 = 0; break;
                  case (S < 5):
                        TCNT2 = 122+((5-R)*2);H50 = 0;
                        break;}}
         else TCNT2 = 120; pomocna++; break;
            }

      else
            { setbit(PORTA,1);
            switch (true)
                  { case (S == 5):
                        TCNT2 = 231; H50 = 0; break;
                   case (S > 5):
                        TCNT2 = 231+((R-5)*2);H50 = 0; break;
                  case (S < 5):
                        TCNT2 = 231-((5-R)*2);H50 = 0;
                        break;}}
```

4.3.7 Kompilace do strojového kódu a programování MCU

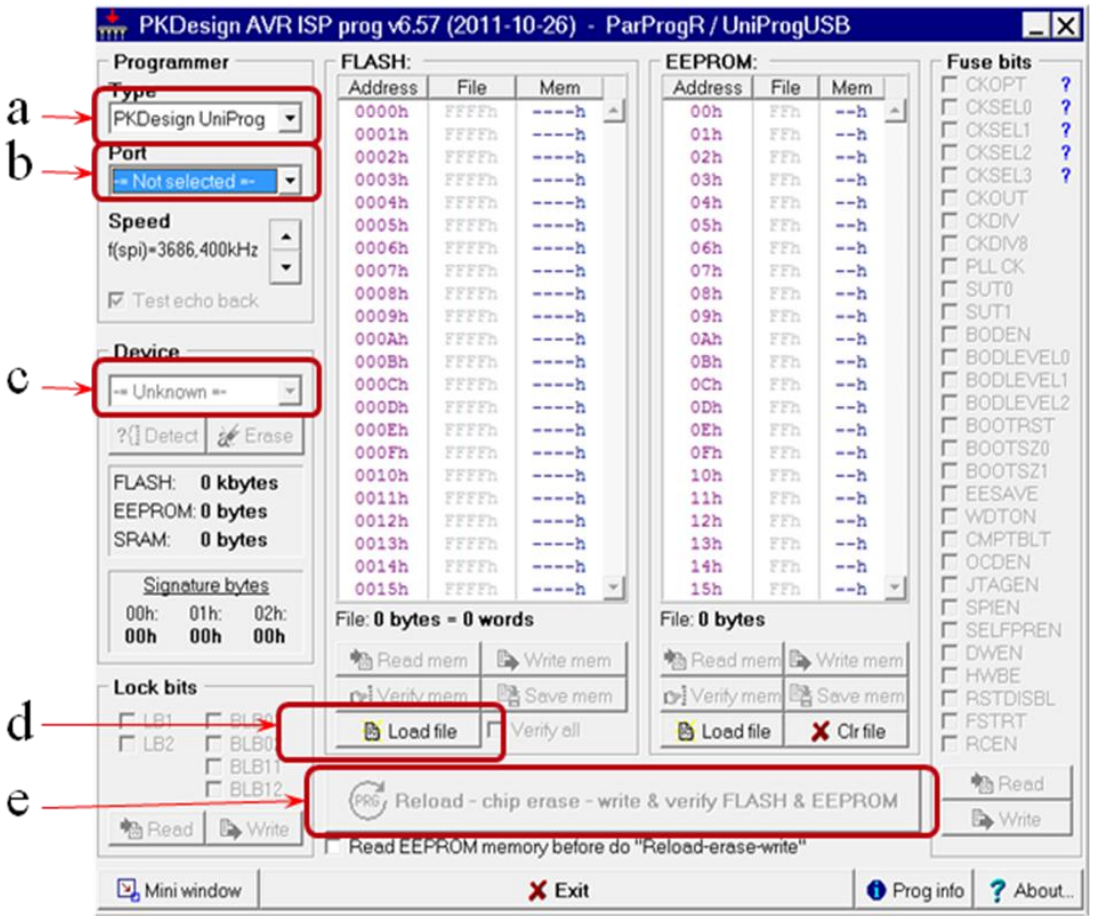
Výše zmíněné zdrojové kódy je potřeba přepsat do funkčního celku a překompilovat do strojového kódu. Jako vývojové prostředí je použit software firmy Atmel AVR Studio 5. Před samotnou tvorbou zdrojového kódu je nutné zadat programu, pro jaký výrobek firmy Atmel je kód určen a jakým způsobem se bude do strojového kódu kompilovat. Jako výrobek byl zadán mikrokontrolér ATmega16 a program překompiloval uvedený zdrojový kód do souboru s příponou .hex. Tento soubor obsahuje data, pomocí kterých ATmega16 plní funkci přijímání regulačních požadavků prostřednictvím UART rozhraní a generuje příslušné PWM signály. Dle principu Harvardské architektury, na kterém ATmega16 pracuje, je nutné nahrát tyto data do datové paměti, jenž využívá. K dispozici je více rozhraní, prostřednictvím kterých lze přenos dat provést. Nahrání dat z PC do programového obvodu je realizováno prostřednictvím ISP rozhraní, konkrétně prostřednictvím hardwarového programátoru UniProgUSB, který zprostředkovává vzájemnou komunikaci. Data jsou z osobního počítače přenášena přes rozhraní USB do programátoru. Z programátoru je potřeba propojit příslušné výstupy ISP rozhraní s příslušnými vstupy MCU ATmega16. Výstupy programátoru pro programování prostřednictvím ISP rozhraní jsou uvedeny v datasheetu programátoru [22] včetně názvů signálů a je třeba propojit je se stejně označenými vstupy MCU ATmega16, jejichž označení je uvedeno v jeho datasheetu [10]. Programátor není napájen prostřednictvím USB, nýbrž z logického obvodu MCU. Ten je napájen prostřednictvím UART rozhraní, je tedy nutné mít ho zapojeno do USB PORTu. Následující schéma (Obr. 16) zobrazuje propojení ISP rozhraní pro nahrání programu do datové paměti MCU včetně signálů pro komunikaci mezi programátorem a MCU.



Obr.16 : Schéma propojení ISP komunikačního rozhraní pro programování MCU ATmega16 včetně signálů mezi programátorem a mikrokontrolérem.

Po propojení programátoru s osobním počítačem je třeba nainstalovat příslušné ovladače pro správu programátoru. Pro programování je použit operační systém Windows 7 x64, pro který je zdarma k dispozici software zprostředkovávající vlastní komunikaci. Ten musí mít informace o pozici připojení programátoru do příslušného USB PORTu, aby s ním mohl komunikovat. Originální návod na instalaci potřebných ovladačů je na adrese :http://www.ftdichip.com/Documents/InstallGuides.htm.

Po nainstalování ovladačů virtuálních PORTů a správném propojení programátoru je třeba zadat programu všechny informace, jenž jsou zvýrazněny v následujícím obrázku (Obr. 17), kde je znázorněn postup (a až e) programování pomocí programu PKDesign AVR ISP prog v6.57, který je k dispozici zdarma ke stáhnuti na [23].

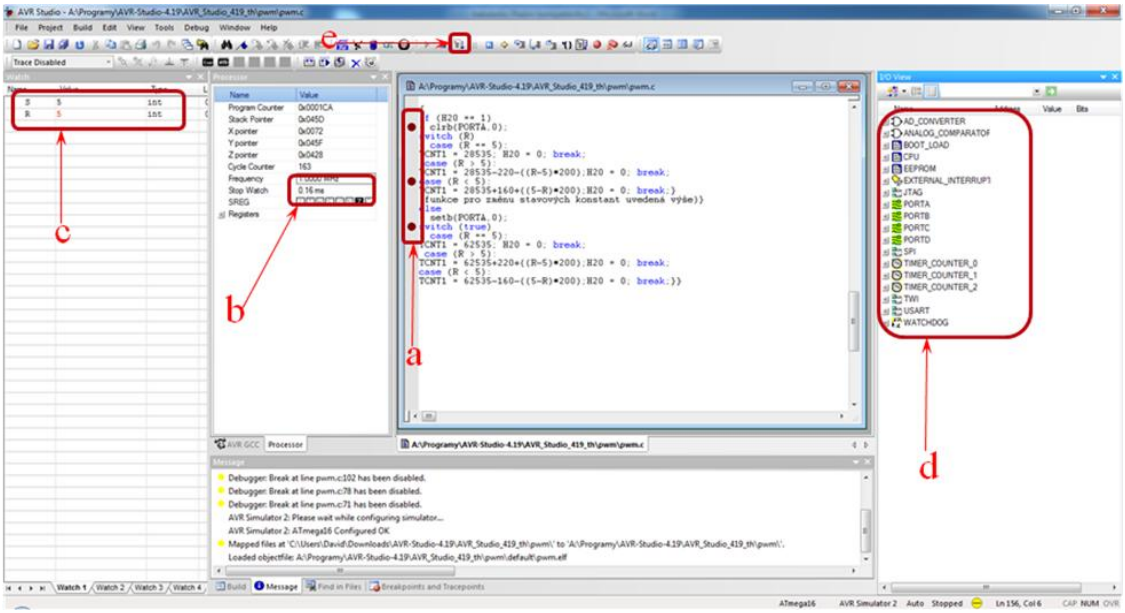


Obr.17: Ukázka programovacího prostředí programu PKDesign AVR ISP prog.v6.57. a) volba programátoru, b) volba virtuálního portu, na kterém je nainstalován programátor, c) volba programovaného zařízení, d) tlačítko pro nahrání .hex souboru, který je potřeba nahrát do FLASH paměti, e) tlačítko pro vymazání příslušné paměti, následné zapsání dat do této paměti a ověření správnosti odeslaných dat.

Pokud je program přenesen do paměti bez indikovaných problémů, MCU začne nahraný program vykonávat ihned po připojení požadovaného napájení. Na nultém a prvním PINu PORTu A je v tuto chvíli generován PWM signál a prostřednictvím UART rozhraní probíhá sériová komunikace dle nastavených parametrů.

5 Experimentální otestování systému

Aby nedošlo k poškození elektromotorů, je nejprve nutné navržený program pro MCU otestovat. K tomu je využito ladící prostředí vývojového software firmy Atmel AVR Studio v4.19. Software dokáže simulovat chod nastaveného MCU pod testovaným programem. Na následujícím obrázku je zobrazeno vývojové prostředí softwaru AVR Studio v4.19 a jsou v něm vyznačeny tlačítka a funkce, které jsou k otestování používány.



Obr. 18: Ladící prostředí Softwaru AVR Studio v.4. Využívané funkce a tlačítka: a) nastavitelné breakpointy, b) simulovaný reálný čas c) nastavitelné sledované proměnné, d) indikace stavů všech registrů simulovaného zařízení

Ve výše uvedeném ladícím prostředí se testovalo generování příslušných PWM signálů. Nevýhoda tohoto prostředí je v simulované frekvenci krystalu, která může být nastavena na maximálně 8MHz. MCU ATmega16 využívá krystal s frekvencí 16MHz, což znamená, že čítače nastavené na dvakrát rychlejší krystal přetečou při simulaci za dvojnásobný čas, než bylo předpokládáno.

Při vlastním testování se nejprve nastaví breakpointy (Obr. 18-a) na místa programu, kde dochází ke změně logické úrovně PINů, na kterých se generují PWM signály. Po spuštění simulovaného chodu programu se chod automaticky zastaví, pokud se zrovna směrník programu nachází na nastaveném breakpointu a poté čeká na opětovné spuštění. V tuto chvíli se generovaný signál nachází na sestupné nebo náběžné hraně. V okně I/O View (Obr. 18-d) se uživatel může podívat na aktuální stav všech registrů MCU a tedy také na aktuální stav PINů. Program během simulace chodu programu simuluje reálný čas (Obr. 18-b). V době náběžné a sestupné hrany se přečte hodnota simulovaného času a když je od aktuální hodnoty odečtena hodnota předchozího času, tak rozdíl tvoří dobu trvání horní nebo dolní hrany. Simulovat jde

také chování při změně stavových proměnných. Po přepsání stavových proměnných v okně Watch (Obr. 18-c) simulace okamžitě pracuje s hodnotou proměnných jenž byla změněna. Přepisovat hodnoty libovolných proměnných lze i během chodu simulace programu.

Simulací generovaný PWM signál odpovídá signálu požadovanému s dvakrát delšími spodními a horními hranami. Signál generovaný pomocí osmi-bitového čítače má mírně horší rozlišení, protože při použití děličky není schopen čítat časy menší než 0,016ms.

6 Závěr

V úvodní části této bakalářské práce proběhlo seznámení se s vlastnostmi a výhodami užití jednodeskových počítačů v praxi. Po následném rozdělení SBC podle uvedených konstrukčních kritérií v kapitole 2.2 byly představeny některé používané SBC dnešní doby, na kterých byly demonstrovány parametry moderních SBC za účelem získání bližší představy rozsahu možností jejich využití v praxi.

Zadáním bakalářské práce je navrhnout strukturu řídícího systému automobilu pohybujícím se v autonomním konvoji a jako řídící člen je použit jednodeskový počítač BeagleBoard-xM. Verze tohoto počítače s příponou -xM je zatím nejmodernější a pro ukázku výhod tohoto nového systému je v kapitole 3.1 uvedeno srovnání s verzí předchozí. Pro vhodný návrh řídící struktury systému bylo nutné podrobně prostudovat možnosti tohoto počítače. Konkrétně byla pozornost věnována hardwarovému rozhraní v kapitole 3.2 a v kapitole následující komunikačnímu rozhraní. BeagleBoard-xM byl na základě svých funkcí, parametrů a podporovaných rozhraní označen za vhodný pro plnění žádané funkce v řídícím systému.

V kapitole 4 byla navržená struktura systému podrobně popsána. Byly rozebrány využité vstupní periferie a rozhraní pro ovládání výstupních periferií. V této části bakalářské práce byly popsány parametry elektromotorů automobilu a způsob, jejímž se ovládají. Kapitola 4.3 pojednává o tvorbě softwaru pro hardwarové rozhraní, které ovládá automobil a také je uveden způsob vysílání regulačních požadavků BeagleBoardem-xM prostřednictvím UART rozhraní. Pro tento způsob komunikace byla ověřena vhodnost zvolených parametrů. Regulační požadavky na změnu rychlosti a natočení automobilu jsou vysílány v podobě jednoho znaku a kapitola 4.3.2 pojednává o významu regulačního požadavku na základě hodnoty tohoto znaku. Dále byly rozebrány zdrojové kódy určené pro MCU ATmega16, konkrétně pro příjem informací prostřednictvím UART rozhraní, vyhodnocení přijatého regulačního požadavku a kód za pomocí kterého jsou na PINech generovány požadované PWM signály. Také byl ukázán způsob kompilace navržených zdrojových kódů do strojových instrukcí a přistup, jímž byly nahrány do datové paměti MCU. Kapitola 6 popisuje způsob otestování vlastností navržených programů.

Výsledky testování systému prokázaly, že řídící systém tak jak je navržen může plnit požadovanou funkci. Navržená struktura řídícího systému založená na jednodeskovém počítači BeagelBoard-xM jako řídícím členu může být účelně využita pro jakoukoliv jinou praktickou aplikaci vyžadující řízení na základě dat, získaných z kamery.

Seznam použitých zdrojů

[1] Winn Rosch, *Hardware Bible Fifth Edition*, Que , 1999 ISBN0-7897-1743-3 p. 50-51

[2] Single-board computer: Wikipedia, the free encyklopedia. Wikipedia [online]. 2012, 2013-05-07 [cit. 2013-05-21]. Dostupné z: http://en.wikipedia.org/wiki/Single-board_computer

[3] The SBC (Single Board Computer) vs. motherboards. Stealth.com [online]. 2008, 2008-11-21 [cit. 2013-05-21]. Dostupné z: http://www.stealth.com/phpkb/article-12.html

[4] How does the ARM architecture differ from x86?: Stack Overflow. In: Stack overflow [online]. 2013-02-10 [cit. 2013-05-21]. Dostupné z: http://stackoverflow.com/questions/14794460/how-does-the-arm-architecture-differ-from-x86/14795541#14795541

[5] FUHRMAN, Oliver. Single-board computer: History. Single-board computer [online]. 2013-03-14 [cit. 2013-05-21]. Dostupné z: http://singleboardccomputer.blogspot.cz/2013/03/history.html

[6] ODROID: Hardkernel. HARDKERNEL CO. ODROID [online]. [cit. 2013-05-21]. Dostupné z: http://www.hardkernel.com/renewal_2011/products/prdt_info.php?g_code=G133999328931&tab_idx=1

[7] ArndaleBoard. INSIGNAL CO. ArndaleBoard.org [online]. 2012 [cit. 2013-05-21]. Dostupné z: http://www.arndaleboard.org/wiki/index.php/WiKi

[8] Cstick Cotton Candy. FXI TECHNOLOGIES. Store cstick [online]. 2012 [cit. 2013-05-21]. Dostupné z: http://store.cstick.com/cotton-candy.html

[9] Comparsion of single-board computers: Wikipdia the free encyklopedia. Wikipedia, the free encyklopedia [online]. 2013-02-13 [cit. 2013-05-21]. Dostupné z: http://en.wikipedia.org/wiki/Comparison_of_single-board_computers

[10] ATMEL CO. ATmega16. 2010-10-07, 357 s. Dostupné z: http://www.atmel.com/Images/doc2466.pdf

[11] TEXAS INSTRUMENTS INC. BeagleBoard-xM System Reference Manual. 2010-04-04, 164 s. Dostupné z: http://beagleboard.org/static/BBxMSRM_latest.pdf

[12] Instruction manual: X supermotive pro. Dostupné z: http://www.hobbex.se/internt/artiklar/116378/3957_116378_-_X_SUPERMOTIVE_PRO.pdf

[13] BREMER, Mark, William COLLETO, Joshua COX, Daniel JOANES, Mattias VAN DE HOEF a Samuel ZHANG. Vehicle Saftey System: System Design Document. Německo, 2011-01-07. Dostupné z: http://mvdhoef.wesg.ca/PDF/SDD.pdf. Semestrální práce. McMaster University.

[14] Open a RS232 Serial Terminal Connection. PIXHAWK research project: ETH PIXHAWK: MAV Computer vision [online]. [cit. 2013-05-21]. Dostupné z: https://pixhawk.ethz.ch/tutorials/serial_terminal#beagleboard

[15] Harvard architecture: Wikipedia, the free encyklopedia. Wikipedia, the free encyklopedia [online]. 2010-10-17 [cit. 2013-05-21]. Dostupné z: https://en.wikipedia.org/wiki/Harvard_architecture

[16] ATMEL CO. Using the AVR UART in C. 2002, 3 s. Dostupné z: http://www.atmel.com/Images/doc1451.pdf

[17] Intro to UARTs. EXAR, 2010, 23 s. Dostupné z: http://www.exar.com/search/search.aspx?keywords=intro%20to%20uarts

[18] Krystal. Wikipedia, the free encyklopedia [online]. 2006-07-23 [cit. 2013-05-21]. Dostupné z: http://cs.wikipedia.org/wiki/Krystal_%28elektronika%29

[19] AVR Baud Rate Calculator. Wormfood.com [online]. 2012 [cit. 2013-05-21]. Dostupné z: http://www.wormfood.net/avrbaudcalc.php

[20] AVR device-specific IO definitions. Savannah, the software forge for people committed to free software [online]. 2012-01-03 [cit. 2013-05-21]. Dostupné z: http://www.nongnu.org/avr-libc/user-manual/group__avr__io.html

[21] USART. Geisterstunde.org: Hacking on stuff [online]. [cit. 2013-05-21]. Dostupné z: http://www.geisterstunde.org/avr/usart1.html

[22] PK DESIGN. UniProg-USB v1.0: Uživatelský manuál. 2007-08-20. Dostupné z: http://www.pk-design.net/Datasheets/UniProgUSB_v10_doc_ver_v11_20070820.pdf

[23] PK Design: AVR ISP Programmer. PK Design: Výukové a vývojové systémy [online]. 2011-10-26 [cit. 2013-05-21]. Dostupné z: http://www.pk-design.net/HtmlCz/SoftUtilities.html#AtmelAVR3

Seznam Příloh

CD-R obsahující: Elektronickou kopii tohoto dokumentu ve formátu PDF.